DLP: Achieve Customizable Location Privacy With Deceptive Dummy Techniques in LBS Applications

Jiezhen Tang, *Student Member, IEEE*, Hui Zhu, *Senior Member, IEEE*, Rongxing Lu, *Fellow, IEEE*, Xiaodong Lin, *Fellow, IEEE*, Hui Li, *Member, IEEE*, and Fengwei Wang, *Student Member, IEEE*

***Abstract*—As a straightforward consequence of advances in the Internet of Things (IoT), location-based service (LBS) applications have been pervasive in our daily lives. Nevertheless, since those LBS applications will continuously collect and disclose users' location data, major concerns on privacy leakage are raised. Aiming at the challenge, in this article, we first build up a detect module (DM) and employ it to investigate more than 80% of LBS applications are keen on tracking users. Then, to thwart the threats from those LBS applications, we exploit the deceptive dummy techniques and design a dummy-based location privacy preserving scheme, named dummy location provider (DLP), which comprises three algorithms, namely, Spread, Shift, and Switch. Specifically, Spread and Shift are in charge of generating deceptive dummies and trajectories. And with Switch, users' real locations are replaced with dummy trajectories before being submitted to LBS applications. As a result, users can not only prevent applications from accessing location data arbitrarily, but also avoid being questioned by applications in terms of honesty. Furthermore, to guarantee necessary functions of LBS, DLP offers customizable privacy-preserving strategies for users, which can achieve flexible location data usage control. Finally, our DLP can also attain achievable and effortless deployment over smart devices. Detailed security analysis indicates that DLP resists inference attacks even facing skeptical applications. In addition, for performance evaluation, a DLP application (DLPA) is developed on the Android platform and tested in the real environment, and the extensive experimental results demonstrate that the DLPA is indeed effective and high efficiency in practice.**

***Index Terms*—Dummy location provider (DLP), location privacy preservation, location-based service (LBS), smart device security.**

Manuscript received June 4, 2021; revised August 22, 2021; accepted September 15, 2021. Date of publication September 27, 2021; date of current version April 25, 2022. This work was supported in part by the National Natural Science Foundation of China under Grant 61972304 and Grant 61932015; in part by the National Natural Science Foundation of Shaanxi Province under Grant 2019ZDLGY12-02; and in part by the Technical Research Program of Public Security Ministry under Grant 2019JSYJA01. *(Corresponding author: Hui Zhu.)*

Jiezhen Tang, Hui Zhu, Hui Li, and Fengwei Wang are with the State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an 710126, China (e-mail: jztang@stu.xidian.edu.cn; zhuhui@xidian.edu.cn; lihui@mail.xidian.edu.cn; xdwangfengwei@gmail.com).

Rongxing Lu is with the Faculty of Computer Science, University of New Brunswick, Fredericton, NB E3B 5A3, Canada (e-mail: rlu1@unb.ca).

Xiaodong Lin is with the Faculty of Computer Science, University of Guelph, Guelph, ON N1G 2W1, Canada (e-mail: xlin08@uoguelph.ca).

Digital Object Identifier 10.1109/JIOT.2021.3115849

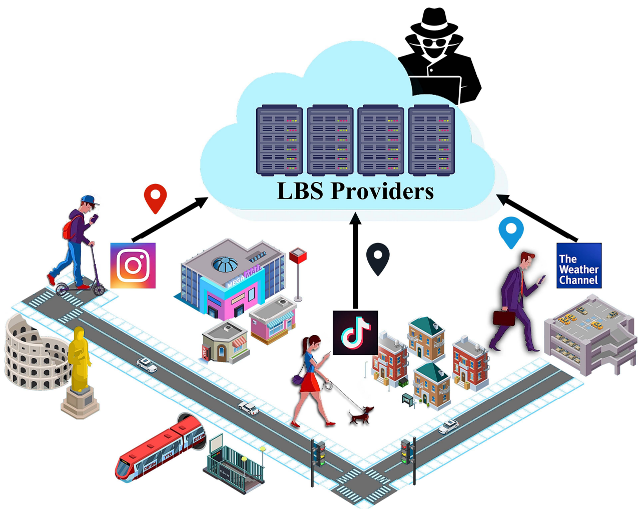


Fig. 1. Conceptual architecture of LBS services and queries.

I. Introduction

LOCATION-BASED SERVICE (LBS), which accesses users' location data to provide various personalized information, has brought great convenience to people in a variety of contexts, including navigation, advertising, social network, etc. Nevertheless, when users enjoy the benefits from LBS-enabled applications, their location privacy has also been threatened. As Fig. 1 depicts, in traditional LBS queries, users' accurate locations are submitted to LBS providers via applications. Thus, the real-time geographical data that users provide can be easily collected by LBS providers and applications. Unfortunately, the data is strongly correlated to users' privacy, since it can be easily used to infer users' personal information, such as identities, home addresses, mobile tracks, and so on. Eventually, leakages of location data may cause computer-assisted crime [1]–[3], worse still, it can do harm on users' property and lives [4]–[6].

To thwart the privacy threats from the source, OS developers of smart devices have attempted to restrict LBS applications' location data abuse by introducing permission architecture, i.e., applications should request corresponding permissions while assessing users' location data. However, the permission architecture has been proved to be a rough strategy [7]–[9]

for users' location privacy preservation due to the following facts: 1) applications can refuse to provide LBS when permission requests are denied [10], which places users in a dilemma of whether to protect their location privacy or accessing expected LBS; 2) applications always apply for superfluous high-level permissions to collect unnecessary precise geographical information [11], for instance, although weather applications can definitely perform forecasts with cities or districts, they are still greedy about users' accurate locations; and 3) applications tend to make every attempt to stay alive, which makes it possible to constantly track devices [12], e.g., users need social media applications invoking location data when posting only, but applications are capable of recording locations even in background services. As a result, the permission model cannot fully satisfy the requirement of protecting users' location privacy, since applications still track users precisely and continuously.

In recent years, various location data preserving schemes have also been proposed [13]–[17], including anonymity, obfuscation, dummy-based schemes, etc. Specifically, anonymity schemes protect one specific user's location by providing requests from multiple users in simultaneous LBS sessions, in which additional servers and suspicious participants need to be introduced. Obfuscation schemes deliberately reduce the precision of coordinates to preserve location data, which can hardly strike a balance between functional LBS and location privacy due to coordinates coarsened or transformed. Dummy-based schemes generate dummy locations and send both dummies and real locations in service requests, resulting in the following issues: 1) the generation of dummies will bring extra computation costs, and queries with such dummies also lead to communication costs increase and 2) it is nearly impossible for any LBS application to provide services for users who simultaneously present numerous locations, e.g., a navigation app can get confused if multiple starting points are used to generate one real-time route. More than aforementioned limitations, proposed the location privacy schemes are lack of effectiveness while facing skeptical LBS applications in practice. Because for one thing, when applications detect users submitting irrational coordinates, they can question users' honesty and suspend LBS. For another, despite mentioned mechanisms, applications may still infer users' real locations from collected data.

In this article, aiming at tackling the challenges mentioned above, we propose a novel dummy-based location privacy-preserving (DLP) scheme, featured with the following properties: 1) adopting deceptive dummy techniques, DLP can provide spatially continuous dummies (trajectories) to mislead applications and dispel their suspicion on users, which achieves protecting location privacy under inference attacks from skeptical applications; 2) DLP offers finer-grained and flexible location data access control, which is able to balance location privacy protection and functioned LBS; and 3) DLP is a system solution which equipped with features of easily-installing, low consumption, and requiring no coordination on LBS providers, platforms or trusted third parties (TTPs). Specifically, the contributions of this article are summarized threefold.

1) For investigating how LBS applications abusing location data, we build a detecting module (DM) on the Android platform to perform the pursuance of applications' invocation on locations. With DM, we analyze 50 hottest mobile applications by testing their location data usage.
2) To protect users from privacy infringement, we propose the DLP scheme with three core algorithms. Specifically, DLP generates deceptively unreal location data (dummies) with the *Spread* algorithm. When third-party LBS applications (adversaries) are detected invoking location data, DLP can intercept location-related methods. Then DLP constructs spatially continuous dummy paths in the *Shift* algorithm, and finally, such data will be deployed to hijacked methods via the *Switch* algorithm.
3) We develop the DLP application (DLPA), which is able to protect users' privacy in stand-alone mode, or run with our optional server-side dummies provider. In simulation and performance evaluation, different categories of LBS applications are involved, and the experiments demonstrate DLPA runs efficiently.

The remainder of this article is structured as follows. In Section II, we recur how applications reach location data in LBS and build up DM to analyze popular applications. Then we formalize the system model, adversaries model, and our basic ideas in Section III. Section IV details the location privacy-preserving scheme, as well as the exhaustive design of DLP. Section V presents several experiments and evaluations of such mechanism. Finally, we describe some related work in Section VI and draw the conclusion in Section VII.

II. LBS Mechanism and Analysis

In this section, we first describe how applications reach location data in LBS, then build the DM to analyze the extent of the abuse of location data, and finally show that LBS applications are greedy to users' location data.

A. LBS Mechanism Over Smart Devices

In this section, we select the Android location service as the representative location architecture [18], [19], with which we can introduce the mechanism of how OS empowers applications to access location data. In the Android location service, *LocationManager* is the central component and entry class, which can be directly accessed by LBS applications. Under *LocationManager*, *LocationProvider* class plays the role of providing geographic location data concretely. Specifically, when applications need to reach users' last known locations or request an update of location data, *LocationManager* should first be invoked. Then, the entry class will require querying for the list of *LocationProvider*. Finally, three providers are employed to get the physical locations indeed, and each provider involves different sets of criteria for service.

1) *GPS Location Provider* determines locations using satellites, which is able to provide high-accuracy locations, but GPS has limitations of requiring a substantial amount of time to get locations and could not work properly where the clear sky is not available [20].

TABLE I
CHINESE APP STORE RANKING (FEBRUARY 25TH, 2021)

App Stores	Market Share	MAU*
Huawei App Market	37.38%	261 389 000
Tencent My App	30.97%	216 556 000
Oppo Software Store	26.53%	185 538 000
VIVO App Store	18.31%	128 016 000

MAU*: Average number of monthly active users.

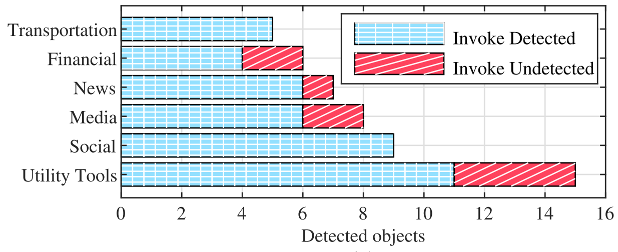
2) *Network Location Provider* is based on availability of cell tower and Wi-Fi access points which retrieve results by means of network lookup, thus devices must be connected to cell towers or in range of Wi-Fi networks. With such a provider, devices are able to process expeditious but inaccurate location determination.
3) *Passive Provider* is utilized to passively receive location updates when other applications or services have requested it.

