The framework and algorithm for preserving user trajectorywhile using location-based services in IoT-cloud systems

[1]**PRIYABRATA SAHOO,**
Gandhi Institute of Excellent Technocrats, Bhubaneswar, India
[2]**SUBHALAXMI PANDA,**
Capital Engineering College, Bhubaneswar, Odisha, India

Abstract Internet of things (IoT) based location-based services (LBS) are playing an increasingly important role in our daily lives. However, since the LBS server may be hacked, malicious or not credible, there is a good chance that interacting with the LBS server may result in loss of privacy.As per journal instruction, author photo and biography are mandatory for this article. Please provide. Thus, protecting user privacy such as the privacy of user location and trajectory is an important issue to be addressed while using LBS. To address this problem, we first construct three kinds of attack models that may expose a user's trajectory or path while the user is sending continuous queries to a LBS server. Then we construct a novel LBS system model for preserving privacy, and propose the *k*-anonymity trajectory (KAT) algorithm which is suitable for both single query and continuous queries. Different from existing works, the KAT algorithm selects k_1 dummy locations using the sliding window based *k*-anonymity mechanism when the user is making single query, and selects k_1 dummy trajectories using the trajectory select mechanism for continuous queries. We evaluate the effectiveness of our proposed algorithm by conducting simulations for the single-query and continuous-query scenarios. The simulation results show that our proposed algorithm can protect privacy of users better than existing approaches, while incurring a lower time complexity than those approaches.

Keywords Location privacy · Trajectory privacy · *k*-anonymity · LBS · Internet of things

1 Introduction

Cloud computing offers networked and remote computing resources to process, manage, store and share huge volume of internet of things (IoT) data, and allows small and scattered IoT devices to interact with its powerful backend capabilities related to data analytics and control [1–4]. Its great potentiality to those mobile and IoT devices provides substantially improved Quality of Service (QoS) to the applications requiring low latency, high bandwidth, location awareness, strong mobility, and widespread geographical distribution [5,6]. Consequently, cloud computing makes it suitable for location-based services in IoT.

The rapid development in communication and mobile device technology has enabled the sensing and analysis of a user's environment with respect to the user's context. Location and location related information forms the core context in a pervasive computing environment. Advances in sensing technology allow us to easily obtain location and positioning information with high accuracy using global positioning

systems (GPS), differential GPS (DGPS), and wide area augmentation systems (WAAS). The standard accuracy of GPS is approximately 15 m, and can be augmented to 3–5 m with DGPS, and to about 3 m with WAAS. Such location data is being used in a variety of location-based services (LBS) [7–11], which have gained rapid adoption during the recent past. Using LBS, users can easily get the information they want by using the hand-held terminals (e.g., smart phones, tablets). For example, users can find the nearest restaurant or hospital [12].

Privacy issues have not been addressed with respect to emerging IoT sensors, devices, cars, home appliances, drones and other applications that are expected to reach and use magnitude of location data as well as personal data. The identity of location data leads to other types of crime. One of the most important definition of privacy is the right to control the flow of one's personal data. The key research areas include:

- location based privacy;
- privacy of cloud computing data;
- effective privacy definitions including some domain-specific privacy definitions;
- effective monitoring of social groups interactions;
- some aspect of privacy-driven policies and software development paradigm;
- accountability techniques for privacy violations;
- techniques and mechanisms for tracking information flow and control of that flow;
- techniques on large scale location based privacy data analytics.

However, using LBS can cause privacy concerns [13,14]. To use LBS, users send requests to the LBS provider (LP) with their precise location information. However, there is no guarantee that the LP is credible and trustworthy [15]. A malicious LP may steal or expose location data, even sell user location information to a third party. In the era of Big Data, the data obtained from the third party may be mined for more information, not only the user's position, but also user's motion patterns, gender, age, occupation, hobbies, etc. This may result in the users getting spam mail, advertisements, or cause other more serious issues that can affect the user's normal life.

