Protecting Location Privacy of Users Based on Trajectory Obfuscation in Mobile Crowdsensing

Zhigang Gao, Yucai Huang, Leilei Zheng, Huijuan Lu, Bo Wu, *Member, IEEE*, and Jianhui Zhang

***Abstract*—In mobile crowdsensing activities, it is usually necessary for participants to upload sensing data and related locations. The existing location privacy-preserving mechanisms cannot well protect a user's trajectory privacy because attackers can mine the user's trajectory features through data analysis techniques. Aiming at the trajectory privacy protection problem, this article proposes a differential location privacy-preserving mechanism based on trajectory obfuscation (LPMT). LPMT first extracts the stay points as the features of a trajectory based on the sliding window algorithm, and then obfuscates each stay point to a target obfuscation subregion through the exponential mechanism, and finally performs the Laplace sampling in the target obfuscation subregion to obtain the obfuscated GPS points. Compared with the baseline mechanisms, LPMT can reduce data quality loss by more than 20% while providing the same level of obfuscation quality, which indicates that LPMT has the advantages of strong security and high quality of service.**

***Index Terms*—Differential location privacy, exponential mechanism, mobile crowdsensing, trajectory obfuscation.**

I. Introduction

MOBILE crowdsensing (MCS) is a kind of emerging distributed intelligence technology. MCS brings a new data collection paradigm which uses mobile devices (e.g., smartphones, vehicular sensors, etc.) widely distributed in the surrounding environment to execute sensing and calculation, such as temperature and humidity monitoring, road condition monitoring, etc. [1].

Although MCS brought great chances for enterprises and organizations, it is often involved in privacy concerns. In most MCS activities, whether in the task assignment phase or the task execution phase, it is necessary to collect the users' location data. If a user's location data are obtained by malicious attackers, they can use clustering and other methods to extract the user's trajectory features, such as stay points or points of interest, which may cause immeasurable risk to the safety of life and property of the user.

In the past decade, researchers have proposed many Location Privacy-Preserving Mechanisms (LPPMs) [2]–[4]. However, the traditional LPPMs focus on discrete location-based systems, such as navigation, online ordering, etc., which only use location data in discrete periods. Because the location data in above-mentioned scenarios are quite sparse, LPPMs only need to consider the spatial dimension, and take measures to obfuscate or anonymize the actual location during the service period to protect location privacy. However, in the MCS scenario, it usually requires participants to turn on the device sensors for a long time to perform sensing tasks and continuously upload sensing data to the data center. Compared with the discrete location-based systems, there are denser user location data in the MCS scenario. If the traditional LPPMs are applied to this situation, an attacker can extract the user's features through data analysis methods such as clustering and regression analysis, and then uses background knowledge to obtain the user's private information. For example, Boukoros *et al.* [5] used the DBSCAN clustering algorithm to extract the points of interest of users, and then combined the information posted by these users on their social platforms such as Twitter and LinkedIn to successfully infer the workplaces of 35% users.

In order to solve the location privacy problem in the MCS scenario, this article proposes a Location Privacy-preserving Mechanism based on Trajectory obfuscation (LPMT). Because there is no trusted server in LPMT, the main operation of location obfuscation is performed by the user devices, and then the user devices upload the obfuscated data to the data center. LPMT consists of three phases, i.e., the trajectory feature extraction phase, the stay points obfuscation phase, and the stay points generalization phase. In the trajectory feature extraction phase, LPMT takes the stay points as the feature of trajectories, and then

Manuscript received September 30, 2021; revised December 21, 2021; accepted January 12, 2022. Date of publication January 27, 2022; date of current version June 13, 2022. This work was supported in part by the National Key R&D Program of China under Grant 2021YFC3320301, in part by the National Natural Science Foundation of China under Grant 61877015, in part by the Zhejiang Provincial Natural Science Foundation under Grant LY21F020028, and in part by the Key Laboratory of Brain Machine Collaborative Intelligence of Zhejiang Province under Grant 2020E10010. Paper no. TII-21-4291. *(Corresponding author: Huijuan Lu.)*

Zhigang Gao, Yucai Huang, Leilei Zheng, and Jianhui Zhang are with the College of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou 310018, China, and also with the Key Laboratory of Brain Machine Collaborative Intelligence of Zhejiang Province, Hangzhou 310018, China (e-mail: gaozhigang@hdu.edu.cn; ychuang97@hdu.edu.cn; zll_zhengleilei@163.com; jh_zhang@hdu.edu.cn).

Huijuan Lu is with the Key Laboratory of Electromagnetic Wave Information Technology and Metrology of Zhejiang Province, Hangzhou 310018, China, and also with the College of Information Engineering, China Jiliang University, Hangzhou 310018, China (e-mail: hjlu@cjlu.edu.cn).

Bo Wu is with the School of Computer Science, Tokyo University of Technology, Hachioji 192-0982, Japan (e-mail: wubo@stf.teu.ac.jp).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TII.2022.3146281.

Digital Object Identifier 10.1109/TII.2022.3146281

generates an obfuscation region r for each stay point. In the stay points obfuscation phase, LPMT outputs a target obfuscation subregion from r through exponential mechanism. In the stay points generation phase, LPMT performs the Laplace sampling in the target obfuscation subregion, and replaces the original GPS points with the sampled GPS points, which will be uploaded to the data center. The main contributions of this article are as follows.

1) This article proposes a location privacy-preserving mechanism based on trajectory obfuscation, which can effectively resist background knowledge attacks in the MCS scenario by ensuring the differential privacy of stay points.
2) This article proposes a trajectory feature extraction method based on sliding window, which can quickly extract the stay points from trajectories.
3) This article proposes a Stay Points Obfuscation Algorithm (SPOA). SPOA maps the original stay points to the target obfuscation subregions by the exponential mechanism, and can provide ε-*differential privacy*.
4) This article proposes a utility calculation method that combines the Euclidean distance and Location Context Similarity (LCS), which can reduce the loss of the quality of the sensing data and provide strong location privacy.

The rest of this article is organized as follows. Section II reviews the related work in the field of location privacy protection. Section III introduces the related definitions of differential privacy. Section IV presents the framework and workflow of LPMT. Section V describes the implementation of LPMT in detail. Section VI explains the experimental process and analyzes the results. Finally, Section VII concludes this article.

II. Related Work

In the current research on location-based systems, there are two general LPPMs, i.e., the anonymity-based LPPMs and the obfuscation-based LPPMs.