B. LBS Applications Analysis

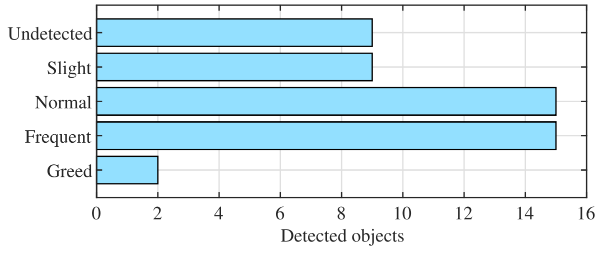
According to our research of the LBS mechanism, whenever any application try to reach location data, it has to invoke location-related methods from *LocationManager* and *LocationProvider* classes. Hence, to analyze applications' abuse of location data, we can: 1) record the invoking procedures; 2) infer the invoker applications; and 3) demonstrate applications' location data usage. To that end, we build up DM. In particular, first, when location-related methods are invoked by any applications, the context of such methods will first be recorded by DM. Then, DM involves *AndroidAppHelper*, an extending *Object* class from the Android system, to analyze grabbed context and reference the information of corresponding applications. Therefore, the very application which invokes location-related methods are exposed. Finally, DM reports the timestamps and invoker applications of invoking procedures. As the outcome, DM can estimate the frequency of location data collecting by each application, which significantly contributes on demonstrating the extent of applications' location data abusing.

In the experiment, we select 50 applications to investigate their location data usage. Specifically, we select the most universal Android applications markets [21] as Table I presents, and totally pick up 143 top downloaded applications from them and Google Play Store. By cross-comparing these applications, 50 most frequently appearing applications in all five markets are extracted and divided by functions into six groups. Apparently, the applications chosen in our research possess high market share and cover sufficient categories, it is a rational argument that they represent the most popular applications worldwide.

By detecting and logging applications' location-related methods invoking status in 24 h, we draw the conclusion in Fig. 2(a). The result indicates that 41 of the 50 applications request location data in background services. To depict the frequency of detected applications invoking location data (invoking location data times per 24 h), we classify them into five groups through accumulate statistics: 1) undetected (0);



(a)



(b)

Fig. 2. Analysis on chosen applications. (a) Applications classification and detection. (b) Location data invoking frequency.

2) slight ([1,10]); 3) normal ([11,100]); 4) frequent ([101, 1000]); and 5) greed (more than 1000), which illustrates in Fig. 2(b).

To draw the conclusion, closer inspection of the detection results indicates that the majority of such universal applications require location data. Even worse, some applications show their fanaticism on users' privacy by frequently tracking devices in both foreground and background services. From our detecting statistics, it is apparent that applications' are greedy for location data, resulting in users suffering from the risk of privacy leakage.

III. MODELS AND BASIC IDEAS

As mentioned, it is proved that applications are greedy and full of curiosity about location data, thus they are deemed as adversaries. In order to prevent adversaries from reaching users' locations continuously, DLP is proposed as users' location privacy-preserving instrument, which is designed to cover the real data with deceptive dummy techniques. In this section, we formalize the system model of location preservation at first, then introduce the adversaries which represent location data collecting applications, finally provide the basic idea and motivation of our solution, DLP.

A. System Model

In the system model, three entities are taken into consideration: 1) smart devices users; 2) lbs applications and providers; and 3) location privacy-preserving services, we illustrate them in Fig. 3. Assuming that each user is equipped with smart devices, which present location data to applications and providers for LBS. Meanwhile, when users decide to protect location data from malicious applications, location privacy-preserving services will be introduced. Hence, users' precise coordinates can be protected.

1) Smart Devices and Users: Nowadays, users are surrounded with many and varied forms of smart devices, for

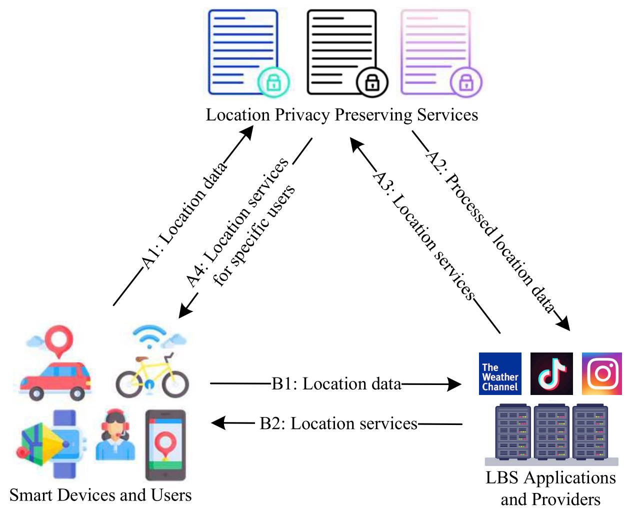

Location Privacy Preserving Services
A1: Location data
A4: Location services for specific users
A3: Location services
A2: Processed location data
B1: Location data
B2: Location services
The Weather Channel
Smart Devices and Users
LBS Applications and Providers

Fig. 3. System model under consideration.

instance, smartphones, wearable gears, intelligent cars, etc. Generally, smart devices are applied to provide specific data for users to request sorts of convenient services from services providers, including devices-installed-applications or cloud services providers.

2) LBS Applications and Providers: To fulfill requirements under sophisticated and diverse situations, profuse LBS applications and providers have emerged. On the one hand, most of the applications can solely provide services with locations, e.g., navigation applications can locate users or plan routes when devices are offline. On the other hand, sorts of applications also require corporation and coordination with LBS providers. For instance, weather applications send users' locations to weather providers for retrieving weather forecast. Indeed, although locations are indispensably required for LBS, some malicious applications and providers may go to great lengths to force users to submit locations at anytime and in anywhere. As a result, users' location data can be easily collected and their privacy will be exposed.

3) Location Privacy-Preserving Services: When users need to protect their location data from applications and providers, location privacy-preserving services will be deployed. In general, location privacy-preserving services can be divided into *local* and *server* types. Specifically, the former ones deal protecting procedures via modules or applications installed on smart devices. In contrast, the latter ones require server-side supporting, i.e., they are TTPs relayed types. In a word, with whichever privacy-preserving services, location data will be disposed before being submitted. As a result, applications and providers will essentially grab processed data. Note that we will only focus on thwarting the threats from applications in this article, since it is applications that directly collect locations from smart devices, despite the fact that both of LBS applications and providers can be considered as adversaries.

B. Adversaries Model

1) Applications' Capabilities: Assuming that applications tend to gather users' real locations, which is sustained both in foreground and background services. To achieve this goal, applications will: 1) bind all services to location data; 2) request finer coordinates in all situations; and 3) locate devices when running in background. In other words, applications may refuse to provide services when location data requests are denied, or coarse coordinates are provided only, or background invoking failed. When mentioned conditions are all met and applications can reach location data as pleases, they will go even further to verify the reliability of gathered coordinates. Specifically, when: 1) data are not satisfied with basic LBS requirements, for instance, multiple locations are provided in the same LBS session by one user; 2) gathered coordinates are not spatially continuous, e.g., users are detected moving from Beijing to New York City in a second; and 3) applications find users providing unreasonable locations like top of the Mount Everest, applications may question users' honesty and suspend LBS.

2) Adversaries Definition: Applications are classified into three types according to users' needs.

1) Applications which users need their functional LBS on foreground services but refuse them to use LBS in background services, i.e., users allow such applications to precisely position devices on foreground services only.
2) Applications which assumed to be irrelevant or have low correlation with LBS, hence no real location data should be permitted to them.
3) Applications which are trusted and wished to locate devices at any moment, like SOS applications. Hence, the first ones can be defined as *semi-trusted adversaries*, the second ones are defined as *mistrusted adversaries*, and the last ones are trusted applications.

C. Basic Ideas

Based upon our system model and adversaries model, to inhibit adversaries from gathering users' locations, as well as meeting users' necessity, we establish DLP on the following principles.

1) Deceptive: As mentioned, adversaries would refuse to provide services when failed on invoking required locations. Moreover, they would also question users' honesty when gathered coordinates are suspicious. Therefore, a fundamental goal of DLP is misleading adversaries.

1) To fulfill adversaries' requirements on location data invoking, dummies are introduced to be presented to them. DLP should allow adversaries to access high-precision coordinates under any circumstances.
2) To reassure adversaries, it is obliged to prevent the occurrence of unreasonable coordinates and spatially discontinuous paths. Thus, DLP should feed only single dummies to particular adversaries in individual LBS sessions. Meanwhile, semantic information related and spatially continuous dummies and trajectories are supposed to be generated and deployed

2) Customizable: DLP should restrict adversaries from reaching users' locations with disparate granularity and customizable strategy. Specifically, certain applications are defined as different adversaries and deployed with respective privacy strategies.

1) Semi-trusted adversaries' are applications set to gather real location coordinates at foreground on users'

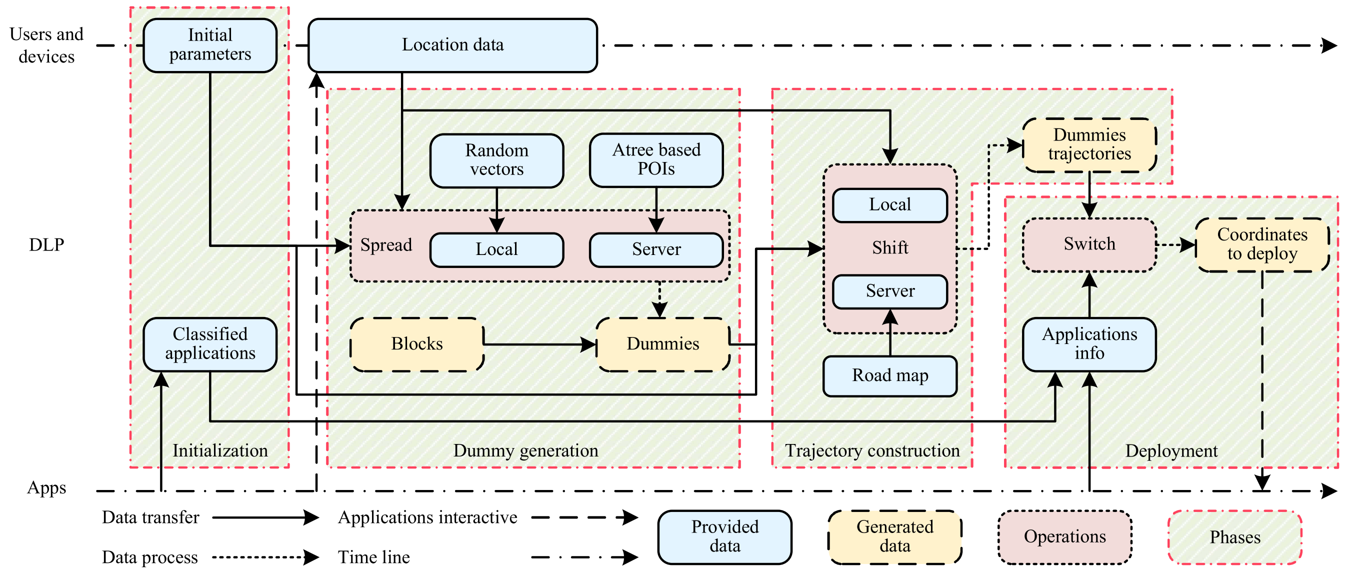


Fig. 4. DLP overview.

demand, but receive dummy locations while providing background services. Accordingly, semi-trusted adversaries' LBS will be guaranteed when running in foreground.