Many researchers have proposed solutions [16–19] to address privacy preservation issues for LBS for single query. The *k*-anonymity technique, which was originally developed for complying with privacy requirements in HIPAA [20], has been adapted for addressing the privacy preservation problem in LBS [21]. The objective of the original k-anonymity algorithm is to ensure that any individual cannot be uniquely identified within a group of k people. In the adapted k-anonymity for LBS technique, the user sends true position data along with $k - 1$ dummy or false location data when requesting for service, so that LBS server cannot distinguish the user's real location from the k locations. So the *k*-anonymity technique plays a significant role in protecting user's position when a single query is made. However, in reality users may send service requests continuously for a period of time [22]. We call this kind of request "*continuous queries*", and are assumed to arise from a certain behavior "model". For example, a tourist submits a request to a LBS server for finding the nearest hotel, but when he reaches the hotel he finds it unsatisfactory (too expensive, not clean etc.) and so he initiates another request till he finds a desirable hotel. Because of the correlation of various positions in the continuous queries, *k*-anonymity technology is no longer suitable. There is no guarantee that the tourist's movements have not been exposed. The tourists' trajectory information (hotels visited), which are part of their behavior models, can be exploited to uniquely identify the individual and allowing deducing the user's information, e.g., earning ability, social class, and marital status situation etc. Therefore, protecting user location and trajectory information in LBS is an important issue that needs to be solved.

In this paper, we design an efficient algorithm based on the *k*-anonymity technique to protect user trajectory privacy in location based services. Our scheme is not only suitable for the continuous queries, but also applicable for the single query scenario. The main contributions of this paper are as follows:

- As we noted that the LP may be malicious and not credible. The traditional *k*-anonymity technique is not suitable to preserve the user trajectory privacy, since there are some scenarios and attacks where the user's trajectory privacy may be exposed. We consider three main attacks in this paper: *shared attack*, *roadless attack* and *probability attack*.
- We propose the *k*-anonymity trajectory algorithm (KAT) for preserving privacy, which can be applied to both single and continuous query scenarios. KAT is superior to the dummy location selection (DLS) algorithm proposed in [12], since the DLS algorithm is time consuming in selecting other $k-1$ candidate locations form the $2k$ dummy locations. Whereas the KAT algorithm collects other $k-1$ candidate locations by using the *sliding window* based *k*-anonymity mechanism, which can significantly to reduce the time complexity.
- To protect trajectory privacy, we introduce the *maximum entropy* and the *trajectory select mechanism* into our KAT algorithm for choosing other $k-1$ dummy trajectories to resist attacks in the continuous query scenario.

The remainder of this paper is organized as follows. Section 2 discusses the related work. The problem statement is described in Sect. 3. Section 4 gives the preliminaries and

system model. Section 5 presents the detailed description for the proposed KAT algorithm for protecting location privacy and trajectory privacy. The simulation results and analysis are given in Sect. 6. Section 7 gives the discussion, and Sect. 8 concludes the paper.

2 Related work

Significant attention has been paid on the problem of user privacy preservation for single query in LBS. For example, the *k*-anonymity scheme applied in single query scenario obtains considerable success with respect to protecting users' location privacy [23–27]. There are three commonly used methods to realize *k*-anonymity to combat a malicious LBS provider (LP).

- When user issues a request, the user advertises *k* positions including one real location and $k-1$ dummy locations. In such a situation an attacker can infer a user's real location with probability of $1/k$ [28].
- For privacy preserving, the work in [29,30] realize *k*-anonymity with a trusted anonymizer server. When a user issues a request, he/she will first send request to the anonymizer server. After receiving request message of *k* users, the anonymizer server generates a cloaking region for the *k* users and uses the cloaking region as the shared location of the *k* users. Then the anonymizer server sends requests for *k* users to LBS provider (LP). So the LP receives information of *k* users with the same location, and infers each user with probability of $1/k$.
- In the distributed LBS system [31], *k*-anonymity is realized through an aggregation protocol. When a user issues a request, the user will aggregate other $k-1$ users using aggregation protocol and select a representative user from the *k* users. The representative user will send all request packets for the *k* users to the LP. Therefore, the LP cannot distinguish each user and infer one user with the probability of $1/k$.