The idea of the anonymity-based LPPMs is to cut the connection between user identities and their location information. The k-anonymity scheme and the mix-zone scheme are two common implementations of the anonymity-based LPPMs. Gruteser and Grunwald [6] proposed a spatial cloaking scheme based on k-anonymity [7], which generalized a user's location to an anonymous region, which contained at least other $k-1$ users. All users in the region were anonymous with each other, therefore, the probability of a malicious attacker successfully distinguishing the actual location of a user was less than $1/k$. A user's location will be generalized into a vague location region in the traditional anonymity-based LPPMs. However, in the MCS scenario, generalizing a user's location into a vague location region will reduce the efficiency of task allocation. Zhang *et al.* [8] proposed a virtual selection scheme considering the geographical semantic information characteristics of locations. This scheme used a multicenter clustering algorithm based on the max–min distance method to generate a virtual candidate set for spatial cloaking, which could balance the contradiction between privacy protection and task allocation efficiency. Zhao *et al.* [3] proposed a credibility-based k-anonymity scheme, which could effectively prevent location injection attacks while ensuring quality of service. Different from the k-anonymity scheme, the mix-zone scheme can be used without user identity information. Beresford and Stajano [9] first proposed the mix-zone scheme. The mix-zone scheme protected the privacy of a user by frequently changing a user's name or pseudonym in the mixed zones. Guo *et al.* [10] proposed an independent mix-zone scheme (IndMZ) that combined the collaborative mix zone with the self-established mix zone. IndMZ could ensure k-anonymity, and the average cost of extended beacon message was $k/2$. Memon *et al.* [11] proposed a scheme based on multiple mix zones, which linked the starting point of queries to a nearby point of interest, and ensured that the starting point of queries and the point of interest were indistinguishable. However, the current anonymity-based LPPMs still have many disadvantages, such as the vulnerability to resist background knowledge attacks.

The obfuscation-based LPPMs protect the location privacy by obfuscating the actual locations of users with the fake ones. The fake location scheme and the location mapping scheme are the two common implementations of this mechanism. The fake location scheme protects users' locations by mixing multiple fake locations with their actual ones. You *et al.* [12] used random and rotating patterns to generate virtual trajectories of users. Lei *et al.* [13] rotated the trajectories of users beyond a certain distance deviation to achieve the indistinguishability between the actual trajectories and the fake ones. The location mapping scheme maps actual locations to other ones through some mapping rules, and then sends the obfuscated locations to the data center. Andres *et al.* [14] proposed the concept of Geo-Indistinguishability (GeoInd) based on the differential privacy mechanism. They first used the Laplace sampling to discretize a 2-D map, and then added noise to the actual locations. Finally, they restricted the obfuscation region through a truncation mechanism. It has proved that the differential privacy mechanism could resist background knowledge attacks. However, GeoInd did not consider the loss of data quality, so it was not suitable for the MCS applications. Wang *et al.* [15] proposed a differential privacy obfuscation scheme based on grid domain decomposition for the sparse MCS applications. They used a data adjustment function and an uncertainty-aware inference algorithm to reduce the loss of data quality and improve the practicality of this scheme.

III. Foundation of Differential Privacy

Differential privacy is a general privacy protection strategy proposed by Dwork [16]. It has proved by the rigorous mathematical theories that differential privacy strategy can resist background knowledge attacks [14]. In order to make it easy to understand the rest parts of this article, this section will briefly introduce the definitions and theorems related to differential privacy.

Definition 1. ε-Differential Privacy: Suppose there are two datasets D and D' that are different from each other at most one row record. For a randomization algorithm M, if the outputs on the datasets D and D' meet the inequality (1), then the algorithm $M(D)$ can provide ε-differential privacy

$$Pr(M(D) \in S) \leq \exp(\varepsilon) \times Pr(M(D') \in S) \tag{1}$$

where $Pr(M(D) \in S)$ denotes the probability that the output of M on the dataset D belongs to S, the parameter ε is called the

privacy budget or privacy level. In general, smaller ε will lead to higher privacy protection level. However, data quality will decrease as the increment of privacy protection level.

Definition 2. Sensitivity: Suppose there is a function $f: D^n \rightarrow R^d$, its input is an n-dimension dataset, and its output is a d-dimension vector. For any datasets D and D' that are different from each other at most one row record, the sensitivity of the function f is defined as

$$\Delta f = \max_{D,D'} \|f(D) - f(D')\| \tag{2}$$

where Δf is the sensitivity, which denotes the greatest impact caused by deleting/inserting one record from/into the dataset D.

There are two main implementation mechanisms of the differential privacy, i.e., the Laplace mechanism and the exponential mechanism. The Laplace mechanism and the exponential mechanism are usually used for the protection of continuous private data and discrete private data respectively.

The Laplace mechanism achieves the obfuscation effect by adding the Laplace noise to the original private data. For example, D can be obfuscated by

$$M(D) = f(D) + Y \tag{3}$$

where

$$Y \sim L\left(0, \frac{\Delta f}{\varepsilon}\right) \tag{4}$$

and $f(D)$ denotes the function which queries private data, Y is the Laplace noise which obeys the distribute of $L(0, \frac{\Delta f}{\varepsilon})$, and $M(D)$ is the output of the algorithm M. It can be seen that when the privacy budget ε becomes smaller, the noise value will becomes larger, and the level of privacy protection will becomes higher.

The exponential mechanism [17] maps the original private data to a candidate set, and then uses a utility function to calculate the scores for all the mappings, and finally outputs a sampled value with a given probability. In fact, the exponential mechanism ensures differential privacy by approximating the actual value of data with the help of utility function. The formal description of the exponential mechanism is defined by Theorem 1.

Theorem 1: Suppose the input of the randomization algorithm M is the dataset D, and the output is an entity object $r \in S$. $u(D, r) \rightarrow R$ is the utility function, and Δu is the sensitivity of the function u. If M chooses r with the probability which is in proportion to $\exp(\varepsilon \frac{u(D,r)}{2\Delta u})$ from S and outputs r, it can provide ε-differential privacy.

When implementing a differential privacy mechanism, there are two main privacy budget compositions, i.e., the sequential composition and the parallel composition. They are defined in Definition 3 and Definition 4, respectively.

Definition 3. Sequential Composition: Suppose the algorithm M consists of m independent steps, i.e., $M = \{M_1, \ldots, M_m\}$. If any step M_i can provide ε_i-differential privacy, the algorithm M can provide $(\sum_{i=1}^{m} \varepsilon_i)$-differential privacy.

Definition 4. Parallel Composition: Suppose the algorithm M consists of m independent steps, i.e., $M = \{M_1, \ldots, M_m\}$. If any step M_i can provide ε_i-differential privacy, the algorithm M can provide $\max\{\varepsilon_1, \ldots, \varepsilon_m\}$-differential privacy.

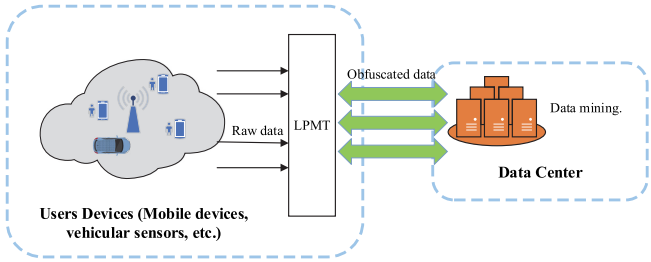

Fig. 1. Location privacy-preserving framework.

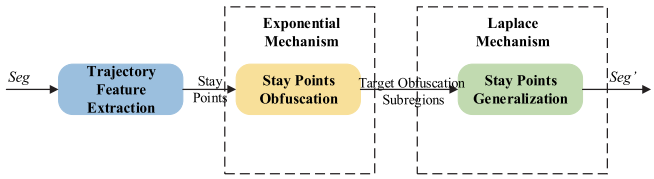

Fig. 2. Workflow of LPMT.