2) Mistrusted adversaries are only provided with dummies, it is intractable for them to reverse users' location data if dummies were generated with no limits. However, users may require mistrusted adversaries providing coarse LBS with low accuracy coordinates, hence dummies near real locations are preferred.

3) Achievable: DLP is designed to be an achievable standalone user-facing tool. In particular, the DLPA is needed to be constructed and tested on smart devices. Meanwhile, to avoid the participation of suspicious TTPs, DLP should be able to provide location privacy-preserving services independently. Finally, the energy consumption and communication cost of which are supposed to be maintained at a lower level.

IV. Our Proposed DLP

In this section, we propose and detail the designment of DLP, which mainly consists of the following phases, as Fig. 4 illustrated: 1) DLP initialization and dummies generating; 2) dummy trajectories construction; and 3) location data deployment. Correspondingly, DLP adopts three algorithms as core parts, *Spread*, *Shift*, and *Switch*. Specifically, *Spread* is implemented to generate dummy locations, *Shift* indicates how to modify dummy locations' coordinates for trajectories construction, and *Switch* provides such trajectories for both background and foreground adversaries.

A. DLP Initialization and Dummies Generating

When DLP activates in the initialization, a complete scan on installed applications will first be implemented, thus LBS applications are enumerated for users to deal with the adversaries classification. Then, as Table II presents, four initial parameters t, n, k, and α are defined by users. Finally, shown in Fig. 5, the

TABLE II
Notions of Initialization and *Spread*

Notion	Description
t	Parameter for trajectories generating, time *Shift* takes.
n	The number of generated dummies in each block.
k	The side length of each block.
α	The Recovery parameter.
B_{i_p}	The specific block that user currently stays.

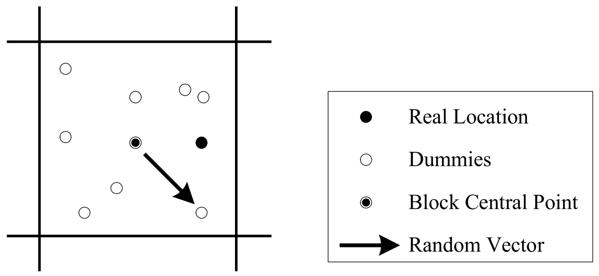


Fig. 5. Generate dummies via *Spread* in local type.

DLPA will segment map into blocks with k meters as sides, and restore such pregenerated blocks for upcoming phases.

When *Spread* generates dummies, it will first determine which block B_{i_p} covers users' real locations. Then users should designate the dummies generator in either local or server type. Finally, the corresponding generator will output dummies as follows.

1) When local type is selected, the DLPA will provide dummies with users' real locations and random vectors, illustrated in Fig. 5. Such procedures can be independently operated by DLPA, which means it can preserve location privacy without server-side collaboration.

2) Relatively, if server is assigned, it provides points of interests (POIs)-based dummies in the current block. POIs are consist of four components that

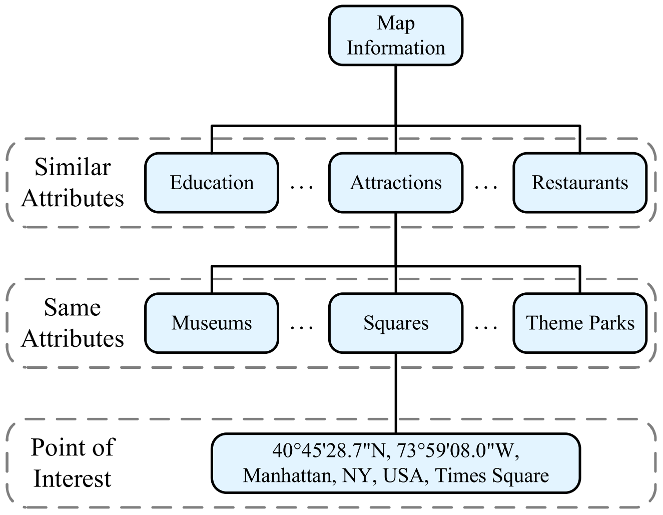


Fig. 6. POI restored in Atree.

are: 1) geographic coordinates; 2) semantic coordinates; 3) semantic names; and 4) classifications. For instance, 40°45′28.7″N 73°59′08.0″W/ Manhattan, NY, USA/ Times Square/Attractions, Square. In this work, we use POIs to classify and restore semantic coordinates, together with an attribute-based hierarchical tree (ATree), as Fig. 6 indicates. Hence, users' geographic coordinates will first be converted into semantic-information-included POIs, and other POIs with the same or similar attributes are also available. Then the server generates a set, in which *n* coordinates in current block are included. During the generating process, POIs which contain same or similar attributes with users' POIs are mostly preferred. However, it is possible that quantities of required POIs are not desirable, or even worse, the total number of POIs in the blocks could not meet the requirement. To deal with it, stochastic coordinates will also be generated till coordinates in the collection meet the requirement as *n*. Finally, a random choice is applied to the coordinates collection, with which we are able to get a dummy POI in range of the current block, and masked LBS can be provided with such a dummy.

B. Dummies Trajectories Construction

With the given outset and destination, *Shift* is designed to generate sets of dummy locations for constructing virtual trajectories, which are devoted to imitate patterns of human beings movement and mislead adversary that users are moving from one position to another. To describe *Shift*, we give a description of notations in Table III, as well as more details below.

As mentioned, dummies generated by *Spread* can be divided into local and server types. Correspondingly, *Shift* also constructs two sorts of trajectories.

Algorithm 1 shows how DLPA constructs trajectories in local mode.

1) As the first step, according to a outset PO_0 and destination PD_0, the algorithm will calculate the distance between the first outset PO_0 and destination PD_0 as the initial d_{0_0}.

TABLE III
NOTIONS OF *Shift*

Notion	Description
Tra_i	The dummy trajectory under constructed.
PO_i	Outset positions in Tra_i, change with users' real location or service condition.
PD_i	Destination positions in Tra_i, change with users' real location or service condition.
PN_{i_p}	Position of the *p*th dummy in Tra_i, central of circle O_{i_p}.
O_{i_p}	The *p*th circle in Tra_i, used for generating $PN_{i_{p+1}}$.
r_{i_p}	Radius of circle O_{i_p}.
d_{i_p}	Distance from current PN_{i_p} to PD_p.
t	Time *Shift* takes.

Algorithm 1 *Shift*

```
Input:
  $p = 0$, time $t$,
  outset dummy position $PO_i$,
  first dummy position $PN_{i_0} = PO_i$,
  destination dummy position $PD_i$.
Output:
  Current dummy trajectories $Tra_i$ ,constructed by $PN_{i_p}$
  while $p \neq t - 1$ do
    count distance $d_{i_p}$ between $PD_i$ and $PN_{i_p}$;
    count radius $r_{i_p} = d_{i_p}/(t - p)$
    set up circle $O_{i_p}$ with center $PN_{i_p}$ and $r_{i_p}$
    pick up random $PN_{i_{p+1}}$ in $O_{i_p}$
    $p + 1$
  end while
```

2) Generally, the algorithm is designed to imitate a virtual human that move on a virtual trajectory in *t* seconds, according to users' assignments in DLP initialization. Hence, the algorithm will generate *t* dummy locations to construct such a trajectory. For generating each dummy $PN_{i_{p+1}}$, we first set up circle O_{i_p} using $d_{i_p}/(t - p)$ as radius and PN_{i_p} as center. In such circle O_{i_p}, a 120° circle sector will be set with median $\overrightarrow{PN_{i_p}PD_i}$. Finally, we pick up a random point as $PN_{i_{p+1}}$ in the sector. The process iterative until $p = t$ and PN_{i_p} has the same coordinate as PD_i. *Shift* is not only effective for immobile users, but also performs competently on facing moving users by updating both PO_i and PD_i, as the algorithm is illustrated in Fig. 7.
3) As the sets of dummy locations have been generated, we return them to location-related methods by time. Also shown in Fig. 7, when DLP detects adversaries invoking locations at time t_1, t_2, and t_3, it will report dummies PN_{i_1}, PN_{i_2}, and PN_{i_3}, respectively.

By contrast, when server type is selected, trajectories will be built up based on the street map information. DLP server will first find a road between outset PO_0 and destination PD_0 using the street map. Then the server computes the distance d_{0_0} of this road, and *t*-1 points in this road will be equidistantly determined. Hence, a virtual trajectory is created under real-world

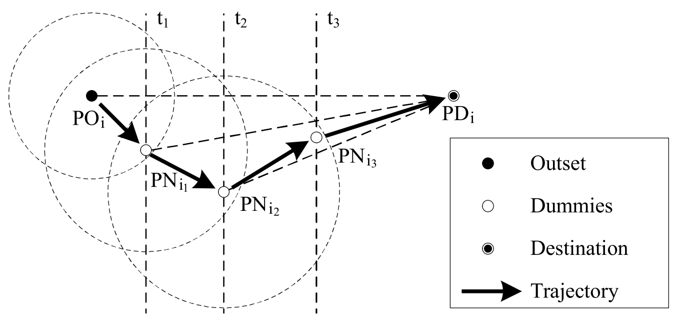


Fig. 7. Dummy trajectory constructed by *Shift*.

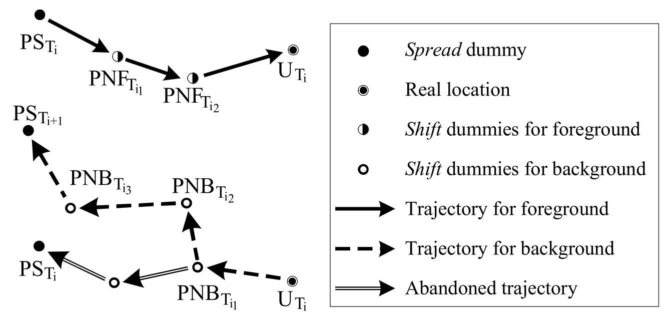

PSTi
PNFTi1
PNFTi2
UTi
PSTi+1
PNBTi3
PNBTi2
PSTi
PNBTi1
UTi
Spread dummy
Real location
Shift dummies for foreground
Shift dummies for background
Trajectory for foreground
Trajectory for background
Abandoned trajectory

Fig. 8. Dummy trajectories deployed by *Switch*.

constraints, which presents a simulation of a human being shifting from PO_0 to PD_0 in t seconds.

