Research Article

LPPS: A Distributed Cache Pushing Based K -Anonymity Location Privacy Preserving Scheme

Ming Chen,¹ Wenzhong Li,¹ Xu Chen,^{2,3} Zhuo Li,⁴ Sanglu Lu,¹ and Daoxu Chen¹

¹State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210046, China

²Institute of Computer Science, University of Göttingen, Göttingen, Germany

³School of Data and Computer Science, Sun Yat-Sen University, Guangzhou 510006, China

⁴School of Computer Science & Technology, Beijing Information Science & Technology University, Beijing 100192, China

Correspondence should be addressed to Wenzhong Li; lwz@nju.edu.cn

Received 3 December 2015; Accepted 2 June 2016

Academic Editor: Juan C. Cano

Copyright © 2016 Ming Chen et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Recent years have witnessed the rapid growth of location-based services (LBSs) for mobile social network applications. To enable location-based services, mobile users are required to report their location information to the LBS servers and receive answers of location-based queries. Location privacy leak happens when such servers are compromised, which has been a primary concern for information security. To address this issue, we propose the Location Privacy Preservation Scheme (LPPS) based on distributed cache pushing. Unlike existing solutions, LPPS deploys distributed cache proxies to cover users mostly visited locations and proactively push cache content to mobile users, which can reduce the risk of leaking users' location information. The proposed LPPS includes three major process. First, we propose an algorithm to find the optimal deployment of proxies to cover popular locations. Second, we present cache strategies for location-based queries based on the Markov chain model and propose update and replacement strategies for cache content maintenance. Third, we introduce a privacy protection scheme which is proved to achieve k -anonymity guarantee for location-based services. Extensive experiments illustrate that the proposed LPPS achieves decent service coverage ratio and cache hit ratio with lower communication overhead compared to existing solutions.

1. Introduction

Mobile social network applications are booming in recent years. With the rapid development of localization technologies (such as GPS) and Mobile Internet, mobile social network applications with location-based service (LBS) embedded are very popular such as Foursquare and Twitter. As a result, people use LBS more and more often. With the help of LBS, these mobile social network applications help users to connect to each other better. Typical applications include discovering popular restaurants in local area [1], traffic navigation [2], recommendation of friends nearby, and advertisement based on locations (such as "Promoted Tweets" [3]). According to the recent report, LBS is envisioned to become an over 10-billion-per-year business by the year 2016 [4].

As each coin has two sides, the LBS for mobile social applications may leak the users' location trajectory and so on, with a rising concern in location-based service about privacy

protection. In order to obtain LBS, a user needs to submit his query and location to the server and fetches the desired answer. The leak of user location information will increase the risk of adversary tracking the daily life of the user or will receive customized ADs which is unwilling or even revealing his private activities such as visiting a bank or going to a hospital [5]. It is important to protect user's location privacy for LBS.

Many efforts have been made to protect user's location privacy. The k -anonymity model was proposed in [6], which declared that when k -anonymity was satisfied, each individual should be indistinguishable from $k - 1$ other individuals. A user can achieve k -anonymity by sending out k queries with different locations to the server and choosing the desired answer from the responses. However, such method wastes network bandwidth and causes extra overhead in both client and server sides. A few works introduced trust-worthy middleware or cache proxies by using random noise to conceal

FIGURE 1: An example of location privacy preservation.

user's real ID and locations [7–13]. However, once the middleware or the proxies are compromised, the location privacy is not guaranteed. In a word, most existing works adopt the pull-based strategy where the user submits the location-based query to the LBS server or the third-party server. The disadvantage of such pull-based strategy is the difficulty of avoiding the compromise of LBS servers or proxies in different levels.

To deal with this problem, we propose location privacy preservation by using cache pushing. The basic idea is applying distributed cache proxies to store the most popular location-related data and pushing the data to the users proactively. If the desired data is available from the cache, the user does not need to send out the location-based query; thus his privacy is preserved.

We use Figure 1 to illustrate the motivation of our work. Assume a user is working on a road and he wants to request the information regarding location A (A could be a bank or a hospital, which is the sensitive information to be preserved). In the traditional pull-based strategy, the user will send the query to the LBS server. Once the LBS server receives the query, it knows that the user is heading for A. In our push-based strategy, the nearby popular location-related data about locations A, B, and C could have been stored in the cache proxy beforehand (e.g., such locations have been requested by other users previously and have been stored in the cache proxy), and they will be pushed to the user when he passes by the wireless access point. As a result, the user obtains the desired data without contacting the LBS server or reporting his location to the proxy. Since the LBS server is not aware of the query and the proxy only knows the user may go to A, B, or C, both of them cannot decide the user's real destination.

We propose the Location Privacy Preservation Scheme (LPPS) to achieve location anonymity. The LPPS need to answer three key questions: where to deploy cache proxies, how to organize and maintain cache content in proxies, and how to achieve k -anonymity assurance. We address these issues; we first introduce a greedy deployment algorithm to calculate the most frequently visit locations of mobile users (known as stay points) and deploys proxies to cover such regions as possible. Then we introduce a cache pushing strategy using group index to record the popular cache items and pushing the requested content to the users in batch. Cache replacement and updating strategies are proposed to

mine real popular places of interest from a lot of fake feedbacks when cache miss occurred. Thereafter, we propose a location privacy protection scheme to achieve k -anonymity assurance. To the best of our knowledge, this is the first work to apply push-based distributed caching architecture for privacy protection in location-based services.

Compared to the pull-based strategy, our push-based strategy has several advantages. First, it is easy to implement and resilient to the compromise of servers. Second, both the LBS server and proxies have only partial information of the mobile users (a user contacts the LBS server only when the requesting data is not found in the proxy), which avoids the continuous location tracking. Third, the deployment of distributed cache proxies will facilitate data queries and reduce the workload and network traffic to the LBS server.

Our main contributions are summarized as follows:

- (i) *A Location Privacy Preservation Scheme (LPPS)*. The proposed scheme introduces a distributed cache layer to store the most popular location-related data and push the data to mobile users, which reduces the communication overhead to the servers and enhances the location privacy of the users.
- (ii) *A Strategy for Cache Proxy Deployment*. We present a strategy to reveal the users' frequently visited locations and deploy cache proxies in such areas to improve the utility of caching data.
- (iii) *A Strategy for Organizing Cache Contents Based on Markov Chain Model*. We present a strategy to find the most popular location-related contents based on Markov Chain model constructing from history statistics and propose strategies to deploy popular cache contents on the proxies.
- (iv) *Distributed Cache Maintenance Strategies*. The proposed cache pushing strategy divides cache content into group and broadcast the cache items in batch. To maintain the cache contents, a cache updating strategy is applied by using the theorem of ball-to-bins to identify real popular location-related data.
- (v) *K-Anonymity Location Privacy Preservation*. We proved that the proposed privacy preservation scheme based on cache pushing can achieve k -anonymity assurance, which is resilient to the compromise of LBS server or proxies.
- (vi) *Trace Driven Simulations*. We evaluate the system performance using the real mobile trace collected in Beijing and cab trace collected in San Francisco, which shows the efficiency of LPPS.

The rest of the paper is organized as follows. Section 2 reviews the related work on system architecture and models for privacy preservation in LBS. Section 3 presents details of Location Privacy Preservation Scheme (LPPS) that provides k -anonymity location privacy. Section 4 conducts experiments to evaluate the efficiency of the proposed LPPS strategy by both real and synthetic datasets. Section 5 concludes this paper.

2. Related Work

In this section, we review the related work on system architecture and models for privacy preservation in LBS and distributed architectures and pull-push algorithms used in other domains.