Protecting user trajectory information has received attention recently. The work in [31,32] focuses on trajectory privacy protection for the continuous queries scenario. The work in [32] proposed a virtual path programming solution for trajectory preserving and [31] proposed a location anonymity scheme based on the fake queries in continuous location-based services. In [33] the authors employed a Track False Data method with pseudonymization and perturbation, so as to not release user's real trajectory data, by using the false tracks when user sends requests to the LP. For example, assume that the real path of a user is t_1, which is composed of original locations l_1, l_2 and l_3. First, the Track False Data method completes pseudonymization of l_1, l_2 and l_3, and generates three pseudo locations l_1^r, l_2^r and l_3^r. On the basis of these pseudo locations, we can construct the other three new locations l_4^r, l_5^r and l_6^r by using perturbation. Thus it can construct a false track t_2 including the locations l_4^r, l_5^r and l_6^r. When a user with real path t_1 needs to submit a request to the LP, the user will send the false path t_2 to LP rather than the real path t_1 to protect their location and trajectory privacy. The larger the range of pseudonyms and perturbation, the higher the degree of trajectory protection. However, this will result in a significant effect on the real location data, and may greatly reduce the quality of service for users. For protecting users' trajectory privacy, in [34] Xu and Cai used the method of historical data generalization, based on a trusted anonymization server which is a third party between the LP and user. When a user needs to send request to the LP, the user will first send it to the trusted anonymization server. Assume that the users' real path is $t_r = \{l_1, l_2, \ldots l_i, \ldots, l_m\}$. The trusted anonymization server will convert the t_r area into $R = \{r_2, \ldots r_i \ldots, r_m\}$. Each cloaking area r_i is selected carefully based on the users' historical location data and real location l_i. Then the trusted anonymization server sends $R = \{r_1, r_2, \ldots r_i \ldots, r_m\}$ to the LP. Thus, the cloaking area r_i replaces user's real location and protects trajectory privacy. However, if there are many users using the LBS system, the anonymization server can be a performance bottleneck or single point of failure. The researches in [35–40] consider the continuous query scenario for protecting trajectory privacy in LBS. In this work, we propose an effective method which is suitable for both the continuous queries and the single query scenario to protect location and trajectory privacy.

3 Problem statement

In this work we mainly focus on solving two problems: (i) location privacy preservation for single query, (ii) preserving trajectory privacy for continuous queries. For the first problem, we design an efficient solution by improving the DLS [12] algorithm. Based on the side information (user's query probability), each location has a probability of being queried in the past. Assume that the probability of real location is p_d. To achieve *k*-anonymity, the DLS algorithm chooses other $k-1$ dummy locations from $2k$ candidates. Thus, it should loop C_{2k}^{k-1} times to achieve the maximum entropy. Hence, the time complexity of the DLS algorithm increases exponentially with the growth of the value of *k*. Although the DLS algorithm has guaranteed the maximum entropy, the *k* probabilities of locations may not be similar. Some of them may be much smaller or larger than the probability p_d. Attackers can easily filter out the impossible locations and guess the real location of user with a higher probability. To better preserve location privacy and reduce the time complexity, we design a sliding window based *k*-anonymity mechanism in this work.