C. Dummies Deployment

When adversaries are detected invoking location methods, DLP will replace real location data by dummies, we define such operations as deployment, shown in Algorithm 2. First, DLP maintains a particular background service which calls on and restores users' physical location data with a fixed time. According to users' coordinates, DLP can generate sets of dummy locations by *Spread*, then produces masqueraded paths through *Shift*, as shown in Fig. 8. Finally, *Switch* deploy dummies trajectories to meet the needs under various circumstances, which can be classified into the Generation, the Recovery, and the Transformation. For better describing the algorithm, notations are listed in Table IV.

1) Generation: In order to answer the call from mistrusted adversaries or when semi-trusted adversaries moving from foreground to background services, the Generation implements trajectories' construction and deployment in sequence. Such operation imitates that users' are moving from their physical locations to the *Spread*-generated dummies.

2) Recovery: In the situation of Recovery, current dummy locations are set as the outset, while user's physical location coordinates PD_n representing the destination. Trajectories constructed with such processes should be expressed when the semi-trusted adversary swapping from background services to foreground. In such process, an additional parameter $\alpha \in [0, 1]$ should be determined by the user, which make it possible to adjust the final coordinates represented to semi-trust adversaries. Specifically, adversaries can obtain users' real locations PD_ns when $\alpha = 1$, on the contrary, the final coordinates will depart from PD_ns while α tending to 0. Meanwhile, different ts are applied to various adversaries, for instance,

TABLE IV
NOTIONS OF *Switch*

Notion	Description
T_i	The time list in *Switch*.
T_S	The time when user switch Foreground and background applications.
X	Application X.
XF, XB	Application X in foreground / background service.
Y	Application Y.
YF, YB	Application Y in foreground / background service.
PF_{T_i}	Position DLP send to foreground applications.
PB_{T_i}	Position DLP send to background applications.
PS_{T_i}	Generated new dummy position with *Spread*.
U_{T_i}	User's real location.
t	Time *Shift* takes.
PO_{T_i}	Outset positions in algorithm *Shift*.
PD_{T_i}	Destination positions in *Shift*.
$PNF_{T_{ip}}$	Position of the pth dummy for foreground application.
$PNB_{T_{ip}}$	Position of the pth dummy for background applications.

navigation applications need precise coordinates immediately when services come to foreground, but social applications need no urgent coordinates services, thus users should set distinct ts for corresponding adversaries.

3) Transformation: As for the Transformation, we assume that users can switch adversaries' conditions whenever necessary, and their physical positions are under an unsteady state.

While applications status changing, we take the following situations into consideration.

1) When semi-trusted adversaries are swapped to the background, or when a background-running application is defined as a mistrusted adversary, *Switch* should deploy trajectories constructed in the Generation phase to them.
2) Correspondingly, a Recovery trajectory will be deployed to the new foreground semi-trusted adversary, or when a mistrusted adversary regains trust from the user.

As for users hold in moving state, *Switch* will preserve dynamic starting points PO_is and destinations PD_is and continuously update them for both Generation and Recovery trajectories' construction. For instance, DLP first constructs a trajectory at time T_i for a new background semi-trusted adversary, and since then it simulates user's movement from real location U_{T_i} to PS_{T_i} as the Generation, in other words, $\overrightarrow{PO_{T_i}PD_{T_i}} = \overrightarrow{U_{T_i}PS_{T_i}}$. Then, at time T_{i+x}, if DLP detects that the user has moved to $U_{T_{i+x}}$, the algorithm will produce a trajectory from current dummy $PNB_{T_{ix}}$ to newly generated dummy $PS_{T_{i+x}}$. Finally, the trajectory $\overrightarrow{PB_{T_i}PB_{T_{i+x+t}}}$ can be expressed by $\overrightarrow{U_{T_i}PNB_{T_{ix}}} + \overrightarrow{PNB_{T_{ix}}PS_{T_{i+x}}}$, as shown in Fig. 8. Otherwise, the generated path will only comprise $\overrightarrow{U_{T_i}PS_{T_i}}$, if the user stays still. Similarly, users' real location U_{T_i}s will be taken as destinations for Recovery trajectories, and dummies should be generated to gradually approaching the final U_{T_i}.

D. Security Analysis

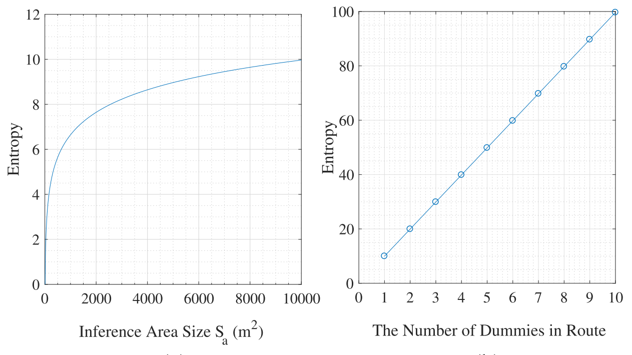
Following the Adversaries Model, applications may continuously invoke location data and verify the reliability of gathered coordinates. To prevent applications from

Algorithm 2 *Switch*

```
Input:
    $X, Y, T_i, T_S, t, PS_{T_i}$, user's real location list $U_T$.
Output:
    Position list DLP send to background applications, $PB_T$.
    Position list DLP send to foreground applications, $PF_T$.
1:  Before $T_S : X = XF,\ Y = YB,\ PF_{T_i} = U_{T_i}$
2:  if $U_{T_i} \neq U_{T_{i-1}}$ then
3:      $PB_{T_i} = PS_{T_i}$;
4:  end if
5:  if $U_{T_i} = U_{T_{i-1}}$ then
6:      $PB_{T_i} = PB_{T_{i-1}}$;
7:  end if
8:  When $T_S : XF \rightarrow XB,\ YB \rightarrow YF$
9:      $Shift_{XF\rightarrow XB}\ (p, t, PO_{T_i} = U_{T_i}, PD_{T_i} = PB_{T_i})$
10:     $Shift_{YB\rightarrow YF}\ (p, t, PO_{T_i} = PB_{T_i}, PD_{T_i} = U_{T_i})$
11: After $T_S : X = XB,\ Y = YF$
12: if $U_{T_i} \neq U_{T_{i-1}}$ then
13:     Break $Shift_{XF\rightarrow XB}, Shift_{YB\rightarrow YF}$
14:     while $PB_{T_i} \neq PS_{T_i}$ do
15:         $Shift\ (p, t, PO_{T_i} = PNB_{T_{ip}}, PD_{T_i} = PS_{T_i})$
16:         $PB_{T_i} = PNB_{T_{ip}}$
17:     end while
18:     while $PNF_{T_{ip}} \neq U_{T_i}$ do
19:         $Shift\ (p, t, PO_{T_i} = PNF_{T_{ip}}, PD_{T_i} = U_{T_i})$
20:         $PF_{T_i} = PNF_{T_{ip}}$
21:     end while
22: end if
23: if $U_{T_i} = U_{T_{i-1}}$ then
24:     while $PNB_{T_{ip}} \neq PS_{T_i}$ do
25:         $PB_{T_i} = PNB_{T_{ip}}$
26:     end while
27:     while $PN_{T_{ip}} \neq U_{T_i}$ do
28:         $PF_{T_i} = PNF_{T_{ip}}$
29:     end while
30: end if
```

constantly tracking users, DLP first feeds modified coordinates to both mistrusted and background running semi-trusted adversaries. Therefore, DLP can fully satisfy adversaries' invocation requirements and prevent adversaries from accessing real locations, i.e., tracking-based attack is resisted in our proposed scheme. In addition, DLP generates POI-based high-precision dummies and constructs dummy trajectories to mislead adversaries that users are moving as normal human beings. As a result, users will not be questioned on honesty. Moreover, DLP is able to preserve location privacy without server-side collaboration, hence our mechanism is also jamming [22] and colluding attacks resistance.

Even though DLP applies privacy protection against varied adversaries and attacks, skeptical applications may suppose that users adopt location privacy-preserving measures by default, in other words, they will still try to expose users' real locations through the collected data. Hence, it is necessary for us to focus on defending inference attacks. Academically, entropy is widely prevalent to measure the randomness degree of privacy-preserving mechanisms [23], [24], i.e., processed



(a) (b)

Fig. 9. Inference entropy raising. (a) Inference area expanding. (b) Dummies increasing.

data gets harder to be revealed when protector systems achieve higher entropy. In this work, we also adopt entropy to represent the uncertainty of DLP-generated dummies and the opportunity of adversaries inferring users' physical coordinates. For adversaries, they grab and analyze users location data in different degrees, thus various inference attacks may be processed. Essentially, we classify inference attacks into two types.

Normally, adversaries can first assume that the user's real location is in an approximate area S_a, for instance, a 1000-m^2 area. Then the adversary will try to infer where the user actually is in the presumptive S_a, along with accessing a range S_h as the definition of the precise location. Finally, if the adversary gets the users' real location with an area S_h that is small enough to expose users' precise coordinate, for instance, 10 m^2, we define it as a successful inference attack. In such procedures, it is a rational argument for us to postulate that adversaries have to establish the certain area S_a via grabbed historical location data. In our scheme, dummies are provided in blocks, hence adversaries are able to depict the shape and central point of such blocks when locations are tremendously accumulated. As a result, the particular block will be considered as the inferring area S_a. Therefore, to compute the entropy, the possibility for adversaries inferring S_h can be denoted by $p_i = S_h/S_a$, which holds the sum of all probabilities equal to 1. Aiming to achieve maximum entropy

$$H_{\max} = -\sum_{i=1}^{k} p_i \times \log_2 p_i$$

the random coordinates in our scheme are equiprobably chosen, hence $H_{\max}$ positively correlates with S_a as illustrated in Fig. 9(a). In other words, the entropy H rises when block size expands, because the H reaches superior value when S_h/S_a gets lower. That is, when the initial parameter k increases and inferring area S_a correspondingly expands, adversaries will find it more arduous to perform precise inference attack. As for users' route inferring, when a user shifts in various blocks, adversaries are not able to reveal the actual trajectories with a small range of S_a, which excludes real coordinates. On the contrary, when S_a contains both real and dummy locations, the maximum entropy $H_{\max}$ are computed with $P_i = (S_h/S_a)^n$, where n denotes the number of dummy locations in such routes, thus the entropy of users' routes can be denoted with

$$H_{\text{route}} = -\sum_{i=1}^{k} (S_h/S_a)^n \times \log_2(S_h/S_a)^n$$

as Fig. 9(b) depicts.