IV. Framework and Workflow of LPMT

Generally, the current location privacy-preserving frameworks can be classified into two categories according to whether they include a trusted server. If a framework includes a trusted server, the user's location will first be uploaded to the trusted server, and then the trusted server performs anonymization or obfuscation operations and delivers the processed data to the data center [18], [19]. However, the communication channel from the user devices to the trusted server is unreliable and vulnerable to man-in-the-middle attacks [20]. In addition, deploying a trusted server is very difficult in reality [21]. If a framework does not contain a trusted server, user devices are responsible for obfuscating location data, and then upload the results to the data center [15], as shown in Fig. 1. Compared with the first type of framework, the non-trusted server framework not only is more practical and safer, but also has higher requirements for the computing performance of user devices, which is suitable for the distributed intelligence scenario.

In Fig. 1, LPMT is based on the non-trusted server framework, the user devices can be mobile phones and vehicular sensors. User devices will execute the obfuscation of the sensing data, and then upload the results to the data center. As shown in Fig. 2, LPMT includes three phases, the first phase is trajectory feature extraction, the second phase is stay points obfuscation, and the third phase is stay points generalization. In these three phases, the trajectory feature extraction phase and stay points generalization phase are deployed on the user devices. In the stay points obfuscation phase, the calculation of the location context similarity is deployed on the data center, and the rest operations are deployed on the user devices.

In the trajectory feature extraction phase, LPMT extracts the stay points as the feature of the user's trajectory and then generates obfuscation regions and subregions. A stay point is a spatiotemporal point that may leak the user's privacy information.

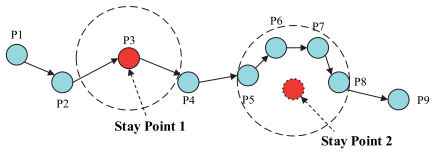

Fig. 3. Two types of stay points.

Definition 5. Stay Point: A characteristic point of trajectories where a user stays on a specific location for a timespan greater than or equal to a time threshold.

Compared with the original GPS points, almost every stay point has a specific meaning, such as the user's workplaces, home, dining room, etc. Two types of stay points are summarized in the literature [22], as shown in Fig. 3. The stay point 1 indicates that the user stays at this place for a long period, and the stay point 2 indicates that the user is wandering around the place for a while.

At the end of the trajectory feature extraction phase, LPMT will generate an obfuscation region r for each original stay point, and then divide r into several subregions at the same size in order to obtain the candidate obfuscation subregions. The reason for generating obfuscation regions and subregions is as follows. If the stay points of the trajectory belong to the type of the stay point 1 in Fig. 3, the sampled points will be evenly distributed around the stay point, an attacker can obtain the feature of the trajectory by clustering on the sampled points. For example, GeoInd in [14] takes an original GPS point as the center of a circle, performs the Laplace sampling in the circle with a radius l, and then replaces the original GPS points with the sampled points. The method in [14] will expose the stay point 1 when it is used to the continuous sampling trajectories. Therefore, LPMT generates obfuscation regions and subregions in order to prepare for the next phase (i.e., stay points obfuscation phase).

In the stay points obfuscation phase, LPMT selects obfuscation subregions from r through an exponential mechanism and makes them meet the requirements of differential privacy (see Theorem 1). There are two reasons for applying the exponential mechanism to the second phase. First, both the input and output of SPOA algorithm are discrete data because the exponential mechanism is usually used for the protection of discrete private data, the exponential mechanism is suitable for the second phase. Second, in the exponential mechanism, the utility function can help us approximate the actual value of the sensing data as much as possible while ensuring differential privacy. In the process of implementation, for a stay point sp, LPMT first calculates the utility value when sp is obfuscated to each subregion according to the utility function, and then calculates the sampling probability according to the utility value. Finally, LPMT samples and outputs the target obfuscation subregion. Additionally, LPMT optimizes the utility function by combining LCS with the Euclidean distance, which can provide a better balance between obfuscation quality and data quality.

In the stay points generalization phase, LPMT uses the Laplace distribution to generalize the obfuscated stay point in order to ensure that the sampling process also meets the differential privacy constraints. As shown in Fig. 3, both stay point 1 and stay point 2 are converged from multiple GPS points.

TABLE I
TABLE OF ABBREVIATIONS

ABBREVIATIONS	FULL NAMES
LCS	Location Context Similarity
LPMT	Location Privacy-preserving Mechanism based on Trajectory obfuscation
LPPMs	Location Privacy-Preserving Mechanisms
MCS	Mobile Crowdsensing
RMSE	Root Mean Square Error
SPOA	Stay Points Obfuscation Algorithm
TFEA	Trajectory Feature Extraction Algorithm

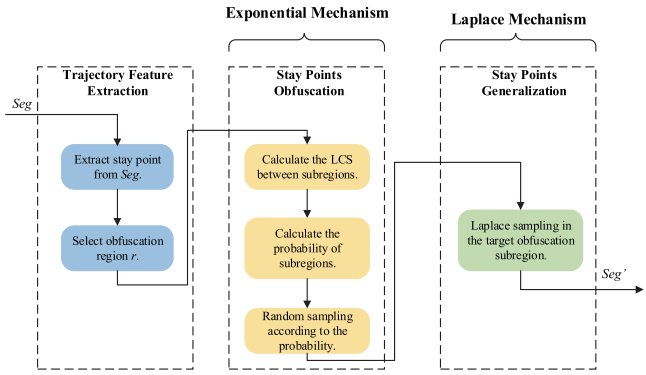

Fig. 4. Implementation of LPMT.

Therefore, after obtaining the target obfuscation subregion r of sp, we need to perform generalization sampling in r, and then replace the original GPS points with the sampled GPS points.

As shown in Fig. 2, LPMT is composed of the exponential mechanism and the Laplace mechanism in its implementation process, each of which meets ε-*differential privacy*. According to Definition 3, LPMT can provide $2 \times \varepsilon$-*differential privacy*. Abbreviations frequently used in this article and their full names are summarized in Table I.

V. IMPLEMENTATION OF LPMT

In this section, we will elaborate on the implementation of LPMT, including the implementation of the trajectory feature extraction algorithm and the stay points obfuscation algorithm, as well as the detailed process of the stay points generalization. The steps of each phase are shown in Fig. 4.

A. Trajectory Feature Extraction Algorithm

In this section, we first introduce some definitions that will be used in the rest of this article, the notations and their descriptions are summarized in Table II.

In MCS applications, non-real-time tasks account for a large proportion [1]. For the non-real-time tasks, the data center does not require the user devices to submit sensing data immediately, and the user devices can buffer the sensing data in its built-in database (e.g., SQLite in Android OS) for a specific timespan. This article defines the user's trajectory buffered during this timespan as a segment *Seg*. If the user devices upload the sensing data to the data center every hour, the local GPS log of the user devices records the user's trajectories during one hour.

TABLE II
NOTATIONS AND DESCRIPTIONS

NOTATIONS	DESCRIPTIONS
Seg	The buffered trajectory segment in user devices.
sp_i	The *i*-th stay point.
p_i	The *i*-th GPS point in *Seg*.
D_{th}	The distance threshold for stay points extraction.
T_{th}	The time threshold for stay points extraction.
r_i	The obfuscation region of sp_i.
$r_{i,j}$	The *j*-th obfuscation subregion in r_i.
$r^c_{i,j}$	The central GPS point of obfuscation subregion $r_{i,j}$.