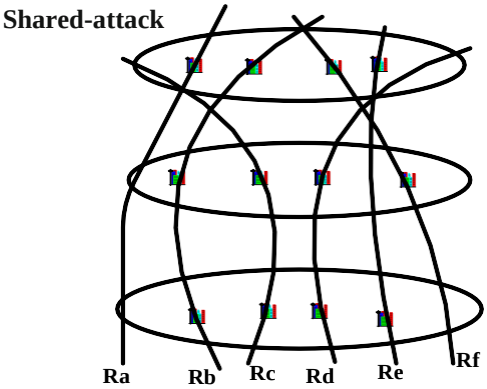


Fig. 1 The shared-attack

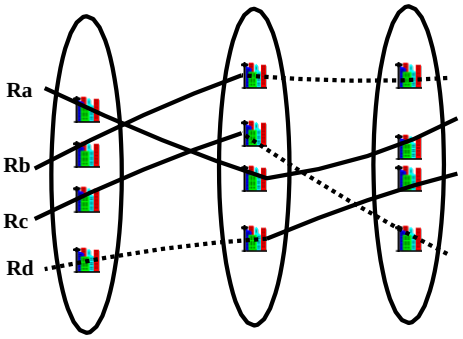


Fig. 2 The roadless-attack

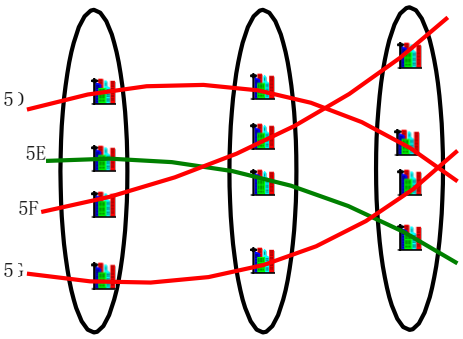


Fig. 3 The probability-attack

For protecting trajectory privacy in continuous queries, we first study the case where trajectory privacy is exposed and propose three main attack models. In our attack model (showed in Figs. 1, 2, 3), we assume that user u_i employs the traditional *k*-anonymity technique for preserving privacy and the LP is malicious. The value of *k* is the minimum degree of privacy that a user can tolerate. Here, we set the degree of privacy as $k = 4$. Without loss of generality, we suppose that each user sends three queries in a time period. Since each user sends query with the real location and the other k 1 dummy locations to the LP, each query includes 4 locations. Therefore, we can get three areas termed as "*anonymity area*" for the three queries.

As shown in Fig. 1, there are four locations (denoted by the small squares) in an anonymity area (denoted by circle). The malicious LP scans all trajectories between these locations and gets six trajectories R_a, R_b, R_c, R_d, R_e, R_f by matching with a map (e.g., from Google Maps). Then the LP can easily deduce that the user's real trajectory is R_b, because the intersection of the three anonymity areas is on the trajectory of R_b. Thus, we should think about the correlation of the requested locations. The user's real trajectory and the alternative trajectories selected by the attacker should be shared in all of the anonymous area. There are at least two trajectories shared in all of the anonymous area, otherwise, user's trajectory privacy may be easily exposed.

Roadless-attack

As shown in Fig. 2, before matching with a map (e.g., Google Maps), the LP may consider the four trajectories are shared in the anonymity areas. However, after matching with the map, the LP discovers that there are no roads on some of the trajectories (pictured using dash line). Accordingly the LP filters the impossible trajectories and infers the user's real trajectory is R_a. Thus, we should prevent choosing those locations that have no roads between them while selecting dummy trajectories.

Probability-attack

Figure 3 indicates that the four trajectories are shared in all three anonymity areas, and we could know that the four trajectories are real after matching with the map. However, if the attacker achieves the side information [12], e.g., query probability of location-vertex and trajectory-edge, then he/she would find that there are three trajectories R_a, R_c, R_d (pictured in the red line) have much smaller probability than the trajectory R_b (pictured in the green line). Thus, the attacker can easily infer that user's real trajectory is R_b.

Considering the three kinds of attack model, we propose the *k*-anonymity trajectory (KAT) algorithm in Sect. 5, to preserve users' trajectory privacy. In our proposed KAT algorithm, it can be used to protect users' location privacy in the single request scenario, as well protect users' trajectory privacy in the continuous requests scenario.

4 Preliminaries and system model

In this section, we first present some basic concepts and some definitions used in this paper, and then describe our system model in detail.

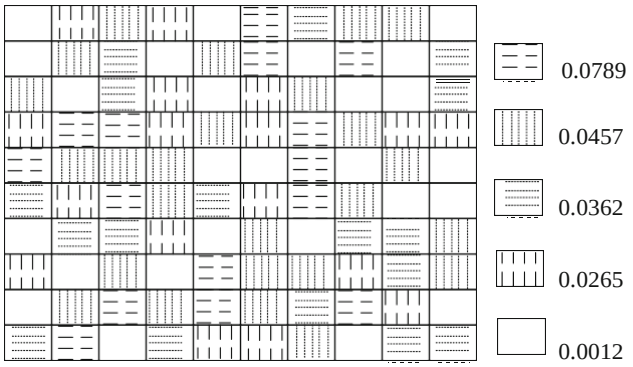


Fig. 4 The historical probability table of location-vertex

Fig. 5 The historical probability table of trajectory-edge

Preliminaries

(1) *User location* The location of a user is denoted by $d(x, y)$, where x represents the latitude and y represents the longitude.

(2) *Request packet* The request packet of a user is denoted by $Req = (PID, L, K, t, r, \theta_{max}, \theta_{min})$, where *PID* is the pseudonym identification to identify the user uniquely, L is the set of positions (e.g., $L = \{d_1, d_2, \ldots, d_k\}$), including the user's real location and $k-1$ dummy locations, K is the minimum tolerable degree of privacy for a user (it denotes the number of candidate locations/trajectories in this work). t denotes the sending time for the request; and r is the serviced content of the request (e.g., entertainment, dining, dating information request). θ_{max} and θ_{min} are used to determine the scope of the area for selecting the dummy location, where θ_{max} and θ_{min} are the maximum and minimum radius of the circular area. The size of θ_{max} and θ_{min} is set according to the user's privacy degree K, walking speed and the time interval of submitting request.