However, some adversaries may also assume that users are actually providing real locations together with dummies, hence the initial parameter n, which represents quantities of dummies in each blocks, will play another important role in computing entropy. Under such a circumstance, adversaries will finally collect n dummies in a specific block, in other words, they always have a probability $p_i = 1/n$ to reveal users' real locations in single attacks. Therefore, when facing such inferences, the entropy will be

$$H_{\text{route}} = -\sum_{i=1}^{k} 1/n \times \log_2 1/n.$$

DLP can apply better protection with larger initial parameter ns. Moreover, adversaries may be also equipped with semantic map information, i.e., they can convert grabbed geographic locations to POIs, which significantly contributes on raising the accuracy of inference attacks. Specifically, DLP will randomly generate dummies when the quantities of POIs in blocks do not meet the requirement of n. As a result, some of such dummies may be unreasonable and irrational, which may be get rid of by adversaries with high probability. Assuming that adversaries have removed m dummies in a specific block, their successful chance on guessing the real location will increase to $p_i = 1/(n-m)$, as a result, the entropy will correspondingly decrease, as follows:

$$H_{\text{route}} = -\sum_{i=1}^{k} 1/(n-m) \times \log_2 1/(n-m).$$

V. Performance Evaluation

In this section, we first describe the implementation detail of proposed DLP. Then, we focus on the performance of our scheme in terms of availability. Finally, to measure the scheme efficiency, two metrics are considered, including computation cost and communication overhead.

A. DLP Implementation

In order to measure the performance of the scheme, we implement the DLPA for Android-powered devices, a server-side dummies provider, and an additional test carrier (TC) application.

1) Implementation Environment: In the construction of DLPA and the server-side dummies provider, we implement the application on a smartphone with 2.3-GHz 4-core processor and 2-GB RAM, which carries Android Oreo OS. In addition, a PC with 2.0-GHz 6-core processor and 16-GB RAM is chosen to set up our server and database. As for logging and debugging, we bring a laptop with a 1.6-GHz CPU and 8-GB RAM into service, and the Android Studio is chosen as our main IDE.

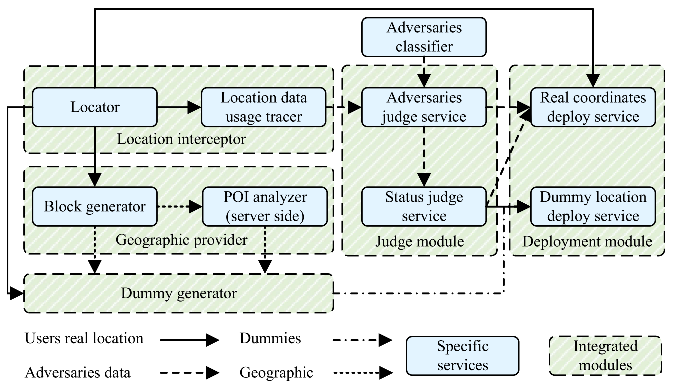


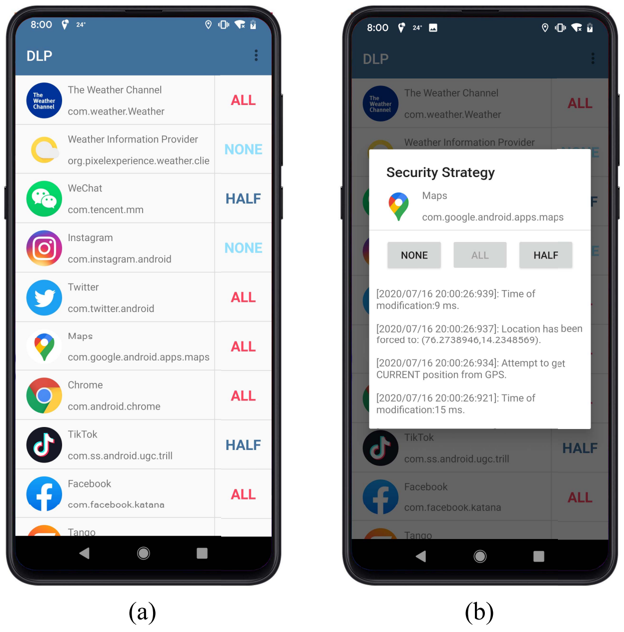
Fig. 10. DLPA workflow.

2) DLP Design: The DLPA is primarily in charge of location data collecting and location-related methods interception. Besides, the application will also generate dummies using grabbed location data and separated blocks. And the most important is, it should deploy dummies to adversaries, including mistrusted and semi-trusted ones. Correspondingly, the server-side dummies provider will continuously gather users' coordinates via the DLPA. Then the provider can produce dummies and trajectories using grabbed coordinates, road map and POI information.

3) DLPA Architecture and Workflow: As Fig. 10 indicates, the DLPA is mainly constructed with the following five modules.

1) In Location interceptor, Locator retrieves users' locations in a constant frequency, hence other services and modules can conveniently reach users' locations. When the Location interceptor detects location-related methods invoking, it should immediately suspends such methods requests and trace the invoker applications.
2) With the detail information about the location-related methods invocator applications, the Judge module can tell the adversary classification of such applications. If the applications are accused as semi-trusted adversaries, their running status should also be reported.
3) Basically, the Geographic provider is in charge of generating blocks and distinguish the specific block that users currently in. Meanwhile, if the server type is selected, the server-side dummies provider will prepare and restore available POIs in Atree.
4) Based on users' real-time locations, separated blocks, and geographic data in varied forms (including POIs and street map information, only necessary when TTPs are required), Dummy generator can produce sorts of dummies and dummy trajectories using *Spread* and *Shift*, mentioned in Section IV.
5) When hijacked location-related methods are invoked by mistrusted or background-running semi-trusted adversaries, the Deployment module should modify the methods and inject dummies via the *Switch* algorithm.

Beyond mentioned description, there are still few issues that merit attention occurred in the implementation of DLPA and server-side dummies provider. First of all, to make sure that the DLPA can provide dummies and trajectories immediately,


(a)

8:00 24°

DLP

The Weather Channel com.weather.Weather	ALL
Weather Information Provider org.pixelexperience.weather.clie	NONE
WeChat com.tencent.mm	HALF
Instagram com.instagram.android	NONE
Twitter com.twitter.android	ALL
Maps com.google.android.apps.maps	ALL
Chrome com.android.chrome	ALL
TikTok com.ss.android.ugc.trill	HALF
Facebook com.facebook.katana	ALL
Tango	

(b)

Security Strategy

Maps
com.google.android.apps.maps

NONE ALL HALF

[2020/07/16 20:00:26:939]: Time of modification:9 ms.

[2020/07/16 20:00:26:937]: Location has been forced to: (76.2738946,14.2348569).

[2020/07/16 20:00:26:934]: Attempt to get CURRENT position from GPS.

[2020/07/16 20:00:26:921]: Time of modification:15 ms.

Fig. 11. Implementation of the DLPA. (a) Adversaries list. (b) Detailed reports.

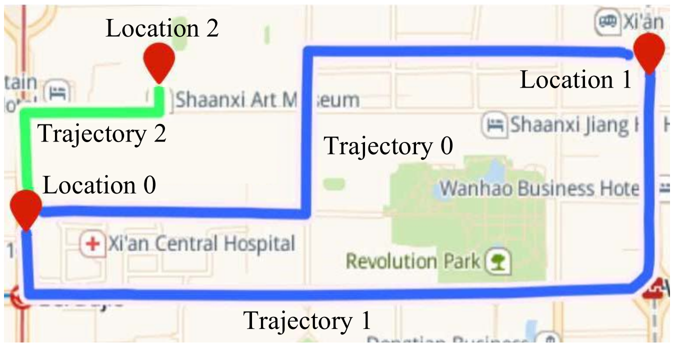


Fig. 12. Test on scheme availability.

sorts of services and modules are designed to run permanently in the background. For instance, the Locator service, Geographic provider, Dummy generator, etc. Besides, in the process of building up the DLPA, the *PassiveProvider* should be banned in order to make our scheme colluding attack resistant. Finally, note that DLPA is designed to construct dummies and trajectories without POIs or street map information, since smart devices can hardly satisfy the requirements for supporting the DLPA working independently in terms of storage and computing power.

4) DLPA and TC Implementation:

1) Illustrated in Fig. 11(a), DLPA detects location data required applications and enumerates to users at first. Meanwhile, adversaries' states are also reported to users in this activity. When users click on listed adversaries, the location data calling records of corresponding application will be detailed, as Fig. 11(b) shows. In addition, users can also define the application in various adversary types.
2) TC is mainly integrated with the location recording function, i.e., it locates devices and reports real-time locations reached by applications in both foreground and background. Therefore, relying on TC, we can verify whether DLPA is successed on hijacking and modifying location-related methods. Besides, TC is adjustable in terms of status and frequency, so users can define TC as different types of adversaries and set TC to record locations once per 3 or 10 s.

B. DLP Effectiveness

To validate the effectiveness of our proposed scheme, we test DLP in terms of dummies generating and deployment. Specifically: 1) we first install DLPA and TC on the same Android-powered device; 2) then TC is defined as both mistrusted and semi-trusted adversaries in DLPA, so we can record locations reached by TC; and 3) by comparing users' real locations and the output of TC, we demonstrate the availability of our scheme under all the situations. In addition, the experiment is conducted under real-world constraints in Xi'an, China, and involved POIs can be accessed in public databases [25]. The experiment is detailed in the following.

1) As shown in Fig. 12, we set an immobile device at location 0. The parameters t, n, k, and α for DLPA are defined as 600, 1000, 2000, and 1, respectively.
2) When we turn the trusted application TC to a mistrusted adversary, the location reported by TC shows conspicuous change. According to the locations provided by TC, we draw a trajectory from Location 0 to Location 1, that is Trajectory 0. Trajectory 0 indicates that DLPA successfully generates dummys, hijacks the mistrusted adversary TC, and modifies location data invoking procedures.
3) Redefining TC as a semi-trusted adversary, Trajectory 1 illustrates that DLP can recognize the foreground running semi-trusted adversary and correspondingly provide Recovery trajectories.
4) When execution of Recovery completes and TC is able to report Location 0 again, we swapping the semi-trusted adversary to background, leading to DLP deploys another *Spread* trajectory, as Trajectory 2 indicates.
5) Finally, to verify the universality of DLP, we make the test on applications which are detected location invoking from Section II. The result indicates that all of the 41 apps are hijacked when location requests detected and dummies are deployed in perspective, detailed in the Appendix.

To draw the conclusion, DLP accesses deceptive against skeptical adversaries, since high precision and semantic information related dummies are generated and deployed to reassure adversaries. Meanwhile, DLP also attains disparate granularity and customizable strategy on restricting adversaries from reaching users' locations. For adversaries in varied types and running status, DLP can provide dummies and trajectories with respective privacy strategies. Hence, it is an rational expectation that DLP works effectively to protect mobile users' location privacy and achieves our design goal in Section III-C.