2.1. Privacy Preservation Architectures. There are three kinds of most used architectures of privacy preservation for LBS, that is, noncooperative architecture [14], centralized architecture [11, 13, 15], and peer-to-peer architecture [16, 17]. In the first one, users only use their own knowledge to protect their location privacy. In the second one, a trusted third-party component is added to provide location privacy protection for users. In the third one, users preserve their privacy cooperatively.

In the noncooperative architecture, mobile clients are assumed to have powerful computation capability and can complete the privacy preservation of their own requirements by generating fake locations or pseudonyms independently [14]. The advantage of this structure is the simpleness and easiness to deploy. However, due to the lack of global information, the ability of privacy preservation is weak.

In the centralized architecture, a third-party component is responsible for location anonymization incorporating with the user and LBS server [11, 13, 15]. This trusted middleware improves the capacity of the privacy protection. But the disadvantage is that it may become the bottleneck of the whole system and easy to be attacked.

The peer-to-peer architecture [16, 17] assumes users are cooperative and they trust each other. Each mobile user has enough capacity for computing and storage. However, it cannot avoid attacks from malicious users.

2.2. Privacy Preservation Models. The most commonly used model to evaluate privacy protection level is k -anonymity, introduced by Sweeney in [6]. In this model, attackers cannot distinguish one user from a k user group. Privacy preservation models can be categorized into three kinds as follows.

2.2.1. Submission of Fake Position [8, 18]. Users send fake positions to the server. The fake positions have to be around the real ones in order to get the information or service needed for users. The distance between them is related to the degree of privacy protection and quality of service. It is found that the larger distance yields the higher protection degree and the poor service quality. Li et al. [18] introduce a model in which users send several fake positions to the server and request points of interest surround these fake positions and to conceal the real point of interest the user wants.

2.2.2. Pseudonym [9, 10, 19]. Hiding the identities of users through pseudonyms were presented in [19], where privacy protection degree depends on the strength of the relationship between users' identities and particular locations. To achieve better anonymous, users need to update their pseudonyms frequently [9]. Beresford and Stajano [10] introduced a model based on dynamic pseudonyms, in which spaces are

separated into "Mix Zones" and "Application Zones." Every time users move into "Mix Zones," the pseudonyms are changed. Reference [20] proposes an incentive mechanisms to encourage users to cooperate with each other in this kind of "Mix Zones." Liu et al. investigated the optimal multiple Mix Zones placement problem for location privacy protection [21]. Gong et al. [22] take social ties into consideration and motivate users to participate in socially-aware pseudonym change game. However, attackers may find the patterns for the changes of pseudonyms. For this issue, a method involving path confusion was developed in [23]. However, the service delay may be large in this method. Taking prediction into consideration, CacheCloak [7], which is also a kind of path confusion, uses predictions to protect the privacy and decreases the delay of service. However, it needs a third-party server and it takes a lot of resources for calculating and storage space. Zhu and Cao proposed APPLAUS in which colocated Bluetooth enabled mobile devices to mutually generate location proof, and update to a location proof server [24]. Periodically changed pseudonyms are used by the mobile devices to protect source location privacy from each other and from the untrusted location proof server. However, it needs a third trusted party Certification Authority server and takes a lot of resources for calculating.

2.2.3. Spatiotemporal Cloaking [11–13]. Users usually send a region instead of their accurate position to enhance the risk of location privacy. Such service usually needs to be delayed for a period of time for collecting more than one user's requests in the same region to send them to the LBSs at once. Attacker can only know that the users are in a particular region in a particular period of time and thus cannot track a user continuously. The server has to retrieve query results in the user's cloaking region that protect the privacy of users. However the server workload dramatically increases, as well as the amount of downloading traffic. Further, not only may the service be delayed, but also the performance of LBS may not be good because of inaccurate location information. Galdames and Cai proposed processing queries as a batch instead of one by one independently to reduce the workload of servers [25]. Wang et al. proposed L2P2 to find the smallest cloaking area for each location request so that diverse privacy requirements over spatial and/or temporal dimensions are satisfied for each user [26]. Vu et al. proposed a mechanism based on locality-sensitive hashing (LSH) to partition user locations into groups each containing at least K users (called spatial cloaks) [27]. The mechanism is shown to preserve both locality and K -anonymity.

2.2.4. Encryption [28, 29]. The server encrypts the users' location and query separately and stores them on different database; it could prevent users from information leak. Xue et al. designed a novel index structure to provide fast search for users when they check in at the same venue frequently and outsources the heavy cryptographic computations to the server to reduce the computational overhead for mobile clients [28]. Li and Jung designed a Location Query Protocol (PLQP), which allows different levels of location query on encrypted location information for different users, and it is

efficient enough to be applied in mobile platforms. However, these kinds of encryption take a lot of resources for calculating, communication, and storage space [29]. Wei et al. separate user identities and anonymized location updates onto two entities. Users' location privacy is protected if either entity is compromised by the adversary [30]. However, if both entities are compromised, it could not prevent users from information leak.

2.3. Distributed Architectures and Pull-Push Algorithms. Distributed architectures and pull-push algorithms have been used for many general applications such as social networks, p2p networks, and location-based advertising. Doer et al. analyzed why rumors spread fast in social networks; one of the reasons is that nodes of small degree use push-pull models to build a shortcut between those having large degree (hubs): node of a small degree quickly pulls the rumor from node with large degree and again quickly pushes it to another node with large degree [31]. Zhang et al. presented an unstructured peer-to-peer network called GridMedia for live media streaming employing a push-pull approach which greatly reduces the latency and inherits most good features (such as simplicity and robustness) from the pull method [32]. Unni and Harmon presented effectiveness of push versus pull mobile location-based advertising; the results indicate that privacy concerns are high, and perceived benefits and value of LBA are low [33].

In summary, most existing architectures and models for privacy preservation in LBS are pull-based, which are hard to prevent the compromise of servers. Different from their works, we propose a push-based strategy to reduce the risk of server compromise while preserving k -anonymity location privacy. To the best of our knowledge, this is the first work on proactive push-based privacy protection scheme for location-based services.

3. LPPS: A Location Privacy Preservation Scheme Based on Distributed Cache Pushing

In this section, we present a Location Privacy Preservation Scheme (LPPS) that provides k -anonymity location privacy by using distributed cache pushing.

The basic idea of LPPS is deploying a distributed cache layer to store the most popular location-related data and pushing the data to mobile users in range. By listening to the broadcast from the cache layer, the mobile users are able to obtain the desired data without querying the LBS server, which will eliminate the chance of leaking their location information.

The system architecture is illustrated in Figure 2. There are three major components in the system: LBS server, distributed cache layer, and mobile users. The distributed cache layer could be physically deployed on wireless access points or could be a set of proxy servers connecting to the Internet. Throughout this paper, we refer to the entities in the cache layer as *cache proxies*. Cache proxies are distributed and

FIGURE 2: System architecture of LPPS.

deployed in different regions and they can work independently without knowing the existence of each other. Each cache proxy only stores the location-related data. When a mobile user is within the communication range, the cache proxy pushes the caching data to the user. If the user's desired data is found in the cache, we call it *cache hit*; otherwise we call it *cache miss*. When cache miss occurs, the user will send $k - 1$ random queries with the real one to the LBS server. When the LBS server sends back the response, the user generates a *feedback index* containing the same $k - 1$ random location/keywords with the real one to the cache proxy to inform the proxy to update cache. Proxy uses the *feedback index* to request cache data from LBS server and get *feedback content* later, which could update its cache content.