(3) *Historical probability of location-vertex* As shown in Fig. 4, we divide a large area (location map) into many small grids of cells according to [12]. We can get the request times of each small area in accordance with the historical statistics, called as frequency of the request, denoted by f_i. It is very time-consuming to count the request times of each small area, so it is counted by a special counter server and updated once a week or a month. Then we can calculate the corresponding historical probability of each small cell, denoted by $p_1, p_2, \ldots, p_n$.

$$p_i = \frac{f_i}{\sum_{l=1}^{n} f_l}, \quad i = 1, 2, \ldots, n \tag{1}$$

Figure 4 shows that each location has a probability of being queried in the past [12]. A user using the LBS service can get the probability of their location from the historical probability table of location-vertex.

(4) *Historical probability of trajectory-edge* We can construct an undirected graph (V, E) by modeling Fig. 4. The graph (V, E) is shown in Fig. 5. V is the point set of graph, which is the center of the small area. E is the edge set of graph. There exists an edge if the distance of two points meets the required threshold and there exists a real road on geographical map, otherwise it is no edge between the two points. The distance threshold of two points is related to user's moving speed and the time interval between requests. Since it is impossible far away infinitely between the positions of the user's request at the current moment and the next moment, there is no edge beyond the threshold. We call the sum of the two endpoints frequency of an edge as the edge's frequency. For example, for the edge $e = (v_1, v_2)$, the edge frequency can be calculated as follows:

$$f_e = f_{v1} + f_{v2} \tag{2}$$

Then we can calculate historical probability of trajectory edge $e$as following:

$$q_{e_i} = \frac{f_{e_i}}{\sum_{j=1}^{m} f_{e_j}}, \; j = 1, 2, \ldots, m \tag{3}$$

Theorem 1 *The greater the probability of the two endpoints on an edge is, the larger the probability of the corresponding edge is.*

Proof For a point of interest on one route, we count the number of historical request and calculate the corresponding historical probability of this position. If this probability is high, it implies that many users pass the route. Hence, the probability of the corresponding edge (the route) is also high. ∎

(5) *Entropy* In this work, we measure the degree of anonymity based on entropy. According to the knowledge of information theory, entropy reflects the uncertainty of a random variable. The greater the entropy is,

the more uncertain information has. So we can know that if the entropy of the probability of location-vertex or trajectory-edge is maximal, the LP cannot determine the real user in which position or in which trajectory at the most extent.

We denote $\{p_1, p_2, \ldots, p_k\}$ as the k probabilities for the occurrence of an event, where the sum of all p_i is 1. Then the entropy H for the k probabilities is defined as follows:

$$H = -\sum_{i=1}^{k} p_i . \log_2 p_i \tag{4}$$

Therefore, $H_{\max} = \log_2 k$, where $p_i = 1/k$, $i = 1, 2, \ldots, k$.

When a user sends request *Req* to the LP, the content L of *Req* includes the candidate set of locations, denoted by $\{d_1, d_2, \ldots, d_k\}$, whose corresponding historical probability is $\{p_1, p_2, \ldots p_i \ldots, p_k\}$. We note that the sum of k selected probabilities is less than 1, whereas the sum of all the probabilities in the *historical probability of location-vertex* is 1. Thus, we need to normalize these probabilities, which we denoted as $\{p_{d1}, p_{d2}, \ldots, p_{dk}\}$, before computing the *entropy*.

$$p_{di} = \frac{p_i}{\sum_{j=1}^{k} p_j}, \quad j = 1, 2, \ldots, k \tag{5}$$

Now, we can ensure the $\sum_i p_{di} = 1, i = 1, 2, \ldots, k$. Then we can compute the entropy.

$$H = -\sum_{i=1}^{k} p_{di} . \log_2 p_{di} \tag{6}$$

When a user sends requests, there must be a trajectory between the two continuous queries. Thus, the trajectory has a historical trajectory probability q. Matching with the *historical probability table of trajectory-edge*, we can carefully choose $k - 1$ dummy trajectories $\{q_1, q_2, \ldots, q_{k-1}\}$.