C. Energy Consumption

Owing to the requirements of detecting and hijacking location-based methods invoking from adversaries, the DLPA

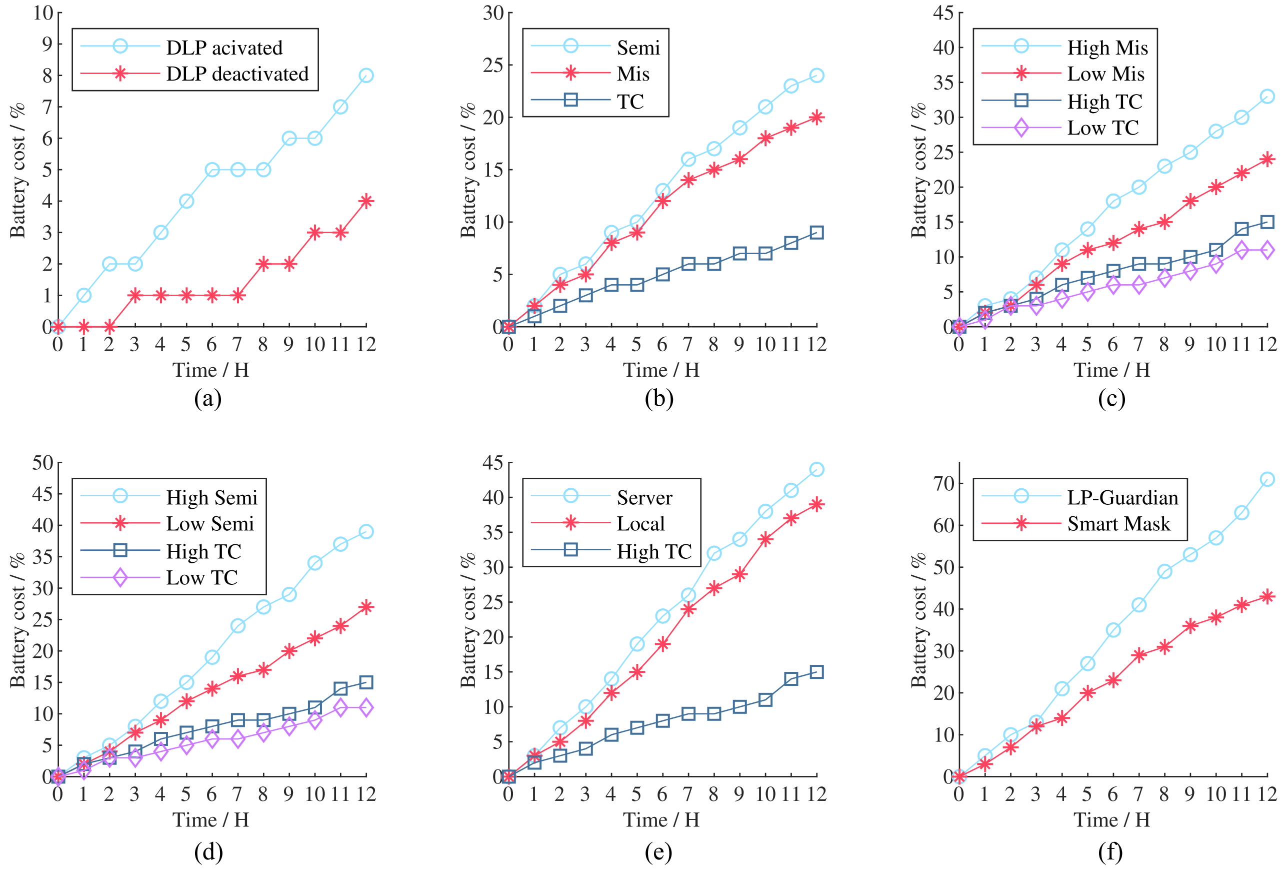


Fig. 13. DLP energy consumption. (a) Cost when the DLPA activated, compared with application deactivated. (b) Cost of the DLPA while hijacking mistrusted and semi-trusted adversary. (c) Energy consumption increase due to mistrusted adversaries invoking frequencies rise. (d) Energy consumption increase due to semi-trusted adversaries invoking frequencies rise. (e) Energy consumption increase due to the DLPA requests dummies from the server. (f) Scheme comparison.

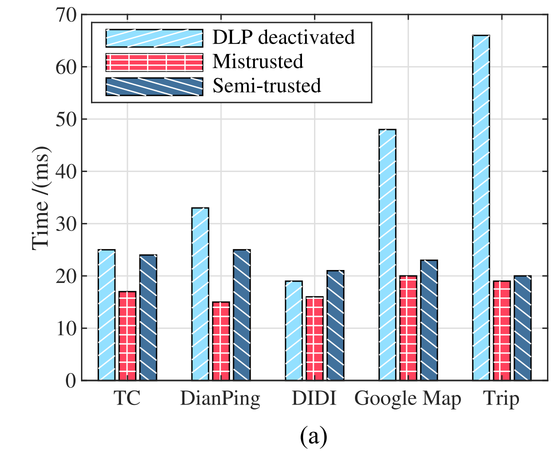
maintains permanently background services, thus its energy cost should be taken into consideration. We set up an experiment on measuring such cost by shutting down unrelated applications but keeping adversaries and DLPA active. Recording the battery usage per hour, results of such experiment can be drawn in Fig. 13.

1) Consumption of DLPA Activation: First, when the application runs in the background with no adversaries invoking location data, the application only locates the device in a constant frequency and generates dummies without deployment, hence the energy cost of the application will be maintained at a low level in such situation. Fig. 13(a) indicates that in 12 h the device we used for evaluation costs 4% of battery power in stand-by mode when the application is at a standstill. By contrast, 4% more of the battery power will be consumed by DLP in the same duration, which is negligible.

2) Consumption of Interception: By setting up TC applications which call on location data continuously, we conduct our next experiment on the DLPA's energy cost while hooking: 1) one mistrusted adversary and 2) one semi-trusted adversary. Compared with deploying dummies and trajectories to mistrusted adversaries, the DLPA needs to perform additional processes of judging running states while hijacking semi-trusted adversaries, as a result the energy cost of the DLPA increases successively. Illustrated in Fig. 13(b), the battery usage of our device comes to 9% after 12 h as a result of the TC calling location data once per 10 s in background service. Meanwhile, if we define the TC as a mistrusted adversary in the DLPA and run the application together with TC, the battery usage will rise to 20%. Furthermore, when the TC was treated as a semi-trusted adversary, the usage will continuously grow up to 24%. Correspondingly, it is obvious that the DLPA cost 11% and 15% of the power while hijacking one mistrusted adversary and one semi-trusted adversary, respectively.

3) Consumption of Interception With Higher Frequencies: Then we describe how energy costs expand with adversaries' location invoking increase. Specifically, we set two adversaries which request location data once per 5 and 3 s, resulting in the DLPA intercepting location methods and deploying dummies more frequently. As Fig. 13(c) indicates, when the application hijacks mistrusted adversaries in a lower frequency, it cost 13% of the battery in 12 h, however, the cost will move up to 15% due to adversaries requesting location data more often. In addition, Fig. 13(d) illustrates that 16% and 21% of power are used while hijacking two kinds of semi-trusted adversaries.

4) Consumption of Interception With Server-Side Communication: Since our scheme consists of the DLPA and server-side dummies provider, the communication between



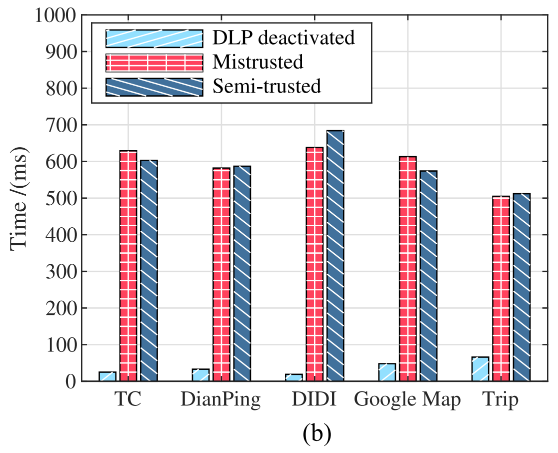


Fig. 14. DLP communication cost. (a) Internal communication cost. (b) Client-server communication cost.

two components will also cause power usage increment. We set a TC with the high location invoking frequency (once per 3 s), and two types of dummies generating methods are deployed successively. When the DLPA request dummies and trajectories from the server, Fig. 13(e) demonstrates that the additional battery cost in 12 h will come to 5%, compared with generating by the application it own, due to the dummies generating services of the DLPA will not stop even if the server is unavailable.

To draw the conclusion, when DLP only works on mistrusted adversaries, the energy cost is maintaining at a stable level, because there is only one judging service that participates in the working flow on background. Even though the engaging of semi-trusted adversaries will slightly raise the battery cost, such extra consumption is totally tolerable. Moreover, although the quantity of background services remains unchanged, when adversaries invoke location data more frequently or when the application deploy dummies from the server instead of generating by itself, the higher cost will be taken. It is worth noting that the fluctuation and modification of our initial parameters cause negligible influences on the total energy cost. In addition, we list the energy consumption of LP-Guardian [26] and Smart Mask [27] in Fig. 13(f) as comparison.

D. Communication Cost

The representation of communication cost is mainly expressed by query time increasing. As the procedures of adversaries' gathering devices' coordinates are hijacked by the DLPA, such steps implement additional communication cost. When the application detects adversaries' calling location-related methods, intercepting processes will be activated. If such invokers have been identified as mistrusted adversaries or background-running semi-trusted adversaries, DLP would request dummies and trajectories from the server-side dummies provider, if necessary. Otherwise, the application should deploy self-generated dummies. Hence, the experiment on communication cost is divided into inspecting of internal delay and client-server communication delay.

Specifically, to measure the communication cost in such procedure, we select several applications from Section II as experiment objects to compare query response latency when the DLPA is activated and deactivated. Meanwhile, we also set a TC as reference by using DLP to hijack its coordinates' calling method. For TC is only equipped with locating function and has no communication with server, such comparison is able to express the communication cost of hooking processes in a more intuitional way. Furthermore, the DLPA is set to deploy dummies generated at local and received from the server, as Fig. 14 illustrates.

As mistrusted adversaries have to invoke devices location before requesting LBSs. However, the communication cost of such invoking process can be decreased due to the DLPA is able to return modified coordinates to mistrusted adversaries directly, i.e., mistrusted adversaries can get modified coordinates faster than calling on users' real location. However, it takes additional steps for the application to judge whether semi-trusted adversaries running on background or foreground, which is consisting of four parts: 1) listing all applications by running sequence; 2) getting last running application; 3) judging whether it is the semi-trusted adversaries; and 4) choosing results to return by last. Thus, the communication cost of DLP intercepting semi-trusted adversaries will be higher than getting users' real locations, just as Fig. 14(a) indicates. Besides, the communication cost increases rapidly when dummies are returned by the server-side dummies provider, which is demonstrated in Fig. 14(b).