The threat model is defined as follows: we assume both LBS server and distributed cache layer are possible to be compromised, and the mobile users could trust themselves. The proposed scheme achieves the following goal: even though the LBS server or the distributed cache layer is compromised, they cannot obtain enough information to track the entire trajectory information of the mobile users.

There are three key questions in implementing LPPS: where to deploy the cache proxies, how to organize, maintain, and distribute cache content in proxies, and how to guarantee k -anonymity. In the following sections, we introduce the strategies for cache proxy deployment, cache maintenance, and cache pushing for privacy preservation. In Figure 3, we show the three key questions and brief answers.

3.1. Where to Deploy Cache Proxies. In this subsection we answer the question "where to deploy cache proxies." In order to maximize the usage of caching content, cache proxies should be deployed near to the places which are frequently visited by users. To find out such "hot spots," we need the historical movement trajectories of users. In fact, we do not care who the users are and users could be anonymous. There are a lot of historical movement trajectories like these, where we can get the historical movement trajectories of users but cannot find a sequence of trajectories belongs to them, because

FIGURE 3: The scheme of LPPS.

the users are anonymous. Normally, when government do a city plan, it can refer to data sets like taxi and bus trajectories. It is possible to obtain anonymous trajectory data of a city. Such mobility trajectory datasets is openly available [2, 34].

Physically deploying proxies may cost a lot on wireless access points. Although the cost is not cheap on power, machines, and renting places for deployment proxies, it has benefits and motivation to deploy distributed cache proxies based on the existing public infrastructure to provide LBS for people passing by, especially for today's O2O commercial business. In New York, the de Blasio administration's plan to convert aging pay phones to Wi-Fi hot spots won unanimous approval from a review committee, clearing the way for the installation of thousands of Wi-Fi sites across New York City over the next decade [35]. Our work could be helpful for this kind of city plans. Duan et al. [36, 37] analyze different mechanisms to motivate the collaboration of smartphone users on both data acquisition and distributed computing; they propose a reward-based collaboration scheme for data acquisition applications and use contract theory for distributed computing applications. Our work could be practically rolled out using a reward-based collaboration scheme to encourage users to collaborate in our system to reduce the cost. Besides, this system can be practically rolled out with VANET (Vehicular Ad hoc Network); our cache proxies here could be road side unit in VANET [38] which can not only reduce the cost, but also provide more kinds of services.

The historical trajectory of user i is represented by $L^i = \{L_0^i, L_1^i, \dots\}$. Consider $L_m^i = (x_m^i, y_m^i, t_m^i)$, where x_m^i and y_m^i are the geographical coordinates of the m th position in the trajectory of user i . t_m^i is the timestamp denoting when user i is at the m th position. Without loss of generality, $t_m^i < t_{m+1}^i$.

The "frequently visited locations" have temporal and spacial meanings. We introduce the concept of *stay points* to represent them. A stay point is defined as the location that a mobile user stays in the area within Δ_d distance for at least Δ_t time, where Δ_d and Δ_t are the thresholds for spacial and temporal differences. The set of stay points S^i of user i can be obtained as follows. Starting from a location $L_m^i \in L^i$, we calculate the largest set $\{L_{m+1}^i, \dots, L_{m+q}^i\}$ that satisfies $(t_{m+q}^i - t_m^i \geq \Delta_t) \wedge (\|L_u^i - L_m^i\| \leq \Delta_d, \forall L_u^i \in \{L_{m+1}^i, \dots, L_{m+q}^i\})$, where $\|L_u^i - L_m^i\| = \sqrt{(x_u^i - x_m^i)^2 + (y_u^i - y_m^i)^2}$ is the Euclidean distance between the two locations. If there exists such a set, we can construct a stay point with coordinate $x_m = (\sum_{u=m+1}^q x_u^i)/q$ and $y_m = (\sum_{u=m+1}^q y_u^i)/q$ accordingly. After

having x_m and y_m , we let $S^i = S^i \cup \{(x_m, y_m)\}$ and remove the set $\{L_m^i, L_{m+1}^i, \dots, L_{m+q}^i\}$ from L^i . Starting from the first location, we repeat the process until L^i is empty. As a result, we obtain the set of stay points S^i to represent the characteristic locations of user movement and filter out the locations which the user only visits temporarily.

With the set of stay points obtained, we are able to devise the strategy of cache proxy deployment. The detailed strategy includes the following steps.

Step 1 (divide the geographical area into grids). Each unit in the grid is a small square of size $r \times r$, which is the communication area of a cache proxy and covered by several Wi-Fi APs. Reference [39] shows the bandwidth of Wi-Fi AP when transmitting location-related caches. Assume there are $A \times B$ squares in total. Each square is assigned a unique ID for reference.

Step 2 (construct the transitions matrix P). We firstly map trajectory of users to their staypoint list; then we map each staypoint list to the squares ID list by using the location information. With such transformation, the user trajectory can be represented by a sequence of square IDs. Now we build a weighted graph $G(V, E)$, V is the set of nodes indicated by square IDs, and E is the set of edges (if a user moves from a node to another, there is an edge between them).

We define the weight $\omega_{i \rightarrow j}$ on the edge (i, j) as the number of users moving from location i to location j , which can be computed as follows. Let $\omega_{i \rightarrow j} = 0$ at the beginning. If a user is found moving from square ID_i to squares ID_j (source stay point is in square ID_i and destination stay point is in squares ID_j), let $\omega_{i \rightarrow j} = \omega_{i \rightarrow j} + 1$. After scanning over the whole trajectory, we get $G(V, E, W)$ with weight ω on every edge.

Step 3 (calculate the visiting frequency). To find the most popular grids, we define a value $VisiteWeight_{ID_i}$ to represent the popularity of the i th grid. $VisiteWeight_{ID_i} = \sum_{j=1}^{A \times B} \omega_{i \rightarrow j}$.

Step 4 (deploy cache proxies). Assume N is the number of cache proxies we plan to deploy. We sort the elements of $VisiteWeight_{ID_i}$ in descending order and choose the top N squares to deploy the cache proxies.

3.2. How to Organize and Maintain Cache Content in Proxies. To answer the problem how to organize and maintain cache content in proxies, we introduce a model to capture the

FIGURE 4: An example of the Markov chain.

mobility pattern of user. We use the Markov chain model to describe the query interests and mobility pattern of users. Based on the model, we propose cache placement, cache pushing, and cache update strategies.

3.2.1. The Markov Chain Model. We use Markov chain to describe the probabilities of a user moving from one location to another. States of the Markov chain are the popular locations found by using the previous mechanism, and the transient probabilities are the probabilities of users moving between locations. The Markov chain can be built as follows.

Based on the weighted graph mentioned above, we can calculate the probability that a user moving from location i to location j as $P_{i,j} = \omega_{i \rightarrow j} / \sum_{j=1}^{A \times B} \omega_{i \rightarrow j}$, which satisfies $\sum_{j=1}^{A \times B} P_{i,j} = 1$.