We first normalize the probability of k trajectory $\{q^r, q_1^r, q_2^r, \ldots, q_{k-1}^r\}$ as follows.

$$q^r = \frac{q}{q + \sum_{i=1}^{k-1} q_i}, \quad q_i^r = \frac{q_i}{q + \sum_{i=1}^{k-1} q_i} \tag{7}$$

Then we have the corresponding entropy.

$$H = -q^r . \log_2 q^r - \sum_{i=1}^{k-1} q_i^r . \log_2 q_i^r \tag{8}$$

We selected k candidate historical probabilities for trajectory edge based on the principle of the maximum entropy, and then we call the k endpoints of dummy trajectories as the k candidate locations.

As shown in the above formulation, we should select the dummy position carefully, whose historical probability close to the probability of user's real location as much as possible. Otherwise, the attacker can easily filter out the impossible position whose historical probability is too large or too less compared with the probability of user's real position, and easily to find out the real location of user. For example, there are l probabilities of dummy locations that are much smaller than the probability of user's real location, the attacker filters out the l dummy locations, and then infers the real location with the probability from $\frac{1}{k}$ to $\frac{1}{k-l}$.

(6) *Anonymity degree* As mentioned before k represents that there are k candidate locations/trajectories. According to *the historical probability table of location-vertex/trajectory-edge*, we can get k corresponding probabilities. In order to ensure the k probabilities are similar, we define the anonymity degree D as follows:

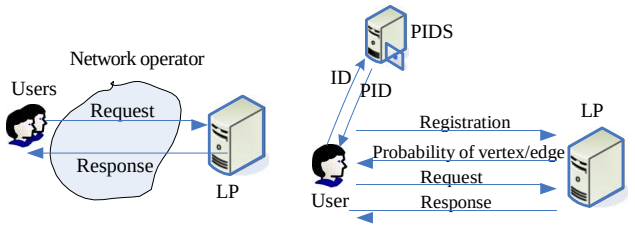
$$D = 2^H \in [K - \varepsilon, K] \tag{9}$$

There is a difference between anonymity degree D and privacy degree K. As mentioned before K is the minimum degree of privacy that user can tolerate. It is set according to user's own requirement before sending request. However, anonymity degree D is applied in the real-time process of anonymity. Generally, the anonymity degree D is less than K. Because the maximum entropy $H = \log_2 k$ at $p_i = \frac{1}{k}$ $i = 1, 2, \ldots, k$, so the $D \leq K$. The smaller the D, the bigger the difference between the k probabilities. Thus, D can be used to limit the similarity in the k probabilities by setting an appropriate value for ε.

Theorem 2 *When the anonymity degree D is much less than privacy degree K (i.e., D K), the DLS algorithm fails to protect the location privacy of the user.*

Proof When D K, some probability of candidate locations are far away from the probability of user's real location. Thus, we can filter out these impossible locations whose probabilities are much smaller or larger than the real location. Assume there are k_n impossible locations. Then we can infer user's real location with a probability of $1/(k - k_n)$, which is higher than the theoretical probability $1/k$. Thus, the DLS algorithm fails to preserve location privacy when D K.

In this work, for preserving location privacy in single query scenario, we design the KAT algorithm to meet the requirement of *anonymity degree*.



(a) The existing LBS system **(b)** Our system model

Fig. 6 The LBS system

System model

Figure 6a shows the existing LBS system. Users can easily submit queries to the LBS server with a hand-held (e.g. Smart phones, tablet) and get the information they want. In the LBS system shown in Fig. 6a, a user directly submits request with real location to a LBS provider, which may be untrusted, resulting in the loss of location and trajectory privacy. For preserving privacy, we propose a novel system model, as showed in Fig. 6b composed of three major components: USER, PIDS and LP. Where the USER is the subscriber, PIDS is the pseudonym identity server generating a pseudo identity for users, and the LP is the LBS provider. We assume that each component in our system model works in accordance with the relevant legal agreement, it does not rule out each component's curiosity to collect data through snooping on others. Our system model can be divided into three main stages as follows.

Stage 1 Pseudo identity (PID) distribution. When a "*new user*" wants to use the LBS services, the user registers with the PIDS. The PIDS verifies and determines that whether the user is legal and distributes a pseudo identity to the legal user. If the user is illegal, the LBS system will refuse to serve him. In this work, a new user not only refers to a newly arriving user who submits a query for the first time, but also refers to an old user who is reusing the LBS. For example, assume that a user is sending continuous requests for LBS, if the user does not find k_1 appropriate dummy locations to form k-anonymity trajectory, then the user must reapply for another PID. At this time, the "old user" becomes a "new user". Therefore, a user may have more than one pseudo identity in the process of processing continuous requests. For the LBS provider, it may be confused and believe that more than one user are sending requests at the period of time. In this stage, a user submits only their real ID as part of the query, so PIDS cannot deduce other information about the user. Each legal user using the LBS service is given a PID by the PIDS.