Because the types of sensing data are different for different MCS scenarios, the buffered timespan can be set according to specific requirements (such as data sampling frequency, real-time requirements, etc.). In this article, we assume that the user devices upload the sensing data to the data center every hour, the local GPS log of the user devices records the user's trajectories during these hours. We assume that a user trajectory has m segments (m>0) in total, the user's trajectory *Tr* is the set of $\{Seg_1, \ldots, Seg_m\}$. In the following sections, LPMT only performs obfuscation on a single segment each time. For simplicity, we use *Seg* to denote the segment, rather than Seg_i. *Seg* consists of a sequence of GPS points $\{p_1, \ldots, p_z\}$. We can extract n stay points from this sequence to denote the segment, and obtain $Seg = \{sp_1, \ldots, sp_n\}$. For a stay point sp_i, it is represented by $(Lat_i, Lngt_i, ArvT_i, LevT_i)$, i.e., Latitude, Longitude, Arrive Time, and Leave Time respectively. The trajectory feature extraction phase includes two steps, i.e., the stay points extraction and the target obfuscation subregion allocation.

In the first step, LPMT extracts the stay points from the trajectory segment *Seg*. The extraction of stay points depends on two scale parameters, i.e., the time threshold T_{th} and the distance threshold D_{th}. For any continuous subsequence of *Seg* denoted by the original GPS points $\{p_x, \ldots, p_y\}$, $1 \le x < y \le z$. If the original GPS points subsequence meets (5) and (6) defined as follows:

$$dist(p_x, p_j) \le D_{\text{th}}, x \le j \le y \tag{5}$$

$$|p_x.T - p_y.T| \ge T_{\text{th}} \tag{6}$$

where $dist(p_x, p_j)$ is the Euclidean distance between p_x and p_j, $p_x.T$ and $p_y.T$ are the collecting time of p_x and p_x, respectively, we can extract a stay point according to (7) defined by

$$\begin{cases} sp.Lat = \sum_{j=x}^{y} p_j.Lat/\,|y-x| \\ sp.Lngt = \sum_{j=x}^{y} p_j.Lngt/\,|y-x| \\ sp.arvT = p_x.T \\ sp.levT = p_y.T \end{cases}. \tag{7}$$

The implementation of the Trajectory Feature Extraction Algorithm (TFEA) is shown in Algorithm 1.

TFEA takes the locally buffered GPS point sequence as input, and initializes the time threshold T_{th} and the distance threshold D_{th} in Line 1. TFEA initializes the left boundary of the sliding window in Line 2. TFEA expands the right boundary of the sliding window from Line 4 to Line 7. In each sliding step,

Algorithm 1: Trajectory Feature Extraction Algorithm.

Input: GPS points sequence $\{p_1, \ldots, p_z\}$.
Output: Stay points set *S*.
1. **Initialize:** Init T_{th}, D_{th};
2. w_left = 1;
3. **while** w_left < = z
4. w_right = w_left + 1;
5. **while** w_right < = z **and** $dist(p_{w\_left}, p_{w\_right}) <= D_{th}$
6. w_right++;
7. **end while**
8. w_right−−;
9. **if** $|p_{w\_left}.T - p_{w\_right}.T| >= T_{th}$
10. $sp.Lat = \sum_{i=w\_left}^{w\_right} p_i.Lat/|y-x|$;
11. $sp.Lngt = \sum_{i=w\_left}^{w\_right} p_i.Lngt/|y-x|$;
12. $sp.arvT = p_{w\_left}.T$;
13. $sp.levT = p_{w\_right}.T$;
14. *S*.add(*sp*);
15. w_left = w_right+1;
16. **else**
17. w_left++;
18. **end if**
19. **end while**
20. **return** *S*

TFEA judges whether the GPS point on the right edge of the window meets (5). If it is true, TFEA algorithm continues to expand the right boundary of the sliding window; otherwise, the expansion process ends. TFEA judges whether the left and right boundaries of the current window meet (6) in Line 9. If it is true, TFEA extracts a stay point according to (7) and adds it into the output set *S* from Line 10 to Line 14, and then replaces the left boundary of the current window with the next point to its right boundary in Line 15; otherwise, TFEA updates the left boundary of the current window in Line 17. The steps from Line 4 to Line 18 are repeated until the window reaches the last element of the input GPS sequence. TFEA finally outputs the set of stay points *S*.

In the second step, LPMT allocates an obfuscation region for each stay point extracted in the first step. There are usually two kinds of obfuscation regions. The first one is a circular region with its original GPS point as the center [23], and the second one is a rectangular region which consists of several equal-sized grids [15]. This article uses the second kind of obfuscation region, as shown in Fig. 5.

The blue points represent the original GPS points and the red points represent the stay points. LPMT takes each stay point sp_i as the center, and draws a square region as the obfuscation region r_i, and then divides r_i into several equal-sized grids as the candidate obfuscation subregions $\{r_{i,1}, \ldots, r_{i,k}, \ldots, r_{i,w}\}$. Because the size of r_i and the number of the subregions will influence the obfuscation effect, we can set the size of r_i according to the Euclidean distance of a user's movement in *Seg*, and set the number of subregions in r_i according to the number of stay points in *Seg*.

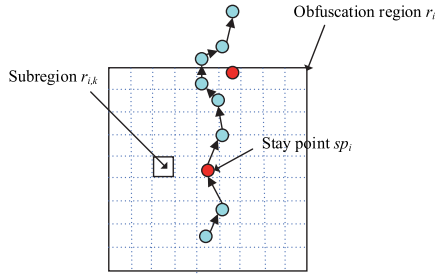

Fig. 5. Obfuscation region of sp_i.

Algorithm 2: Stay Points Obfuscation Algorithm.

Input: Trajectory segment $Seg = \{sp_i, \ldots, sp_n\}$.
Output: Target obfuscation subregions set S.
1. Download location context similarity information from data center;
2. **for each** stay point sp_i **in** Seg **do**
3. **for each** subregion $r_{i,k}$ **in** region r_i **do**
4. Calculate utility value: $U(sp_i, r_{i,k}) = \beta \times lcs(r_{i,sp}, r_{i,k}) - (1-\beta) \times norm(dist(sp_i, r^c{}_{i,k}))$;
5. Calculate sensitivity: $\Delta U = \max_{r_{i,k} \in M(Seg), r'_{i,k} \in M(Seg')} \|U(sp_i, r_{i,k}) - U(sp_i, r'_{i,k})\|$;
6. Calculate sampling probability of $r_{i,k}$: $Pr(r_{i,k}) = \exp(\frac{\varepsilon U(sp_i, r_{i,k})}{2\Delta U}) \Big/ \sum_j \exp(\frac{\varepsilon U(sp_i, r_{i,j})}{2\Delta U})$;
7. **end for**
8. Random sampling according to the probability, output a target subregion $r_{i,t}$ to set S;
9. **end for**
10. **return** S;

B. Stay Points Obfuscation Algorithm

LPMT obfuscates each stay point in *Seg* in turn by using SPOA. SPOA takes the stay points set $\{sp_1, \ldots, sp_n\}$ as input data and outputs the target obfuscation subregions of stay points, as shown in Algorithm 2.