In our model, $P = [P_{i,j}]$ forms the one-step transit matrix of the Markov chain. The probability of moving from one square to another could be measured by counting the visit frequency. For example, after counting all the transmissions on all the staypoints, we can easily get the probability from one staypoint to another staypoint:

$$P_{i,j} = \frac{\omega_{i \rightarrow j}}{\sum_{j=1}^{A \times B} \omega_{i \rightarrow j}}. \quad (1)$$

Figure 4 is an example of the Markov Chain. In this example, $P_{1,2} = 6/(6+9+5) = 30\%$. Given an initial configuration of user distribution, the users move to other locations with the probabilities defined by the transit matrix; thus the distribution evolves over time. Let $\pi^{(t)} = \{\pi_1^{(t)}, \pi_2^{(t)}, \dots\}$ be the distribution at time t , where $\pi_i^{(t)} = \Pr[X_t = i]$ is the percentage of users in location i at time t . At the beginning, the distribution is $\pi^{(0)}$. After one time slot, the distribution becomes $\pi^{(1)} = \pi^{(0)}P$. The system evolves as $\pi^{(0)} \xrightarrow{P} \pi^{(1)} \xrightarrow{P} \pi^{(2)} \xrightarrow{P} \dots \pi^{(t)}$. By induction, it obtains $\pi^{(t)} = \pi^{(0)}P^t$. When the system runs for a long time (i.e., $t \rightarrow \infty$), regardless of what the initial distribution was, $\pi^{(t)}$ converges to a stable distribution. The stable distribution can be obtained by solving the equation $\pi = \pi P$, which represents the stable distribution of mobile users in different locations for a long time.

Three pieces of information are delivered by the Markov chain convergence theorem [40] regarding the stationary distribution: (1) existence: there exists a stationary distribution.

(2) Uniqueness: the stationary distribution is unique. (3) Convergence: starting from any initial distribution, the chain converges to the stationary distribution.

For the existence of stationary distribution, neither irreducibility nor aperiodicity is necessary for the existence of a stationary distribution. In fact, any finite Markov chain has a stationary distribution. Irreducibility and aperiodicity guarantee the uniqueness and convergence behavior of the stationary distribution.

In our problem, if the Markov Chain is reducible, it means users movement is in separated regions. In that case, we can divide the map to submaps to ensure the Markov Chain for each submap is irreducible. And it is a periodicity because it is hard for all the loop in the graph to be divided by 2. So the Markov Chain could have a stationary distribution π . The stationary distribution can be derived by solving the equation $\pi = \pi P$, or it can be obtained by running $\pi^{(t)} = \pi^{(0)}P^t$ for a sufficient long time t .

3.2.2. Cache Placement Strategy. We manage cache content on proxies as follows. We only consider the short-term movement of users; thus we predict the future movement for three steps; that is, $\pi^{(t+3)} = \pi^{(t)}P^3$. For each *cacheProxy*_{ID_i}, let the initial distribution $\pi_{ID_i}^{(0)} = [0, \dots, 0, 1, 0, \dots, 0]$ (all users in square ID_i). Then we calculate $\pi_{ID_i}^{(3)} = \pi_{ID_i}^{(0)} \times P^3$. The vector $\pi_{ID_i}^{(3)}$ indicates the probability that the user moves to other locations in three steps. Given such information, our cache placement and replacement principle are choosing the most probable location to cache and evicting the less probable visited locations.

3.2.3. Cache Pushing Strategy. To preserve users' location privacy, the LPPS scheme pushes cache content to mobile users proactively. The cache pushing strategy of cache hit is described as follows.

Step 1. The cache proxy divides the cache items into several groups with each group having K items. Each item is in the format $\langle \text{Square ID}, \text{Keyword}, \text{Data} \rangle$. To avoid forming patterns, the cache items are grouped together randomly. Each group generates an index in the form $\{\langle \text{Square ID}_i, \text{Keyword}_i \rangle\}$, which indicates the set of cache items in the group. The group IDs and their index are pushed to the users.

Step 2. When a user receives the group index, he compares his query which follows the format $\text{Req} = \langle \text{Loc}, \text{Keyword} \rangle$ with the index. If cache hits, the user will send back the group ID to the cache proxy to acquire the data.

Step 3. On receiving the group ID, the cache proxy will push the cache items in the same group to the user.

Step 4. The user receives the grouped cache items and chooses the desired data. Due to the nature of wireless communication, if other users acquiring data in the same group overhear the broadcast, they can use the data without interaction with the cache proxy.

Input:

Pushing round number, n ;
 K -anonymity, K ;
 Cache proxies $Caches$, $Caches = \{Cache^0, Cache^1, \dots\}$;
 All cache data in $Cache^i$, $Cache^i = \{Cache^i_0, Cache^i_1, \dots\}$;
 For $Cache^i$, each item is in the format $\langle Square\ ID, Keyword, Data \rangle$;
 For $Cache^i$, each group generates an index in the form $\{\langle Square\ ID_i, Keyword_i \rangle\}$;
 Received feedback index for $Cache^i$, $FeedbackIndexList^i$;
 Received feedback content for $Cache^i$, $FeedbackContentList^i$;

Output:

None

```

(1) while True do
(2)    $Cache^i$  broadcasts following to users passing-by  $n$  round with each round  $K$  items:
       $\{\langle Square\ ID, Keyword \rangle\}$  with cache group index.
(3)   if  $Cache^i$  receives the grouped cache items index  $j$  then
(4)      $Cache^i$  broadcasts the  $j$ th group of  $\{\langle Square\ ID_i, Keyword_i, Data \rangle\}$ .
(5)   end if
(6)   if  $Cache^i$  receives feedback index  $F'$  from user then
(7)      $FeedbackIndexList^i = FeedbackIndexList^i \cup F'$ 
(8)     if  $Cache^i$  needs to update its cache then
(9)        $Cache^i$  requests feedback content  $F$  about feedback index  $F'$  from LBS server.
(10)       $FeedbackContentList^i = FeedbackContentList^i \cup F$ 
(11)      if  $FeedbackContentList^i$  reaches the limit of cache size then
(12)        LRU (Least Recently Used) policy will be used replace  $Cache^i$ 
(13)      end if
(14)    end if
(15)  end if
(16) end while
(17) return;
```

ALGORITHM 1: Cache pushing strategy for each cache proxy $Cache^i$.

If the requested data is not found in the index, a cache miss happens. The cache pushing strategy of cache miss is described as follows.

Step 1. It is the same as Step 1 in the cache pushing strategy of cache hit.

Step 2. When a user receives the group index, he compares his query which follows the format $Req = \langle Loc, Keyword \rangle$ with the index. If cache miss occurs, the user needs to retrieve the data from the LBS server. To protect location privacy, the user generates $k - 1$ queries with random locations/keywords and sends them with the original query to the LBS server.

Step 3. After the LBS server sends back the k responses to user, the user uses them to generate a feedback index in the form $\{\langle Square\ ID_i, Keyword_i \rangle\}$; then the user sends back the index to the cache proxy.

Step 4. On receiving the feedback index in the form $\{\langle Square\ ID_i, Keyword_i \rangle\}$, the cache proxy could store it for a time before it needs to update its cache. We assume cache proxy could be disconnected with LBS servers or Internet when it is not updating cache. When the cache proxy needs to update its cache, the cache proxy uses stored feedback indexes

as requests to query about each $\langle Square\ ID_i, Keyword_i \rangle$ from LBS server and then receives the data from LBS server as feedback content to update its cache. If cache proxy reaches the limits of cache size, a LRU policy will be used to replace cache. Algorithm 1 shows the pseudocode of our cache pushing strategy for each cache proxy.

3.2.4. Cache Updating Strategy. Now we consider cache updating and replacement strategy when cache miss occurs. As we know, popular locations change when peoples' interest changes. For example, as a new interesting shop or restaurant begins to work, people around may go to these new places frequently, which forms new hot spots. Cache replacement and updating strategy should consider the dynamic of user visiting interests on cache miss.

After cache miss happens for several times, cache proxy could get a large amount of feedback indexes from users. Since it contains fake random requests due to the reason of privacy preservation, we want to identify the real popular locations from the feedback indexes. We use a bin-ball model introduced by [40] to achieve this. As shown in Figures 5 and 6, because the real hot places are always needed by users, the accumulative probability that a location occurs should be higher than random requests. In Figure 5, we use black

FIGURE 5: Balls to bins model: feedback as balls, locations as bins.