VI. Related Work

There have been numbers of studies to preserve mobile users' location data, in this section, we review permissions model and some existing related mechanisms.

A. Permission Model on Android

Google company designed the permission model as the central point of the Android security architecture [28]. Android applications must request permission to access sensitive user data and certain system features. By default, no application has permissions to perform any operation that would adversely impact other applications, or the operating system, or the users. However, in the first several years of Android, permissions requests are only announced while applications installing, in other words, users are not able to refuse or restrict applications invoking sensitive data with permissions.

Taking such situation into consideration, a new permission model has been added in Android 6.0, where users

TABLE V
FUNCTIONALITY COMPARISON

	Zhang *et al*. [32]	Hara *et al*. [35]	Micinski *et al*. [37]	Fawaz *et al*. [39]	DLP
Deceptive	✗	✓	✓	✗	✓
LBS functional	✗	✗	✗	✓	✓
TTP independence	✗	✓	✗	✓	✓
Various attacks resistance	✗	✗	✗	✗	✓
High-efficiency	✓	✗	✗	✓	✓

can directly manage application permissions at runtime. This model gives users improved visibility and control over permissions, while streamlining the installation and auto-update processes for applications developers. More specifically, any application which requires permissions has to notify users, as well as applying for such permissions in a pop-up window. What's more, users are able to grant or revoke permissions individually for installed applications. To some extent, such a permission control model has contained applications abusing sensitive data. However, many unresolved challenges still exist for users' privacy data surveillance on the Android system. For instance, under the current permission control model, applications can use applied permission any time once users have allowed them, and some applications even provide no services without permissions granted.

B. Location Privacy-Preserving Schemes

To protect users' location privacy, numerous schemes have been proposed, in which the k-Anonymity and the dummy-based schemes are strongly correlated to our work.

In was in 2003 that Gruteser and Grunwald [29] first introduced k-Anonymity into location privacy. The idea of which is that any user should report a spatial and temporal cloaking area containing not only his/her location but also $k-1$ locations of other users, leading the specific user being indistinguishable from other $k-1$ users. The concept is so popular that several approaches have been proposed based upon it to provide location privacy. Among which CliqueCloak [30] was designed to maintain LBS functional while preserving users' privacy, HiSC [31] aimed at balancing the anonymizing workload among TTPs and smartphones, and CSKA [32] tried to reduce the risk of exposure of users' information in continuous LBS, etc. Unfortunately, most of k-Anonymity works rely on TTPs for cloaking regions generating and users anonymization, such trusted infrastructures are unrealistic under real-world constraints.

By means of generating several fake coordinates and sending them with real location to LBS servers, dummy-based methods can confuse adversaries without support of TTPs. In 2010, Suzuki *et al.* [33] proposed a method which generates dummies with consideration of the actual map information to keep consistency on dummies' movement. Then, Kato *et al.* [34] and Hara *et al.* [35] anonymized users' location with generating and arraying dummies considering real environment and geographical constraints. However, it has always been a challenge of generating dummies to be taken as real. Meanwhile, dummy-based schemes have been more sophisticated and complicated due to approaches are remaining constant evolution, resulting in the computation and communication cost raise rapidly.

C. System Level Solutions

To address mentioned challenges, more practical approaches have been proposed to fit within smart phones and platforms. MockDroid [36] can totally prevent applications from accessing location data, it can definitely protect users privacy, but LBS is fully disabled on the other hand. Micinski *et al.* [37] implemented CloakDroid, which modifies existing applications to use truncated location information for users privacy preserving, but the author ignored the location granularity requirements of LBS applications. The Caché [38] system also provides coarsened coordinates to LBS applications, however, developers are required to access location data in designated ways, i.e., the approach needs applications modification. Fawaz and Shin [26] implemented LP-Guardian and LP-Doctor [39] as per-app basis frameworks for location privacy preserving, they achieved protection for each app independently with maintaining applications functionality, unfortunately, the former one requires system-level modification, while the latter one is inapplicable to applications that require accurate or constant location access.

D. Functionality Comparison

Different with existing location privacy-preserving solutions, DLP demonstrates unique features as follows.

1) DLP can fully satisfy adversaries and avoid being questioned in terms of honesty.
2) DLP provides flexible location data usage control for functional LBS.
3) DLP attains stand-alone and effortless deployment over smart devices.
4) DLP protects users from various potential threats, including long-term tracking, jamming, colluding, and inference attacks.
5) DLP is high-efficient in terms of computation cost and communication overhead. In detail, the comparison of DLP and several representative mechanisms is presented in Table V, which indicates that DLP is more practical in the real environment.

VII. CONCLUSION AND FUTURE WORK

Our work began with detecting adversaries' process of invoking coordinates on Android-powered devices, as well as proving LBS providers are greedy to users' location data

TABLE VI
APPLICATIONS ANALYSIS RESULTS

Classification	**App name**	**Package name**	**Invoke location data in background services**	**Frequency (times/ 24 hours)**	**Hijacked and modified with DLP**
Transportation	DiDi	com.sdu.didi.psnger	Detected	46	Succeed
Transportation	Autonavi Maps	com.autonavi.minimap	Detected	263	Succeed
Transportation	Dianping	com.dianping.v1	Detected	372	Succeed
Transportation	Ctrip	ctrip.android.view	Detected	567	Succeed
Transportation	Baidu Map	com.baidu.BaiduMap	Detected	764	Succeed
Financial	JD Central	com.jingdong.app.mall	Undetected	Undetected	Undetected
Financial	Bank of China	com.chinamworld.bocmbci	Undetected	Undetected	Undetected
Financial	Taobao	com.taobao.taobao	Detected	42	Succeed
Financial	Pinduoduo	com.pdd	Detected	129	Succeed
Financial	Vipshop	com.achievo.vipshop	Detected	407	Succeed
Financial	Alipay	com.eg.android.AlipayGphone	Detected	1128	Succeed
News	iReader	com.chaozh.iReader	Undetected	Undetected	Undetected
News	Baidu	com.baidu.searchbox	Detected	3	Succeed
News	Moji Weather	com.moji.mjweather	Detected	32	Succeed
News	iFeng News	com.ifeng.news	Detected	66	Succeed
News	QuKan	com.jifen.qukan	Detected	137	Succeed
News	Tencent News	com.tencent.new	Detected	606	Succeed
News	TouTiao	com.ss.android.article.news	Detected	1478	Succeed
Media	Kugou Music	com.kugou.android	Undetected	Undetected	Undetected
Media	iFeng Video	com.ifeng.video	Undetected	Undetected	Undetected
Media	iQIYI Video	com.qiyi.video	Detected	4	Succeed
Media	QQLive	com.tencent.qqlive	Detected	26	Succeed
Media	Youku	com.youku.phone	Detected	38	Succeed
Media	Kuwo Player	cn.kuwo.player	Detected	76	Succeed
Media	QQMusic	com.tencent.qqmusic	Detected	185	Succeed
Media	YY	com.duowan.mobile	Detected	277	Succeed
Social	Qzone	com.qzone	Detected	2	Succeed
Social	Momo	com.immomo.momo	Detected	4	Succeed
Social	Weibo	com.sina.weibo	Detected	10	Succeed
Social	Wechat	com.tencent.mm	Detected	12	Succeed
Social	WeSing	com.tencent.karaoke	Detected	13	Succeed
Social	QQ	com.tencent.mobileqq	Detected	51	Succeed
Social	BeautyCam	com.meitu.meiyancamera	Detected	106	Succeed
Social	YY Live	com.duowan.kiwi	Detected	227	Succeed
Social	Kwai	com.smile.gifmaker	Detected	257	Succeed
Utility Tools	Sogou Pinyin	com.sohu.inputmethod.sogou	Undetected	Undetected	Undetected
Utility Tools	iFlytek Voice Input	com.iflytek.inputmethod	Undetected	Undetected	Undetected
Utility Tools	Tencent QQPim Secure	com.tencent.qqpimsecure	Undetected	Undetected	Undetected
Utility Tools	WPS Office	cn.wps.moffice	Undetected	Undetected	Undetected
Utility Tools	QQmail	com.tencent.androidqqmail	Detected	1	Succeed
Utility Tools	Meitu	com.mt.mtxx.mtxx	Detected	2	Succeed
Utility Tools	Meituan	com.sankuai.meituan	Detected	4	Succeed
Utility Tools	UC Browser	com.UCMobile	Detected	8	Succeed
Utility Tools	Shoujiduoduo Ringtone	com.shoujiduoduo.ringtone	Detected	12	Succeed
Utility Tools	Homework Helper	com.baidu.homework	Detected	13	Succeed
Utility Tools	Baidu NetDisk	com.baidu.netdisk	Detected	42	Succeed
Utility Tools	Thunder	com.xunlei.downloadprovider	Detected	74	Succeed
Utility Tools	WiFi Master Key	com.snda.wifilocating	Detected	77	Succeed
Utility Tools	Clean Master	com.cleanmaster.mguard	Detected	117	Succeed
Utility Tools	QQ Browser	com.tencent.mtt	Detected	871	Succeed

privacy. To prevent illegal invoking and make such procedure controllable, we proposed the location privacy-preserving scheme and implement the DLPA. DLP presents flexible and customizable strategies for users' location data preserving, while maintaining LBS functional. The evaluation result shows that DLP can resist various attacks and significantly provide users' location data privacy, meanwhile, high LBS quality and low energy cost are guaranteed.

However, latest applications, which gather background information and demographic identifiers of users, raise unprecedented privacy concerns about Persona data. Hence, in the future, to counter applications profiling, we will focus on enhancing trajectories generating with considering users background information.

APPENDIX

See the Table VI.

REFERENCES

[1] Z. Tian, Y. Wang, Y. Sun, and J. Qiu, "Location privacy challenges in mobile edge computing: Classification and exploration," *IEEE Netw.*, vol. 34, no. 2, pp. 52–56, Mar./Apr. 2020.

[2] Y. Zhang, M. Li, D. Yang, J. Tang, G. Xue, and J. Xu, "Tradeoff between location quality and privacy in crowdsensing: An optimization perspective," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 3535–3544, Apr. 2020.

[3] F. Zhao, L. Gao, Y. Zhang, Z. Wang, B. Wang, and S. Guo, "You are where you app: An assessment on location privacy of social applications," in *Proc. Int. Symp. Softw. Rel. Eng. (ISSRE)*, Memphis, TN, USA, 2018, pp. 236–247.