SPOA downloads the LCS data of candidate obfuscation subregions in Line 1; Line 2 enumerates each stay point in *Seg*; Line 3 enumerates each subregion $r_{i,k}$ in r_i; Lines 4 to 6 calculate the probability that $r_{i,k}$ is selected as the target obfuscation subregion; SPOA performs random sampling in Line 8 according to the probability calculated before, and adds the output subregion to S in Line 10. In the rest of this section, we will elaborate on the calculation steps of the utility value, sensitivity, and sampling probability in Algorithm 2.

Utility value calculation. In Section V-A, we have obtained the candidate obfuscation subregions $r_i = \{r_{i,1}, \ldots, r_{i,k}, \ldots, r_{i,w}\}$ of sp_i. In this step, the user devices will query the LCS between two subregions in r_i from the data center. The LCS value between $r_{i,sp}$ and $r_{i,k}$ is calculated based on the historical sensing data, which have been uploaded to the data center ($r_{i,sp}$ is the subregion where the stay point sp_i lies.), and it is defined by

$$LCS(r_{i,sp}, r_{i,k}) = \sum_{j=1}^{24} \overline{v}_{sp,j} \times \overline{v}_{k,j} \Big/ \left\{ \sqrt{\sum_{j=1}^{24} (\overline{v}_{sp,j})^2} \times \sqrt{\sum_{j=1}^{24} (\overline{v}_{k,j})^2} \right\} \tag{8}$$

where $\overline{v}_{k,j}$ is the average value of subregion $r_{i,k}$ in the jth period. LPMT divides one day into 24 periods with equal intervals, and calculates the average value $\{\overline{v}_{i,1}, \ldots, \overline{v}_{i,24}\}$ of the sensing data in each period. For two subregions, if the mean values of their sensing data in the same period are closer, the two subregions have the higher similarity of the location context.

LPMT calculates the utility value by combining the Euclidean distance with LCS, and then obtains the sampling probability of every subregion according to the utility value. The Euclidean distance between the stay point sp_i and the center $r^c{}_{i,k}$ of the obfuscation subregion $r_{i,k}$ can be used to measure the quality of trajectory obfuscation. The obfuscation quality from sp_i to $r_{i,k}$ can be calculated by

$$Q(sp_i, r_{i,k}) = dist(sp_i, r^c_{i,k}). \tag{9}$$

For an obfuscation subregion $r_{i,k}$, the farther the distance between $r^c{}_{i,k}$ and sp_i is, the higher the quality of trajectory obfuscation is. However, the quality of the sensing data will decrease as the quality of the trajectory obfuscation increases. Therefore, LPMT applies LCS to the utility calculation. The data quality loss can be reduced if the stay point is obfuscated to a subregion with the similar LCS value. The utility function U can be derived by

$$U(sp_i, r_{i,k}) = \beta \times \mathrm{LCS}(r_{i,sp}, r_{i,k}) - (1-\beta) \times norm(dist(sp_i, r^c_{i,k})) \tag{10}$$

where $r_{i,sp}$ is the subregion where sp_i is located, $norm(x)$ denotes the normalization function of the output value x, the parameter β is used to control the weight of the Euclidean distance and LCS in the utility function.

Sensitivity calculation. The sensitivity (see Definition 2) of the SPOA algorithm can be derived according to

$$\Delta U = \max_{\substack{r_{i,k} \in M(Seg) \\ r'_{i,k} \in M(Seg')}} \left\| U(sp_i, r_{i,k}) - U(sp_i, r'_{i,k}) \right\| \tag{11}$$

where M refers to the SPOA algorithm, Seg' represents a new trajectory segment formed by adding or deleting a stay point in *Seg*.

Sampling probability calculation. According to the defination of exponential mechanism, the probability that a subregion $r_{i,k}$ is selected as the target obfuscation subregion can be calculated by

$$Pr(r_{i,k}) = \exp\left(\frac{\varepsilon U(sp_i, r_{i,k})}{2\Delta U}\right) \Big/ \sum_j \exp\left(\frac{\varepsilon U(sp_i, r_{i,j})}{2\Delta U}\right). \tag{12}$$

At the end of the stay points obfuscation phase, SPOA performs uniform sampling according to the probability value calculated above, and outputs a target obfuscation subregion $r_{i,t}$.

C. Stay Points Generalization

In the stay points generalization phase, LPMT performs the Laplace sampling in the target obfuscation subregion $r_{i,t}$, and uses the sampled GPS points as the final output of LPMT and uploads it to the data center. LPMT uses the scheme proposed in [14] to perform GPS point sampling, and this scheme belongs to the Laplace mechanism, which includes the following three steps.

Step 1. Polar coordinate transformation. In this step, LPMT converts the probability density function of the 3-D Laplace distribution into the polar coordinate form defined by

$$\begin{cases} D_{\varepsilon,R}(l) = \varepsilon^2 \cdot l \cdot e^{-\varepsilon l} \\ D_{\varepsilon,\Theta}(\theta) = \frac{1}{2\pi} \end{cases} \tag{13}$$

where the random variables l and θ represent the radius and angle in the polar coordinate system, respectively, and $D_{\varepsilon,R}(l)$ and $D_{\varepsilon,\Theta}(\theta)$ represent the probability density functions of the random variables l and θ , respectively.

Step 2. Uniform sampling for the random variables θ and l. In this article, l represents the distance between the sampled point and the center of target obfuscation subregion $r^c{}_{i,t}$. l reflects the size of the distribution range of the sampled points, and the value of l is proportional to the amount of privacy budget consumed. Therefore, given a privacy budget ε, the probability that the distance between the sampled points and $r^c{}_{i,t}$ falls between [0, l) can be calculated by

$$Pr(l) = \int_0^l D_{\varepsilon,R}(\rho)d\rho = 1 - (1+\varepsilon l)e^{-\varepsilon l} \tag{14}$$

and

$$l = -\frac{1}{\varepsilon}\left(W_{-1}\left(\frac{Pr-1}{e}\right)+1\right) \tag{15}$$

where (14) is the cumulative distribution function of radius l, and W_{-1} is the Lambert W function (the -1 branch).

In order to sample the random variable l, LPMT first randomly generates a probability value Pr within the interval [0, 1), and then calculates the value of l according to (15). In addition, θ only needs to be sampled within the interval $[0, 2\pi)$. Finally, LPMT calculates the coordinates of the sampled point z by

$$z = r^c_{i,t} + \langle l \cdot \cos\theta, l \cdot \sin\theta \rangle . \tag{16}$$

After several sampling operations, a large number of GPS points are generated around $r^c{}_{i,t}$, as shown in Fig. 6(a).

Step 3. Truncation operation. LPMT performs a truncation operation to limit the sampled points within the target obfuscation subregion $r_{i,t}$. Since the Laplace sampling range is infinite in theory, a truncation operation must be performed to ensure that the obtained GPS points fall within $r_{i,t}$. The results after the sampled points in Fig. 6(a) are performed a truncation operation are shown in Fig. 6(b).

VI. Experiment and Analyses

In this section, we will evaluate the performance of LPMT in terms of space and time overhead, obfuscation quality and data quality. Obfuscation quality refers to the deviation between the new trajectory and the original one after stay points obfuscation.