FIGURE 6: Balls to bins model: bins receiving more balls stands for real popular locations.

balls to indicate the real requests and white balls to indicate the fake requests (queries with random locations/keywords). Each bin indicates a location related to the requests. The location-based queries equal putting balls (black and white balls) into bins. Each white ball drops to the random bins uniformly, corresponding to users queries with random locations/keywords. So each bin will get $(k-1) \times T/L$ white balls as expected, where T is the total number of cache missing that happened and L is the total number of locations users may need. The distribution of black balls is not random: it is proportional to the popularity of the location. So the distribution of black balls is very different from the white balls, resulting in the fact that a few bins will get most black balls as Figure 6 shows. Finding the bins containing most black balls can indicate the real popular locations.

Suppose we have M balls indicating the number of locations in the feedback indexes and L bins indicating total number of locations. Suppose all balls are thrown into bins independently. Let X_i be the random variable representing the number of balls in the i th bin. We calculate the probability that a bin is receiving more than Q balls. If all balls are uniformly randomly dropped, the Q balls are thrown to bin

i with probability $(1/L)^Q$, and there are totally $\binom{M}{Q}$ distinct combinations of M balls. Therefore,

$$\begin{aligned} \Pr[X_i \geq Q] &\leq \binom{M}{Q} \left(\frac{1}{L}\right)^Q = \frac{M!}{Q! (M-Q)! L^Q} \\ &= \left(\frac{1}{Q!}\right) \prod_{i=0}^{Q-1} \frac{M-i}{L} \leq \left(\frac{1}{Q!}\right) \left(\frac{M}{L}\right)^Q. \end{aligned} \quad (2)$$

L is the number of total locations and M is the numbers of locations in the feedback indexes. In real practice, $M \ll L$, which indicates $(1/Q!) \times (M/L)^Q$ approaches 0 for larger Q .

If Q is big enough, $\Pr[X_i \geq Q]$ is a very small number indicating that it is a small probability event. If a bin receives more than Q balls, it means a small probability event occurs, which indicates that the throwing of balls is not uniformly random, and the bin represents a popular location in real world. For example, if $M = 1000$, $L = 10000$, and $Q = 10$, we obtain $\Pr[X_i \geq Q] \leq 2.7557319223986 \times 10^{-17}$. If such event occurs, it is very confident that the i th location is a popular location, which should be chosen to update to cache proxy.

We propose cache updating principles as follows. For each cache proxy i , when it got M locations in its feedback indexes, it identifies the popular locations by calculating the probability in (2) and requests for feedback content and then inserts the corresponding location-related items into the cache proxy. The least visited locations are evicted from the cache to make room for the new items. The detailed algorithm is shown in Algorithm 2.

3.2.5. Discussion and Analysis. In this section, we discuss the efficiency of the proposed LPPS scheme.

First, the deployment of cache proxy is greedy. Since we choose the most visited square to deploy cache proxy, it is trivial that this is approximate optimal place to deploy the proxies.

Second, the placement of cache content is based on movement prediction, which is optimized by using Markov Chain analysis. In order to find proper location-related content to push, we need to find the most related location with current location, which could be calculated by the transition matrix P . We pick the most probable locations to push to the mobile users passing, which can reduce the risk of location report and the communication overhead to the LBS server.

Third, we analyze the computational cost of the proposed algorithm, which involves two steps:

- (1) Using original data to compute where to deploy cache proxies: Since we choose the most visited square to deploy cache proxy, if the input is the total points number p of historical trajectory and the total grids number n (the area is divided into a $n = A \times B$ grids), the time complexity is $O(p)$, and the space complexity is $O(n^2)$. In fact, this computational cost is huge but it only needs to be calculated once.
- (2) Organizing and maintaining cache content in proxies:

(2.1) To organize cache content, we need to use $P_{i,j} = \omega_{i \rightarrow j} / \sum_{j=1}^{A \times B} \omega_{i \rightarrow j}$, and then we calculate $\pi_{IDi}^{(3)} =$

Input:

Transition Matrix, P ;
 Pushing round number, n ; K -anonymity, K ; Cache proxies $Caches$, $Caches = \{Cache^0, Cache^1, \dots\}$;
 All cache data in cache proxies $Cache^i$, $Cache^i = \{Cache_0^i, Cache_1^i, \dots\}$;
 Received number of locations in $FeedbackIndexList^i$ for $Cache^i$, $ReceivedFeedback^i$;
 Location numbers L ;
 Desired feedback threshold Q ;
 The ball distribution in all bins for $Cache^i$, $P^i = \{P_0^i, P_1^i, \dots\}$;

Output:

$Cache^i$;

- (1) **for** each proxies $Cache^i \in Caches$ **do**
- (2)   **if** $ReceivedFeedback^i \geq L$ **then**
- (3) $ReceivedFeedback^i = 0$;
- (4) $newCache = \phi$;
- (5)   **for** each $P_j^i \in P^i$ **do**
- (6)   **if** $P_j^i \geq Q$ **then**
- (7)   request feedback content about location j from LBS servers
- (8) $newCache = newCache \vee \{cache \text{ about location } j\}$
- (9)   **end if**
- (10)   **end for**
- (11)   Sort $Cache^i$ with Descending order of $P^3[i, j]$
- (12)   Sort $newCache$ with Descending order of P_j^i
- (13) $Cache^i = newCache \vee \{Top(n \times K - |newCache|)inCache^i\}$;
- (14)   **end if**
- (15) **end for**
- (16) **return** $Cache^i$;

ALGORITHM 2: Cache replacement and updating.

$\pi_{IDi}^{(0)} \times P^3$. The vector $\pi_{IDi}^{(3)}$ indicates the probability that the user moves to other locations in three steps. Our cache placement and replacement principle is choosing the most popular locations to cache and evicting the less popular visited locations. If the city is divided into $n = A \times B$ grids, in order to calculate $\pi_{IDi}^{(3)}$, the time complexity is $O(n^3)$, and the space complexity is $O(n^2)$. Fortunately, the computational cost to calculate $\pi_{IDi}^{(3)}$ is huge but it also only needs to be calculated once. After that, if the cache size of $cacheProxy_{IDi}$ is S , we only sort and choose top S most popular locations of n as the cache content for $cacheProxy_{IDi}$. In order to choose the most S popular locations to cache, the time complexity is $O(n \log(n))$, and the space complexity is $O(n)$.

- (2.2) To maintain cache content in proxies, we need the resource including not only the bandwidth resource between proxies and LBS servers for updating cache, but also the computational resource for proxies to calculate which part of cache needs to be replaced and updated, which we explain as follows. We use Algorithm 2 to replace and update cache. In order to find the popular location data which is not in origin cache, we analyze the feedback indexes and use

balls to bins model to get the real popular location data in feedback content which is requested from LBS servers. Then we update cache with it to replace the least popular caches in the origin cache. The same as (2.1), in order to sort and choose the most S popular locations to cache, our algorithm needs $O(n \log(n))$ time complexity and $O(n)$ space complexity.

In summary, although the computational cost which only needs to be calculated once is $O(n^3) + O(p)$ time complexity and $O(n^2)$ space complexity, the computational cost of keeping our system working is $O(n \log(n))$ time complexity and $O(n)$ space complexity.

3.3. How to Guarantee k -Anonymity Location Privacy Preservation. We show that LPPS achieves k -anonymity of location privacy to both cache proxy and LBS server.