[4] J. Valentino-DeVries, "How your phone is used to track you, and what you can do about it," New York Times, New York, NY, USA, Rep., Aug. 2020. [Online]. Available: https://www.nytimes.com/2020/08/19/technology/smartphone-location-tracking-opt-out.html

[5] A. J. Dellinger, "Many popular Android apps leak sensitive data, leaving millions of consumers at risk," Forbes, Jersey City, NJ, USA, Rep., Jun. 2020. [Online]. Available: https://www.forbes.com/sites/ajdellinger/2019/06/07/many-popular-android-apps-leak-sensitive-data-leaving-millions-of-consumers-at-risk/

[6] J. Wakefield, "Google tracks users who turn off location history," BBC, London, U.K., Rep., Aug. 2018. [Online]. Available: https://www.bbc.com/news/technology-45183041

[7] K. Degirmenci, "Mobile users' information privacy concerns and the role of app permission requests," *Int. J. Inf. Manage.*, vol. 50, pp. 261–272, Feb. 2020.

[8] B. Liu *et al.*, "Follow my recommendations: A personalized privacy assistant for mobile app permissions," in *Proc. Symp. Usable Privacy Security (SOUPS)*, Denver, CO, USA, 2016, pp. 27–41.

[9] J. Lin, B. Liu, N. Sadeh, and J. I. Hong, "Modeling users' mobile app privacy preferences: Restoring usability in a sea of permission settings," in *Proc. Symp. Usable Privacy Security (SOUPS)*, Menlo Park, CA, USA, 2014, pp. 199–212.

[10] J. Reardon, Á. Feal, P. Wijesekera, A. E. B. On, N. Vallina-Rodriguez, and S. Egelman, "50 ways to leak your data: An exploration of apps' circumvention of the android permissions system," in *Proc. USENIX Security Symp. (USENIX)*, Santa Clara, CA, USA, 2019, pp. 603–620.

[11] K. Tam, A. Feizollah, N. B. Anuar, R. Salleh, and L. Cavallaro, "The evolution of Android malware and Android analysis techniques," *ACM Comput. Surveys*, vol. 49, pp. 1–41, Feb. 2017.

[12] X. Sun, L. Li, T. F. Bissyandé, J. Klein, D. Octeau, and J. C. Grundy, "Taming reflection: An essential step toward whole-program analysis of Android apps," *ACM Trans. Softw. Eng. Methodol.*, vol. 30, pp. 1–36, May 2021.

[13] L. Zhou, S. Du, H. Zhu, C. Chen, K. Ota, and M. Dong, "Location privacy in usage-based automotive insurance: Attacks and countermeasures," *IEEE Trans. Inf. Forensics Security*, vol. 14, pp. 196–211, 2019.

[14] V. Primault, A. Boutet, S. B. Mokhtar, and L. Brunie, "The long road to computational location privacy: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2772–2793, 3rd Quart., 2019.

[15] P. M. Asuquo *et al.*, "Security and privacy in location-based services for vehicular and mobile communications: An overview, challenges, and countermeasures," *IEEE Internet Things J.*, vol. 5, no. 6, pp. 4778–4802, Dec. 2018.

[16] X. Gong, X. Chen, K. Xing, D.-H. Shin, M. Zhang, and J. Zhang, "From social group utility maximization to personalized location privacy in mobile networks," *IEEE/ACM Trans. Netw.*, vol. 25, no. 3, pp. 1703–1716, Jun. 2017.

[17] B. Niu, Q. Li, X. Zhu, G. Cao, and H. Li, "Enhancing privacy through caching in location-based services," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Hong Kong, 2015, pp. 1017–1025.

[18] Google. (2021). *Android Platform: Location*. Accessed: May 18, 2021. [Online]. Available: https://developer.android.com/reference/android/location/Location

[19] Google. (2021). *User Location: Build Location-Aware Apps*. Accessed: May 24, 2021. [Online]. Available: https://developer.android.com/training/location

[20] S. Amri, F. Khelifi, A. Bradai, A. Rachedi, L. Kaddachi, and M. Atri, "A new fuzzy logic based node localization mechanism for wireless sensor networks," *Future Gener. Comput. Syst.*, vol. 93, pp. 799–813, Apr. 2019.

[21] AppInChina. (2021). *Appinchina: App Store Index*. Accessed: Jun. 2, 2021. [Online]. Available: https://www.appinchina.co/market/app-stores/

[22] M. Achour, M. Mana, and A. Rachedi, "On the issues of selective jamming in IEEE 802.15.4-based wireless body area networks," *Peer-to-Peer Netw. Appl.*, vol. 14, pp. 135–150, Jan. 2021.

[23] H. Alrahhal, M. S. Alrahhal, R. Jamous, and K. Jambi, "A symbiotic relationship based leader approach for privacy protection in location based services," *ISPRS Int. J. Geo-Inf.*, vol. 9, p. 408, Jun. 2020.

[24] J. T. Meyerowitz and R. R. Choudhury, "Hiding stars with fireworks: Location privacy through camouflage," in *Proc. Int. Conf. Mobile Comput. Netw. (MOBICOM)*, Beijing, China, 2009, pp. 345–356.

[25] Google. (2021). *Maps SDK for Android: Businesses and Other Points of Interest*. Accessed: Aug. 9, 2021. [Online]. Available: https://developers.google.com/maps/documentation/android-sdk/poi

[26] K. Fawaz and K. G. Shin, "Location privacy protection for smartphone users," in *Proc. Conf. Comput. Commun. Security (CCS)*, Scottsdale, AZ, USA, 2014, pp. 239–250.

[27] H. Li, H. Zhu, S. Du, X. Liang, and X. S. Shen, "Privacy leakage of location sharing in mobile social networks: Attacks and defense," *IEEE Trans. Dependable Secure Comput.*, vol. 15, no. 4, pp. 646–660, Jul./Aug. 2018.

[28] Google. (2020). *Permissions: Permissions on Android*. Accessed: May 18, 2021. [Online]. Available: https://developer.android.com/guide/topics/permissions/overview

[29] M. Gruteser and D. Grunwald, "Anonymous usage of location-based services through spatial and temporal cloaking," in *Proc. Int. Conf. Mobile Syst. Appl. Serv. (MobiSys)*, San Francisco, CA, USA, 2003, pp. 31–42.

[30] B. Gedik and L. Liu, "Protecting location privacy with personalized k-anonymity: Architecture and algorithms," *IEEE Trans. Mobile Comput.*, vol. 7, no. 1, pp. 1–18, Jan. 2008.

[31] C. Zhang and Y. Huang, "Cloaking locations for anonymous location based services: A hybrid approach," *GeoInformatica*, vol. 13, pp. 159–182, Jun. 2009.

[32] S. Zhang, X. Li, Z. Tan, T. Peng, and G. Wang, "A caching and spatial k-anonymity driven privacy enhancement scheme in continuous location-based services," *Future Gener. Comput. Syst.*, vol. 94, pp. 40–50, May 2019.

[33] A. Suzuki, M. Iwata, Y. Arase, T. Hara, X. Xie, and S. Nishio, "A user location anonymization method for location based services in a real environment," in *Proc. Int. Symp. Adv. Geograph. Inf. Syst. (ACM-GIS)*, San Jose, CA, USA, 2010, pp. 398–401.

[34] R. Kato, M. Iwata, T. Hara, Y. Arase, X. Xie, and S. Nishio, "User location anonymization method for wide distribution of dummies," in *Proc. Database Expert Syst. Appl. (DEXA)*, Prague, Czech Republic, 2013, pp. 259–273.

[35] T. Hara, A. Suzuki, M. Iwata, Y. Arase, and X. Xie, "Dummy-based user location anonymization under real-world constraints," *IEEE Access*, vol. 4, pp. 673–687, 2016.

[36] A. R. Beresford, A. C. Rice, N. Skehin, and R. Sohan, "MockDroid: Trading privacy for application functionality on smartphones," in *Proc. Workshop Mobile Comput. Syst. Appl. (HotMobile)*, Phoenix, AZ, USA, 2011, pp. 49–54.

[37] K. Micinski, P. Phelps, and J. S. Foster, "An empirical study of location truncation on Android," in *Proc. Mobile Security Technol. (MoST)*, San Francisco, CA, USA, 2013, pp. 1–10.

[38] S. Amini, J. Lindqvist, J. Hong, J. Lin, E. Toch, and N. Sadeh, "Caché: Caching location-enhanced content to improve user privacy," in *Proc. Int. Conf. Mobile Syst. Appl. Serv. (MobiSys)*, Bethesda, MD, USA, 2011, pp. 197–210.

[39] K. Fawaz, H. Feng, and K. G. Shin, "Anatomization and protection of mobile apps' location privacy threats," in *Proc. USENIX Security Symp. (USENIX'15)*, Washington, DC, USA, 2015, pp. 753–768.

Jiezhen Tang (Student Member, IEEE) received the B.Sc. degree from Xidian University, Xi'an, China, in 2016, where he is currently pursuing the Ph.D. degree with the College of School of Cyber Engineering.

His current research interests are cyber security and privacy.

Hui Zhu (Senior Member, IEEE) received the B.Sc. degree from Xidian University, Xi'an, China, in 2003, the M.Sc. degree from Wuhan University, Wuhan, China, in 2005, and the Ph.D. degree from Xidian University in 2009.

In 2013, he was with the School of Electrical and Electronics Engineering, Nanyang Technological University, Singapore, as a Research Fellow. Since 2016, he has been a Professor with the School of Cyber Engineering, Xidian University. His research interests include the areas of applied cryptography, data security, and privacy.

Rongxing Lu (Fellow, IEEE) received the Ph.D. degree from the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada, in 2012.

He is currently an Associate Professor with the Faculty of Computer Science, University of New Brunswick, Fredericton, NB, Canada. His research interests include applied cryptography, privacy enhancing technologies, and IoT-big data security and privacy.

Dr. Lu currently serves as the Vice-Chair (Conferences) of IEEE ComSoc Communications and Information Security Technical Committee.

Xiaodong Lin (Fellow, IEEE) received the first Ph.D. degree in information engineering from Beijing University of Posts and Telecommunications, Beijing, China, in 1998, and the second Ph.D. degree (with Outstanding Achievement in Graduate Studies Award) in electrical and computer engineering from the University of Waterloo, Waterloo, ON, Canada, in 2008.

His research interests include computer and network security, privacy protection, applied cryptography, computer forensics, and software security.

Hui Li (Member, IEEE) received the B.S. degree in radio electronics from Fudan University, Shanghai, China, in 1990, and the M.S. and Ph.D. degrees in telecommunications and information system from Xidian University, Xi'an, China, in 1993, and 1998, respectively.

He is currently a Professor with the State Key Laboratory of Integrated Service Networks, Xidian University. His research interests include networks and information security.

Fengwei Wang (Student Member, IEEE) received the B.Sc. degree from Xidian University, Xi'an, China, in 2016, where he is current pursuing the Ph.D. degree with the School of Cyber Engineering.

His research interests include the areas of applied cryptography, cyber security, and privacy.