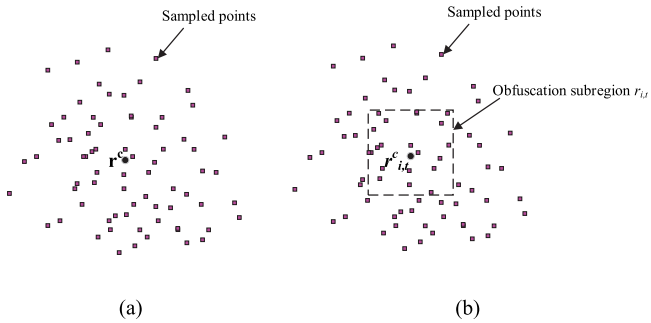

Fig. 6. Sampling and truncation. (a) Sampled points. (b) Results of a truncation operation.

The higher the obfuscation quality is, the higher the protection level of location privacy is. Data quality refers to the errors of the sensing data after stay points obfuscation. LPPMs must minimize the errors as much as possible to ensure that the data center can mine useful information from the sensing data.

A. Experiment Setup

Experiment environment. In the following experiments, the software development platform is PyCharm, and all the algorithms are implemented in Python. The hardware platform is a PC equipped with the Core(TM) i7-8700 CPU of 3.20 GHz, the memory of 16 GB, and the operating system of Win10 x64.

Dataset. The dataset used in the following experiments is Geolife [24], which collects the GPS data of 182 users in Beijing over 5 years. The dataset contains a total of 17 621 trajectories, and they extensively record the users' outdoor movement trajectories, including events such as going to work, going home, entertainment, and traveling. Since 76% users in Geolife have a data collection time of more than one week, we divided the dataset into the following two parts based on the data collection time. 1) The data in the first week are used as historical data to calculate LCS; 2) The rest data are used to test the performance of LPMT. In order to study the data quality loss of LPMT in the MCS scenario, we use the "*Altitude*" field in Geolife as the sensing data of the MCS task because it can reflect the geographic location context similarity of different locations.

Baseline mechanisms. We chose GeoInd [14] and DUM$-\varepsilon$e [15] as the baseline mechanisms, and compared their performance with LPMT in terms of obfuscation quality and data quality. GeoInd uses the Laplace mechanism to add noise to original GPS points to achieve obfuscation; DUM$-\varepsilon$e uses a well-trained obfuscation matrix to obfuscate the location of participants in a grid unit, and then uses data adjustment function and uncertainty-aware inference algorithm to reduce the loss of data quality. In addition, this article also conducted a self-control experiment (LPMT without LCS versus LPMT) to compare the obfuscation quality and data quality of LPMT between before and after integrating LCS.

Experiment parameters. There are four parameters to be set in the following experiments, i.e., the time threshold T_{th}, the distance threshold D_{th}, the privacy budget ε and the utility weight β. ε and β have multiple values for selecting. We compare the obfuscation quality and data quality of baselines under

TABLE III
PARAMETER SETUP

PARAMETERS	VALUES
T_{th}	5 mins
D_{th}	100 m
ε	0.05, 0.1, 0.5, ln2, 1, 1.5, ln6, ln8
β	0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9

different composition of privacy budgets and utility weights. Table III shows the values of parameters used in the following experiments.

B. Performance Indexes

The obfuscation quality is an important index to evaluate the performance of LPPMs. If the distance between the obfuscated stay point and the original stay point becomes larger, the quality of data obfuscation will become higher. Therefore, this article uses the distance between the original stay point and the obfuscated one to quantify the obfuscation quality. The obfuscation quality for *Seg* is defined by

$$\overline{Q}(sp_i, sp_i') = \frac{1}{n}\sum_{i=1}^{n} dist(sp_i, sp_i') \tag{17}$$

where n is the number of original stay points in *Seg*, sp_i represents the original stay point, and sp_i' represents the obfuscated stay point corresponding to sp_i.

In the MCS scenario, the data quality of the sensing tasks affects the quality of service. Because low-quality sensing data will lead to large analysis errors, the data center cannot mine valuable information from low-quality sensing data. Because almost all LPPMs will cause the loss of data quality to a certain extent, how to reduce the loss rate is an important problem. Root-mean-square error (RMSE) can be used as an index to measure the error between the observed values and the actual values. In this article, we use the value of RMSE, V_{RMSE} to measure the loss of data quality after obfuscation, and it is defined by

$$V_{\mathrm{RMSE}}(Seg, Seg') = \sqrt{\frac{1}{n}\sum\nolimits_{i=1}^{n}(V_i - V_i')^2} \tag{18}$$

where V_i is the sensing data of the region where the original stay point is located, V_i' is the sensing data in the target obfuscation subregion $r_{i,t}$. After obfuscating the original stay point to $r_{i,t}$, V_i' will be used as the the sensing data of $r_{i,t}$, so the RMSE between V_i' and V_i can be used to quantify the loss of data quality. The $\mathrm{V_{RMSE}}$ is not fixed for the same data when LPMT operates another time. It is the uncertainty that makes the user's location privacy not be easily disclosed.

C. Experiment Results

1) Space and Time Overhead: We first analyze the space overhead of LPMT. In the trajectory feature extraction phase, the space complexity of LPMT is $\mathrm{O}(n^2)$, where n depends on the number of candidate obfuscation subregions. In this experiment, we use the Geolife dataset. The sampling frequency of GPS points is once every 5 s, and the storage space occupied by the sensing data within one hour is about 40 KB. For most of the existing mobile devices, it is negligible to pay the space overhead of 40 KB.

TABLE IV
TIME COMPLEXITY OF EACH PHASE IN LPMT

PHASES	WORST TIME COMPLEXITY	AVERAGE TIME COMPLEXITY
TRAJECTORY FEATURE EXTRACTION	$\mathrm{O}(z^2)$	$\mathrm{O}(z^2)$
STAY POINTS OBFUSCATION	$\mathrm{O}(n*k)$	$\mathrm{O}(n*k)$
STAY POINTS GENERALIZATION	$\mathrm{O}(n^2)$	$\mathrm{O}(n^2)$

TABLE V
TIME OVERLOAD OF EACH PHASE IN LPMT

PHASE	t_1	t_2	t_3	t_4
TIME CONSUMPTION (MS)	20	182*n	43*n	61*n

In the following parts of this section, we will analyze the time overhead of LPMT. In the trajectory feature extraction phase, it needs to traverse all sequence windows. If the length of *Seg* is z, then the time complexity is $\mathrm{O}(z^2)$; In the stay points obfuscation phase, the location context similarity data is pre-downloaded from the data center, we can focus on SPOA. SPOA traverses each stay point, and selects a target obfuscation subregion for each stay point through the exponential mechanism. If the number of stay points of *Seg* is n, and the number of candidate subregions in r_i is k, the time complexity of this phase is $\mathrm{O}(n*k)$; In the stay points generalization phase, whether it is random sampling or truncation operation, the overall time complexity is $\mathrm{O}(n^2)$. Since the three phases are in sequential order, the overall time complexity of LPMT is $\mathrm{O}(n^2)$. In order to make it easy to follow, we listed the time complexity of each phase in Table IV.

In this experiment, we tested the real-world time consumption in LPMT. Table V shows the time consumption of each phase in LPMT, where n is the number of stay points.