When cache hits occurs, the cache proxy received the group ID from the user. That is, the cache proxy knows that the desired data is within the group. However, there are k different items in the group and they are randomly chosen; the proxy is not able to distinguish the target from the other $k - 1$ items. When cache hits occurs, LBS server and user do not communicate, and LBS server is not able to track user.

When cache miss occurs, the user sends out location-based queries with fake locations and keywords. Since the LBS server receives k different queries from the same user,

it is hard to distinguish the real request from the other $k - 1$ requests. The proxy also receives a feedback index from user, and it is hard to distinguish the real one from the other $k - 1$ locations/keywords in the feedback index too. Besides, the LBS server receives user requests intermittently (only when cache miss occurs), which will make it impossible to track the user location continuously.

In the threat model, proxies and the LBS server are possible to be compromised. They can collude and try to track users. This did not affect the k -anonymity of location privacy. When cache hit occurs, the proxies only know the k destinations the user may be heading to, and the LBS server does not have any information about the user. So it guarantees k -anonymity of location privacy when cache hit occurs. When cache miss occurs, both proxies and the LBS server receive the k locations/keywords with $k - 1$ undistinguished random locations/keywords in it; k -anonymity of location privacy is also guaranteed. In fact, both LBS server and the proxy only know the k probable destinations that the user is heading to and do not know more things; they cannot obtain enough information to track user. Our protocol can keep both LBS server and distributed cache layer only having partial information of user; even when they collude, they still could not obtain enough information. Even if neighbor proxies collude, they still only get the k probable next destinations that user is heading to.

In summary, the proposed LPPS guarantees k -anonymity of location privacy.

4. Performance Evaluation

In this section, we conduct experiments to evaluate the efficiency of the proposed LPPS strategy by both real and synthetic datasets.

4.1. Experiment Setup. The two real datasets are the mobile user traces collected in Beijing [2] and cab mobility traces collected in San Francisco [34]; the statistics of trace datasets are in Table 1. The historical traces are obtained from public published GPS dataset GeoLife and Cabspotting project. The Geolife dataset (published by Microsoft Research Asia) logged the mobility traces (the GPS coordinates) of 178 users; most of the trace locations are in Beijing in a period of over four years (from April 2007 to October 2011). Cabspotting project logged the cabs mobility traces (the GPS coordinates) of 536 users collected in May 2008.

Our simulation area is a square in the Beijing city ranging from latitude 39.75°N to 40.1°N and from longitude 116.2°E to 116.66°E and San Francisco ranging from latitude 37.6°N to 37.85°N and from longitude 122.55°W to 122.23°W . We remove the trajectories which are outside these area from the traces and apply the method described in Section 3.1 to calculate the stay points (with $\Delta_d = 600\text{ m}$ and $\Delta_t = 1200\text{ s}$). We obtain 13,666 and 1,894,208 stay points in total. We assume that the communication area for one proxy is $600\text{ m} \times 600\text{ m}$, and both divide the simulation area into a $A \times B$ grid (GeoLife 65×65 and Cabspotting 47×47 the experiment). We apply the cache proxy deployment strategy to calculate the visiting frequency and deploy N cache proxies

TABLE 1: Statistics of trace datasets.

Traces	Geolife	Cabspotting
Number of users	178	536
Region	Beijing	San Francisco
Duration	4 years	30 days
Number of stay points	13,666	1,894,208
Number of queries generated	7,693	9,266,161
Number of grids	65×65	47×47

in the most popular locations. Figure 7 illustrates the result of cache proxy deployment when N is set to 5% of 65×65 in GeoLife dataset. In this figure, one blue dot indicates one proxy and there are 211 proxies in this figure. Figure 8 shows the result of cache proxy deployment when N is set to 5% of 47×47 in Cabspotting dataset. In this figure, one blue dot indicates one proxy and there are 110 proxies in this figure.

Based on the GPS dataset, knowing the current location, each mobile user generates a location-based query to request location information for his/her next stay point. The query is in the form of $\langle Loc, Keyword \rangle$. We set Loc as the location the user is heading to, which is available from the mobile trace. There are total 7,693 queries generated in Geolife and 9266161 in Cabspotting. For the required k -anonymity, we set $k = 5$.

The synthetic dataset is generated in accordance with the following characteristics. In our synthetic dataset experiment, users distributed in a large area of 100×100 grids; each grid is also $600\text{ m} \times 600\text{ m}$. They generate a number of queries every minute. The distribution of the locations where these queries are generated from obeys the Zipf distribution, and the distribution of the location information that these queries request also obeys the Zipf distribution. In this model, the request generates probability P'_i from location L_i , and the request probability P''_i about location L_i is given by $P'_i = P''_i = C/t^\alpha$, where $C = (\sum_{k=1}^{k=100 \times 100} (1/k^\alpha))^{-1}$. The parameter α ($0 \leq \alpha \leq 1$) is the skewness parameter of the Zipf-like distribution, indicating the degree of concentration of requests. The value of α is set to 0.7, 0.8, 0.9, 1.0 in our simulations dataset. All users' requests follow the same Zipf pattern. Although a synthetic dataset cannot represent a real dataset exactly, it allows us to vary data characteristics in a controlled manner not possible to be used in real dataset.

The performance metrics include service coverage ratio, cache hit ratio, and communication overhead. The reason we picked these as performance metrics is that service coverage determines whether a mobile user can be served, cache hit ratio determines whether a mobile user can be served without LBS server, and communication overhead is also what mobile user cares about. For each of these performance metrics, we evaluated the impact of *proxies number* N and *cache size* S . While evaluating the impact of N (resp., S), we fixed $S = 2\%$ (resp., $N = 14\%$ of total locations).

4.2. Service Coverage Ratio. We introduce *coverage ratio* to evaluate the performance of our cache deployment strategy. A query is said to be covered if the mobile user is in the

FIGURE 7: Deployment of cache proxies in Beijing.

FIGURE 8: Deployment of cache proxies in San Francisco.

communication range of a proxy when the query is generated. The coverage ratio is defined as the proportion of the number of queries covered by the cache layer to the total number of queries. If the number of cache proxies approaches $A \times B$, the service coverage ratio could achieve 100%.

In our experiment, we set the proxy number N varying from 5% to 20% of $A \times B$. Figures 9(a) and 9(b) show the impact of the number of cache proxies. Figure 9(a) shows the coverage ratio varying with N with two real datasets. When $N = 5\%$, the coverage ratio is about 85% for Beijing and 95% for San Francisco. With N increasing, the coverage ratio increases quickly. When $N = 20\%$, the coverage ratio approaches 99% for both real datasets. It indicates that if

cache proxies are deployed in a small portion of the grid, they can serve most of the queries. It also suggests Power-Law distribution in the datasets. We can also see that cab traces in San Francisco are more concentrated than mobile user traces in Beijing. Figure 9(b) shows the results with simulated Zipf-distribution dataset and how the coverage ratio looks like with different α . It shows that, with a fixed α (which indicates how concentrated the simulated Zipf-distribution dataset is), with N increasing, the coverage ratio increases quickly. And it also shows that, for a fixed N , when α increases, the coverage ratio increases quickly. Because cache size cannot affect service coverage ratio, we do not illustrate the impact of cache size S with service coverage ratio.

FIGURE 9: Service coverage ratio.

4.3. Cache Hit Ratio. We use *cache hit ratio* as another performance metric of cache deployment strategy. A query is said to be hit if the mobile user can get the desired content from the cache when the query is generated. The cache hit ratio is defined as the proportion of the number of queries hit by the cache layer to the total number of queries.

Figures 10(a) and 10(b) show the impact of cache proxy number. Figures 10(c) and 10(d) show the impact of cache size S .