Stage 2 The validation process. The main purpose of the validation process is to ensure that users do not forge the PID and once again confirm that they are legal. When the LP receives the request, it will check the user' PID through communicating with the PIDS. If the user is legal, the LP returns some side information. The side information is limited to user' query times for historical locations or trajectories. There are two kinds of side information for our LBS system: historical probability table of location-vertex and historical probability table of trajectory-edge. For example, if a legal user sends a single request, the LP will return the historical probability table of location-vertex. And if he/she sends the continuous request, the LP will return historical probability table of trajectory-edge.

Stage 3 Request processing. Using the information about the historical probability table of location-vertex or historical probability table of trajectory-edge, user chooses other k_1 dummy locations and sends request to the LP. Then the LP responds to the user based on the request content. For example, if a user is asking "where is the nearest restaurant", the LP will return k restaurant messages to the user based on the k positions. Then the user finds out the useful information according to his real location.

5 Algorithm design

In this section, we first present framework of the k-anonymity trajectory (KAT) preservation algorithm. Then we introduce the KAT algorithm which includes the Sliding Window based k-anonymity (SWK) algorithm for single query and the trajectory select mechanism (TSM) algorithm for continuous queries.

The KAT algorithm

Figure 7 describes the framework of the KAT algorithm. It employs the sliding window based k-anonymity (SWK) algorithm while sending a single request or once before sending continuous requests; whereas for continuous requests, the KAT algorithm calls the trajectory select mechanism (TSM) algorithm. If a request does not satisfy the requirement (e.g., the selected k trajectory cannot meet the anonymity degree D), the KAT algorithm will re-call the SWK algorithm. *Algorithm* 1 describes the pseudo code of KAT algorithm.

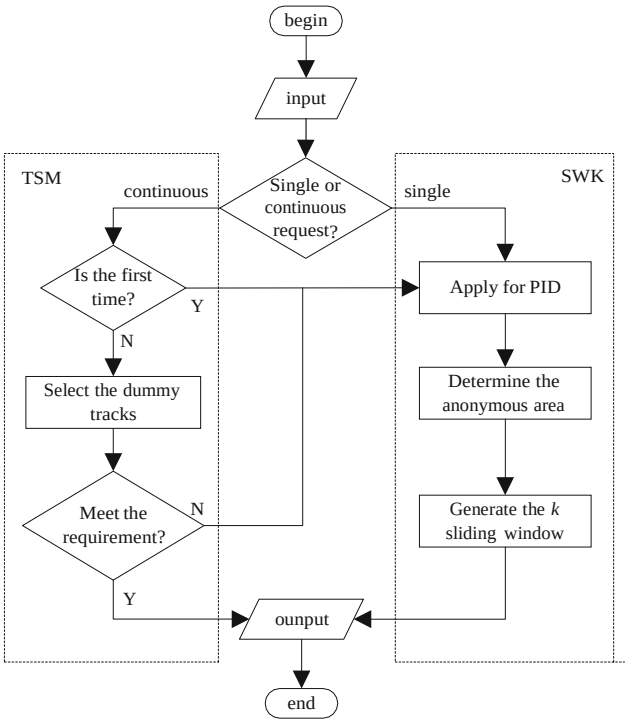


Fig. 7 The framework of the KAT algorithm

***Algorithm* 1: *K*-Anonymity Trajectory (KAT)**

1: **if** (single request)
2: Call the SWK algorithm.
3: **else if** (continuous request)
4: Call the TSM algorithm.
5: **if** (it is the first time request || the false tracks don't meet the requirement)
6: Call the SWK algorithm.
7: **end if**
8: **end if**

The SWK algorithm

In this algorithm, a user sets K, $\theta_{\max}$ and $\theta_{\min}$ based on his/her privacy requirements and gets real location d_1 using some positioning device (e.g. GPS). Also, if time t 0 and $flag$ = 1, it is the first request of a continuous series of queries; and if t 0 and $flag$ = 0, it is a single query. *Algorithm* 2 describes the pseudo code of SWK algorithm.