In the trajectory feature extraction phase, the time consumption depends on the number of GPS points in *Seg*. In this article, we define *Seg* as the local buffered GPS points in one hour. In the Geolife dataset, sensing data is collected every 5 s, so *Seg* contains approximately 720 GPS points. After multiple experiments, it shows the time consumption in the trajectory feature extraction phase is less than 20 ms (t_1). In our experiment, the size of obfuscation region r_i is set to $1 \times 1\,\mathrm{km}^2$, and the size of each subregion in r_i is set to $100 \times 100\,\mathrm{m}^2$. It takes about 182 ms to complete the LCS calculation (t_2), and about 43 ms (t_3) and 61 ms (t_4) to obfuscate a stay point and generalize a stay point, respectively.

Note that LPMT deploys the function of LCS calculation, which is the most time-consuming operation, into the data center. The time overhead of the user devices is negligible, and only incurs small communication overhead.

2) Obfuscation Quality: In this section, we compare the obfuscation quality $\overline{Q}$ among GeoInd, $\mathrm{DUM}{-}\varepsilon\mathrm{e}$, LPMT ($\beta = 0.5$) and LPMT without LCS on different privacy budgets ε. In addition, we also compare the varieties of obfuscation quality

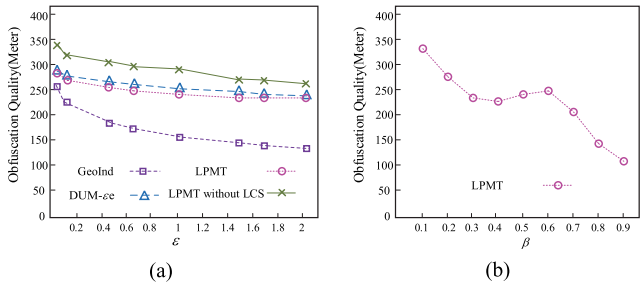

Fig. 7. Obfuscation quality. (a) On different privacy budgets. (b) In different utility weights.

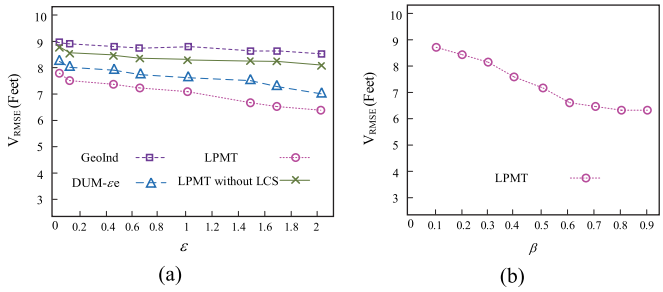

Fig. 8. Data quality loss. (a) On different privacy budgets. (b) In different utility weights.

when we change the utility weight β and keep a fixed privacy budget in LPMT.

Fig. 7(a) shows the obfuscation quality $\overline{Q}$ of LPPMs when increasing the privacy budget ε. It can be seen from Fig. 7(a) that the $\overline{Q}$ of all LPPMs decrease when ε increases, and it is because that LPPMs do not have to obfuscate locations too far away when there are enough privacy budgets. GeoInd has the worst obfuscation quality. It is because the obfuscated GPS points are distributed around the original stay points in GeoInd, which leads to the distance between the obfuscated stay points and the original ones not far enough. Since LCS is considered, the $\overline{Q}$ value of LPMT ($\beta = 0.5$) become a little lower, which is 1.2%–2.8% less than that of DUM$-\varepsilon$e. It is because that the Euclidean distance between the stay point and the target obfuscation subregion is positively correlated with the obfuscation quality, and the intruduction of LCS tends to obfuscate the stay point to a closer subregion. Because of not considering the LCS between subregions, the $\overline{Q}$ value of LPMT without LCS can increased by 5.1%–10.4% compared to those of LPMT ($\beta = 0.5$), which performs the best among all baselines in this experiment.

Fig. 7(b) shows the changes of obfuscation quality $\overline{Q}$ of LPMT when we increase utility weight β and keep $\varepsilon = \ln 2$. As shown in Fig. 7(b), when β increases from 0.1 to 0.4, $\overline{Q}$ keeps decreasing rapidly; When β lies between 0.5 and 0.6, $\overline{Q}$ has a little increment; When β is above 0.7, $\overline{Q}$ drops significantly. As mentioned before, increasing the vaue of β (i.e., the weight of LCS in the utility function) tends to obfuscate a stay point to a closer subregion (in general, LCS between adjacent subregions is higher), which leads to a decrease in the obfuscation quality.

3) Data Quality: In this section, we compare the data quality loss RMSE among GeoInd, DUM$-\varepsilon$e, LPMT ($\beta = 0.5$), and LPMT without LCS on different privacy budgets ε. Moreover, we also compare the RMSE of LPMT when the utility weight β changes while keeping a fixed privacy budget.

Fig. 8(a) shows the changes of RMSE of LPPMs when we increase the privacy budget ε. It can be seen from Fig. 8(a) that the RMSE of all LPPMs decrease when ε increases. It is because that LPPMs do not have to obfuscate locations too far away when there is a lower privacy level. LPMT has the lowest RMSE, which has the best performance on data quality. Compared with GeoInd and DUM$-\varepsilon$e, LPMT can reduce the RMSE value by 12.2%–26.9% and 4%–12.9%, respectively. The reasons are as follows. First of all, because GeoInd does not consider the impact on data quality when it performs location obfuscation, the data error rate after obfuscation is higher than that of other methods. Second, because DUM$-\varepsilon$e obfuscates all the GPS points of participants, while LPMT only obfuscatse the original GPS points sequence $\{p_x, \ldots, p_y\}, 1 \leq x < y \leq z$ that can generate a stay point, and retains the remaining original GPS points, LPMT has a obvious advantage in data quality loss. In addition, compared LPMT ($\beta = 0.5$) with LPMT without LCS, it can be found that the RMSE of LPMT can be reduced up to 20.1% after integrating LCS.

Fig. 8(b) shows the RMSE of LPPMs when we increase utility weight β and keep a fixed $\varepsilon = \ln 2$. As shown in Fig. 8(b), When we increase β from 0.1 to 0.6, the RMSE of LPMT decreases significantly. It is because that the increasing weight of LCS can help LPMT obfuscate the locations to similar ones. After that, the RMSE tends to keep stable when we continues to increase β. From Figs. 7(b) and 8(b), it can be concluded that it will be well balanced between the obfuscation quality and data quality when the privacy budget ε is set to ln2 and the utility weight β is set to 0.6.

VII. Conclusion

This article proposed a differential location privacy-preserving mechanism named LPMT for the distributed intelligence scenario. LPMT is designed based on the non-trusted server framework. The implementation of LPMT includes three phases as follows:

1) in the trajectory feature extraction phase, it uses a sliding window algorithm to accelerate the extraction of stay points;
2) in the stay points obfuscation phase, by introducing a utility function that combines Euclidean distance and location context similarity, it can better balance the obfuscation quality and the data quality loss;
3) in the stay points generalization phase, it uses the Laplace mechanism for GPS point sampling to meet the differential privacy constraint.

Finally, this article conducts experiments on a real-world dataset to calculate the obfuscation quality $\overline{Q}$ and the data quality loss RMSE of LPMT. The experiment results showed that LPMT can reduce the data quality loss up to 20.1% while ensuring a comparable obfuscation quality to baselines.

Currently, LPMT is suitable for near real-time and non-real-time MCS scenarios. We plan to research the location privacy

protection in the real-time MCS scenarios in the future. Furthermore, we will extend LPMT by combining task types and geographic location semantics to optimize the calculation of location context similarity, which could further reduce the data quality loss.