Cache proxy number is an important factor for the performance of our strategies. The larger cache proxy number yields the higher cache hit ratio and provides the better location privacy guarantee. Figure 10(a) shows with real dataset, when cache size is set to 2%, how the average cache hit ratio of proxies varies when cache proxy number N varies from 5% to 20% of total cache proxy number. The cache hit ratio is low when cache size is small. When N reaches 5%, the cache hit ratio is over 0.90 for Beijing and over 0.50 for San Francisco. It increases slowly when N is further increased. As shown in Figure 9(a), when N reaches 8% for San Francisco, the cover ratio barely increased, which causes the cache hit ratio to increase slowly. The number of destinations for a cab is much larger than that for a mobile user, so the hit ratio for Cabspotting dataset is lower than Geolife. Figure 10(b) shows how it looks like with simulated Zipf-distribution dataset. We can see that, with a fixed α , with N increasing, the hit ratio increases. And it also shows that, for a fixed N , when α increases, the hit ratio increases.

Cache size is an important factor for the performance of our strategies. The larger cache size yields the higher cache hit ratio and provides the better location privacy guarantee. Figure 10(c) shows with real dataset, when cache proxy number N is set to 2% $A \times B$, how the average cache hit ratio varies when cache size S varies from 1% to 3% of the size of keyword set. The cache hit ratio is low when cache

size is small, which is about 50% for $S = 1\%$. The number of destinations for a cab is much larger than that for a mobile user, so the hit ratio for cabspotting dataset is lower than Geolife. We also can see that cache hit ratio increases slightly with cache size up to 2% in Geolife dataset. Figure 10(d) shows how it looks like with simulated Zipf-distribution dataset. We can see that, with a fixed α which indicates how concentrated the simulated Zipf-distribution dataset is, with S increasing, the hit ratio increases. And it also shows that, for a fixed S , when α increases, the hit ratio increases.

4.4. Comparison of Communication Overhead. We assume that the communication cost of transmitting one unit of message from the LBS server to a mobile user is C_s , and the cost from a cache proxy to a user is C_p . Normally C_p is much smaller than C_s ; not only is the cache proxy local and is the LBS server a remote site, but also the proxies can broadcast messages to many users at the same time. We set $C_p/C_s = 0.25$ in our experiments. The communication overhead is calculated by the total number of messages exchanged between the users and the LBS server and the proxies multiplied by their corresponding communication cost.

We use the classical k -anonymity strategy [6] as the baseline to compare the proposed LPPS. Figures 11(a) and 11(c) are made from real dataset, showing the communication overhead of LPPS as the percentage of the classical k -anonymity strategy. Figure 11(a) indicates the impact of N . It can be seen that, with N increasing, the cost decreases. Figure 11(c) indicates the impact of S , which shows that, with cache size S increase, the cost decreases, compared to k -anonymity strategy. It is shown that the communication overhead decreases when the number of proxies and the cache size increase. Figures 11(b) and 11(d) show, with simulated Zipf-distribution dataset, how the $\text{cost}(\text{lpps})/\text{cost}(k\text{-anonymity})$

FIGURE 10: Cache hit ratio.

TABLE 2: Performance comparison.

Comparison	LPPS	The classical K -anonymity strategy
Basic idea	Push-based strategy	Pull-based strategy
Privacy preservation	Guarantee K -anonymity	Guarantee K -anonymity
Service coverage ratio	99% when $N = 20\%$	No proxy, just query data directly
Cache hit ratio	60% to 100% when $N > 5\%$ and $S > 2\%$	No cache, just query data directly
Communication overhead	25% of the classical K -anonymity strategy in the best case	Much higher than LPPS

looks like with different α . It shows that, with a fixed α , with N and S increasing, the cost decreases quickly. And it also shows that, for a fixed N , when α increases, which means the dataset is more concentrated, the cost decreases quickly. In the best

case, the communication overhead of LPPS is only about 25% of the classical k -anonymity strategy. In Table 2 we show the performance comparison between our LPPS and the classical k -anonymity strategy.

FIGURE 11: Comparison of communication overhead: $\text{cost(lpps)}/\text{cost}(k\text{-anonymity})$.

5. Conclusion

In this paper, we address the location privacy issues in the prevailing location-based services and propose a push-based location privacy preservation scheme called LPPS. The LPPS introduces a distributed cache layer to store the popular location-related data and push them to mobile users. Strategies for cache proxy deployment and cache pushing are proposed to achieve k -anonymity of location privacy. Cache replacement and updating strategies are proposed to mine real popular places of interest from a lot of fake feedback indexes when cache miss occurred. Trace driven simulations illustrate that the proposed scheme achieves high service coverage ratio, decent cache hit ratio, and low communication overhead. In the future, we will focus on personalized cache pushing strategy; for example, people from different directions at different time may have different destinations.

We will focus on more kinds of cache content besides the location data. This system can be practically rolled out with VANET (Vehicular Ad hoc Network) and Wi-Fi city plan; our cache proxies here could be road side unit [38] in VANET which can provide more kinds of services.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

This work was partially supported by the National Natural Science Foundation of China (Grant nos. 61373128, 91218302, and 61321491), the Key Project of Jiangsu Research Program Grant (no. BE2013116), the EU FP7 IRSES MobileCloud

Project (Grant no. 612212), the funding from Alexander von Humboldt Foundation, the Collaborative Innovation Center of Novel Software Technology and Industrialization, and the Sino-German Institutes of Social Computing.