REFERENCES

[1] F. Khan, A. U. Rehman, J. Zheng, M. A. Jan, and M. Alam, "Mobile crowdsensing: A survey on privacy-preservation, task management, assignment models, and incentives mechanisms," *Future Gener. Comput. Syst.*, vol. 100, pp. 456–472, 2019.

[2] X. Yi, R. Paulet, E. Bertino, and V. Varadharajan, "Practical approximate k nearest neighbor queries with location and query privacy," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 6, pp. 1546–1559, Jun. 2016.

[3] P. Zhao, J. Li, F. Zeng, F. Xiao, and C. Wang, "ILLIA: Enabling k-anonymity-based privacy preserving against location injection attacks in continuous LBS queries," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 1033–1042, Apr. 2018.

[4] M. Yamin and A. A. Abi Sen, "Improving privacy and security of user data in location based services," *Int. J. Ambient Comput. Intell.*, vol. 9, no. 1, pp. 19–42, 2018.

[5] S. Boukoros, M. Humbert, S. Katzenbeisser, and C. Troncoso, "On (the lack of) location privacy in crowdsourcing applications," in *Proc. 28th USENIX Secur. Symp.*, 2019, pp. 1859–1876.

[6] M. Gruteser and D. Grunwald, "Anonymous usage of location-based services through spatial and temporal cloaking," in *Proc. 1st Int. Conf. Mobile Syst. Appl. Serv.*, 2003, pp. 31–42.

[7] L. Sweeney, "k-anonymity: A model for protecting privacy," *Int. J. Uncertainty, Fuzziness Knowl.-Based Syst.*, vol. 10, no. 5, pp. 557–570, 2002.

[8] Y. B. Zhang, Q. Y. Zhang, Z. Li, and M. Y. Zhang, "A k-anonymous location privacy protection method of dummy based on geographical semantics," *Int. J. Netw. Secur.*, vol. 21, no. 6, pp. 937–946, 2019.

[9] A. R. Beresford and F. Stajano, "Mix zones: User privacy in location-aware services," in *Proc. 2nd IEEE Annu. Conf. Pervasive Comput. Commun. Workshops*, 2004, pp. 127–131.

[10] N. Guo, L. Ma, and T. Gao, "Independent mix zone for location privacy in vehicular networks," *IEEE Access*, vol. 6, pp. 16842–16850, 2018.

[11] I. Memon, Q. A. Arain, M. H. Memon, F. A. Mangi, and R. Akhtar, "Search me if you can: Multiple mix zones with location privacy protection for mapping services," *Int. J. Commun. Syst.*, vol. 30, no. 16, 2017, Art. no. e3312.

[12] T. H. You, W. C. Peng, and W. C. Lee, "Protecting moving trajectories with dummies," in *Proc. Int. Conf. Mobile Data Manage.*, 2007, pp. 278–282.

[13] P. R. Lei, W. C. Peng, I. J. Su, and C. P. Chang, "Dummy-based schemes for protecting movement trajectories," *J. Inf. Sci. Eng.*, vol. 28, no. 2, pp. 335–350, 2012.

[14] M. E. Andrés, N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi, "Geo-indistinguishability: Differential privacy for location-based systems," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2013, pp. 901–914.

[15] L. Wang, D. Zhang, D. Yang, B. Y. Lim, and X. Ma, "Differential location privacy for sparse mobile crowdsensing," in *Proc. IEEE 16th Int. Conf. Data Mining*, 2016, pp. 1257–1262.

[16] C. Dwork, "Differential privacy: A survey of results," in *Proc. Int. Conf. Theory Appl. Models Computation*, 2008, pp. 1–19.

[17] F. McSherry and K. Talwar, "Mechanism design via differential privacy," in *Proc. 48th Annu. IEEE Symp. Found. Comput. Sci.*, 2007, pp. 94–103.

[18] H. To, G. Ghinita, and C. Shahab, "A framework for protecting worker location privacy in spatial crowdsourcing," *Proc. VLDB Endowment*, vol. 7, no. 10, pp. 919–930, 2014.

[19] R. Assam, M. Hassani, and T. Seidl, "Differential private trajectory obfuscation," in *Proc. Int. Conf. Mobile Ubiquitous Syst.: Comput. Netw. Serv.*, 2012, pp. 139–151.

[20] S. Gisdakis, T. Giannetsos, and P. Papadimitratos, "Security, privacy, and incentive provision for mobile crowd sensing systems," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 839–853, Oct. 2016.

[21] J. Wang, M. Li, Y. He, H. Li, K. Xiao, and C. Wang, "A blockchain based privacy-preserving incentive mechanism in crowdsensing applications," *IEEE Access*, vol. 6, pp. 17545–17556, 2018.

[22] Q. Li, Y. Zheng, X. Xie, Y. Chen, W. Liu, and W. Y. Ma, "Mining user similarity based on location history," in *Proc. 16th ACM SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst.*, 2008, pp. 1–10.

[23] N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi, "Optimal geo-indistinguishable mechanisms for location privacy," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2014, pp. 251–262.

[24] Y. Zheng, X. Xie, and W. Y. Ma, "Geolife: A collaborative social networking service among user, location and trajectory," *IEEE Database Eng. Bull.*, vol. 33, no. 2, pp. 32–39, Jun. 2010.

Zhigang Gao received the Ph.D. degree in computer science and technology from Zhejiang University, Hangzhou, China, in 2008.

He is currently an Associate Professor with Hangzhou Dianzi University, Hangzhou. His current research interests include mobile computing and cyber-physical systems.

Yucai Huang received the B.S. degree in measurement-control technology and instrumentation specialty from Zhejiang Gongshang University, Hangzhou, China, in 2019. He is currently working toward the M.S. degree in computer technology with Hangzhou Dianzi University, Hangzhou.

His research interests include mobile learning, IoT security, neural networks, and location privacy protection.

Leilei Zheng received the B.S. degree in computer science and technology from Taishan University, Taian, China, in 2020. She is currently working toward the M.S. degree in computer technology with Hangzhou Dianzi University, Hangzhou, China.

Her research interests include mobile learning, IoT, interpretable AI models, and so on.

Huijuan Lu received the Ph.D. degree in control theory and engineering from the China University of Mining and Technology, Xuzhou, China.

He is currently a Professor with China Jiliang University, Hangzhou, China. Her research interests include mobile computing and artificial intelligence.

Bo Wu (Member, IEEE) received the Ph.D. degree in human sciences from Waseda University, Tokyo, Japan, in 2015.

He is currently an Assistant Professor with the School of Computer Science, Tokyo University of Technology, Tokyo. He is also a Research Follow with the Advanced Research Center for Human Sciences, Waseda University and DS Laboratory, Kansai University. His current research interests include human motion capture, eye-movement analysis, machine learning and big data analysis, and human-centered application system development.

Jianhui Zhang received the Ph.D. degree in control science and engineering from Zhejiang University, Hangzhou, China, in 2008.

He was the Postdoc with Ottawa University, Ottawa, Canada, in 2010. He is currently a Professor in Computer Science with the Hangzhou Dianzi University, Hangzhou, and the Leader of IoT research group. His research interests include AIoT and embedded system design, smart city computing, and wireless sensor networks.