References

- [1] <http://www.dianping.com/>.
- [2] Y. Zheng, L. Wang, R. Zhang, X. Xie, and W.-Y. Ma, "GeoLife: Managing and understanding your past life over maps," in *Proceedings of the 9th International Conference on Mobile Data Management (MDM '08)*, pp. 211–212, IEEE, Beijing, China, April 2008.
- [3] "Advertise with promoted tweets twitter for business," <https://business.twitter.com/en/help/overview/what-are-promoted-tweets.html>.
- [4] N. Patel, "The 10 billion rule: location, location, location," <http://www.strategyanalytics.com/default.aspx?mod=reportabstract-viewer&a0=6355>.
- [5] W. Karim, "The privacy implications of personal locators: why you should think twice before voluntarily availing yourself to GPS monitoring," *Washington University Journal of Law & Policy*, vol. 14, pp. 485–515, 2004.
- [6] L. Sweeney, "k-anonymity: a model for protecting privacy," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 5, pp. 557–570, 2002.
- [7] J. Meyerowitz and R. Roy Choudhury, "Hiding stars with fireworks: location privacy through camouflage," in *Proceedings of the 15th Annual ACM International Conference on Mobile Computing and Networking (MobiCom '09)*, pp. 345–356, Beijing, China, September 2009.
- [8] H. Kido, Y. Yanagisawa, and T. Satoh, "Protection of location privacy using dummies for location-based services," in *Proceedings of the 21st International Conference on Data Engineering Workshops (ICDEW '05)*, p. 1248, Tokyo, Japan, April 2005.
- [9] M. Gruteser and X. Liu, "Protecting privacy in continuous location-tracking applications," *IEEE Security & Privacy*, vol. 2, no. 2, pp. 28–34, 2004.
- [10] A. R. Beresford and F. Stajano, "Location privacy in pervasive computing," *IEEE Pervasive Computing*, vol. 2, no. 1, pp. 46–55, 2003.
- [11] M. Gruteser and D. Grunwald, "Anonymous usage of location-based services through spatial and temporal cloaking," in *Proceedings of the 1st International Conference on Mobile systems, Applications and Services (MobiSys '03)*, pp. 31–42, ACM, San Francisco, Calif, USA, May 2003.
- [12] B. Gedik and L. Liu, "Location privacy in mobile systems: a personalized anonymization model," in *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems (ICDCS '05)*, pp. 620–629, IEEE, Columbus, Ohio, USA, June 2005.
- [13] M. F. Mokbel, C.-Y. Chow, and W. G. Aref, "The new Casper: query processing for location services without compromising privacy," in *Proceedings of the 32nd International Conference on Very Large Data Bases (VLDB '06)*, pp. 763–774, 2006.
- [14] R. Cheng, Y. Zhang, E. Bertino, and S. Prabhakar, "Preserving user location privacy in mobile data management infrastructures," in *Privacy Enhancing Technologies: 6th International Workshop, PET 2006, Cambridge, UK, June 28–30, 2006, Revised Selected Papers*, vol. 4258 of *Lecture Notes in Computer Science*, pp. 393–412, Springer, Berlin, Germany, 2006.
- [15] B. Gedik and L. Liu, "Location privacy in mobile systems: a personalized anonymization model," in *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems (ICDCS '05)*, pp. 620–629, June 2005.
- [16] G. Ghinita, P. Kalnis, and S. Skiadopoulos, "Mobi-hide: a mobile peer-to-peer system for anonymous location-based queries," in *Advances in Spatial and Temporal Databases: 10th International Symposium, SSTD 2007, Boston, MA, USA, July 16–18, 2007. Proceedings*, Lecture Notes in Computer Science, pp. 221–238, Springer, Berlin, Germany, 2007.
- [17] C.-Y. Chow, M. F. Mokbel, and X. Liu, "A peer-to-peer spatial cloaking algorithm for anonymous location-based service," in *Proceedings of the 14th Annual ACM International Symposium on Advances in Geographic Information Systems (ACM-GIS '06)*, pp. 171–178, November 2006.
- [18] M. Li, S. Salinas, A. Thapa, and P. Li, "n-CD: a geometric approach to preserving location privacy in location-based services," in *Proceedings of the 32nd IEEE Conference on Computer Communications (IEEE INFOCOM '13)*, pp. 3012–3020, Turin, Italy, April 2013.
- [19] F. Espinoza, P. Persson, A. Sandin, H. Nyström, E. Cacciatore, and M. Bylund, "Geonotes: social and navigational aspects of location-based information systems," in *Proceedings of the International Symposium on Ubiquitous Computing (Ubicomp '01)*, Atlanta, Ga, USA, September–October 2001.
- [20] D. Yang, X. Fang, and G. Xue, "Truthful incentive mechanisms for k-anonymity location privacy," in *Proceedings of the IEEE INFOCOM*, pp. 2994–3002, Turin, Italy, April 2013.
- [21] X. Liu, H. Zhao, M. Pan, H. Yue, X. Li, and Y. Fang, "Traffic-aware multiple mix zone placement for protecting location privacy," in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM '12)*, pp. 972–980, Orlando, Fla, USA, March 2012.
- [22] X. Gong, X. Chen, K. Xing, D.-H. Shin, M. Zhang, and J. Zhang, "Personalized location privacy in mobile networks: a social group utility approach," in *Proceedings of the 34th IEEE Annual Conference on Computer Communications and Networks (INFOCOM '15)*, pp. 1008–1016, May 2015.
- [23] B. Hoh, M. Gruteser, H. Xiong, and A. Alrabady, "Preserving privacy in GPS traces via uncertainty-aware path cloaking," in *Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS '07)*, pp. 161–171, November 2007.
- [24] Z. Zhu and G. Cao, "APPLAUS: a privacy-preserving location proof updating system for location-based services," in *Proceedings of the 30th IEEE International Conference on Computer Communications (IEEE INFOCOM 2011)*, pp. 1889–1897, Shanghai, China, April 2011.
- [25] P. Galdames and Y. Cai, "Efficient processing of location-cloaked queries," in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM '12)*, pp. 2480–2488, IEEE, Orlando, Fla, USA, March 2012.
- [26] Y. Wang, D. Xu, X. He, C. Zhang, F. Li, and B. Xu, "L2P2: Location-aware location privacy protection for location-based services," in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM '12)*, pp. 1996–2004, Orlando, Fla, USA, March 2012.
- [27] K. Vu, R. Zheng, and J. Gao, "Efficient algorithms for K-anonymous location privacy in participatory sensing," in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM '12)*, pp. 2399–2407, Orlando, Fla, USA, March 2012.

- [28] G. Xue, X. Zhao, and L. Li, "Checking in without worries: location privacy in location based social networks," in *Proceedings of the Conference on Computer Communications (INFOCOM '13)*, Turin, Italy, August 2013.
- [29] X.-Y. Li and T. Jung, "Search me if you can: privacy-preserving location query service," in *Proceedings of the 32nd IEEE Conference on Computer Communications (IEEE INFOCOM '13)*, pp. 2760–2768, Turin, Italy, April 2013.
- [30] W. Wei, F. Xu, and Q. Li, "MobiShare: flexible privacy-preserving location sharing in mobile online social networks," in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM '12)*, pp. 2616–2620, IEEE, Orlando, Fla, USA, March 2012.
- [31] B. Doer, M. Fouz, and T. Friedrich, "Why rumors spread so quickly in social networks," *Communications of the ACM*, vol. 55, no. 6, p. 70, 2012.
- [32] M. Zhang, J.-G. Luo, L. Zhao, and S.-Q. Yang, "A peer-to-peer network for live media streaming using a push-pull approach," in *Proceedings of the 13th Annual ACM International Conference on Multimedia (MULTIMEDIA '05)*, pp. 287–290, ACM, Singapore, November 2005.
- [33] R. Unni and R. Harmon, "Perceived effectiveness of push vs. pull mobile location based advertising," *Journal of Interactive Advertising*, vol. 7, no. 2, pp. 28–40, 2007.
- [34] M. Piórkowski, N. Sarafijanovic-Djukic, and M. Grossglauser, "A parsimonious model of mobile partitioned networks with clustering," in *Proceedings of the 1st International Conference on Communication Systems and Networks and Workshops (COM-SNETS '09)*, pp. 1–10, Bangalore, India, January 2009.
- [35] <http://www.nytimes.com/2014/12/11/nyregion/plan-approved-for-wi-fi-at-new-york-citys-pay-phones.html?r=0/>.
- [36] L. Duan, T. Kubo, K. Sugiyama, J. Huang, T. Hasegawa, and J. Walrand, "Incentive mechanisms for smartphone collaboration in data acquisition and distributed computing," in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM '12)*, pp. 1701–1709, Orlando, Fla, USA, March 2012.
- [37] L. Duan, T. Kubo, K. Sugiyama, J. Huang, T. Hasegawa, and J. Walrand, "Motivating smartphone collaboration in data acquisition and distributed computing," *IEEE Transactions on Mobile Computing*, vol. 13, no. 10, pp. 2320–2333, 2014.
- [38] C. Lochert, B. Scheuermann, C. Wewetzer, A. Luebke, and M. Mauve, "Data aggregation and roadside unit placement for a vanet traffic information system," in *Proceedings of the 5th ACM International Workshop on VehiculAr Inter-NETworking (VANET '08)*, pp. 58–65, ACM, San Francisco, Calif, USA, September 2008.
- [39] S. Amini, J. Lindqvist, J. Hong, J. Lin, E. Toch, and N. Sadeh, "Caché: caching location-enhanced content to improve user privacy," in *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services*, pp. 197–210, ACM, 2011.
- [40] Mitzenmacher and Upfal, *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*, Cambridge University Press, Cambridge, UK, 2005.

Hindawi

Submit your manuscripts at
<http://www.hindawi.com>