As shown in *Algorithm 2*, the SWK algorithm is suitable for the scenario of single request or the first time of continuous request. The input p is represented the historical probability table of location-vertexes, and d_1 is represented user's real location. The $flag$ is set to differentiate between a single request and a continuous request. It is a single request, when $flag = 0$. Otherwise $flag = 1$ represents a continuous request. After determining the scope of anonymous area A (Line 3), we sort the probabilities of locations included in A in ascending order (Line 4). The area contains k locations at least, one of which is must the user's location d_1. This is guaranteed by the value of $\theta_{\max}$ and $\theta_{\min}$. There are three scenarios:

- number $(p_i < p_{d_1}) \geq \frac{k-1}{2}$ && number$(p_i > p_{d_1}) \geq \frac{k-1}{2}$
- number $(p_i < p_{d_1}) < \frac{k-1}{2}$
- number$(p_i > p_{d_1}) < \frac{k-1}{2}$

Algorithm 2: Sliding Window based *k*-anonymity algorithm (SWK)

Input: $p, K, \theta_{max}, \theta_{min}, d_1, t, flag,$
Output: $L = \{d_1, d_2, \ldots, d_k\}$
1: **if** $((t == 0 \&\& flag == 1) \| flag == 0)$
2: Apply for a PID.
3: Compute the scope of anonymous area A by θ_{max}, θ_{min} and d_1.
4: Sort the probabilities in the area A in ascending order and let p_{d_1} is the probability of location d_1.
5: **if** (number $(p_i < p_{d_1}) \geq (k-1)/2$ && number $(p_i > p_{d_1}) \geq (k-1)/2$)
6: Initialize the sliding window to include k probabilities which is the p_{d_1}-centered and slide to both sides and compute anonymity degree D
7: **while** $(D \notin [K-\varepsilon, K])$
8: Slide the window toward right with one step.
9: **if** ($p_{d_1} \notin$ sliding window|| window.size() < K)
10: Failed to generate the sliding window.
11: **break**
12: **else if** (number $(p_i < p_{d_1}) < (k-1)/2$)
13: Reset the sliding window to include the k smaller probabilities and compute anonymity degree D
14: **while** $(D \notin [K-\varepsilon, K])$
15: Slide the window toward right with one step
16: **if** ($p_{d_1} \notin$ sliding window|| window.size() < K)
17: Failed to generate the sliding window.
18: **break**
19: **else if** (number $(p_i > p_{d_1}) < (k-1)/2$)
20: Reset the sliding window to include the k larger probabilities and compute anonymity degree D
21: **while** $(D \notin [K-\varepsilon, K])$
22: Slide the window toward left with one step
23: **if** ($p_{d_1} \notin$ sliding window || window.size() < K)
24: Failed to generate the sliding window.
25: **break**
26: **if** (Failed to generate the sliding window)
27: Modify the value of K, θ_{max} and θ_{min} and submit request again.
28: **else**
29: **Return** $L = \{d_1, d_2, \ldots, d_k\}$.

When the numbers of location probabilities (both larger and smaller than the probability p_{d1} of user's real location) are more than $(k-1)/2$ (Line 5), the initial p_{d1}-centered sliding window extends to both sides until it contains k probabilities of locations (Line 6). If it does not meet the requirement of the anonymity degree, the sliding window slides towards to the right until finding the k candidate probabilities (Line 8).

When the numbers of location probabilities, which are smaller than the probability p_{d1}, are less than $\frac{k-1}{2}$ (Line 12), the initial sliding window contains k probabilities of locations in ascending order (Line 13). If it does not meet the requirement of D, the sliding window is slid towards the right (Line 15).

When the number of location probabilities, which are larger than the probability p_{d1}, are less than $\frac{k-1}{2}$ (Line 19). The initial sliding window contains k location probabilities in descending order (Line 20). If it does not meet the requirement of D, the sliding window is slid towards the left (Line 22).

All three scenarios are aimed at finding k candidate probabilities, corresponding to the candidate locations which exist in the anonymous area A. If the SWK algorithm fails to generate the sliding window, the three scenarios mentioned above follow the same process, i.e, the user will modify the value of K, $\theta_{\max}$ and $\theta_{\min}$ and submit the request again.

Theorem 3 *The time complexity of SWK algorithm is lower than that of the DLS algorithm.*

Proof DLS algorithm is only suitable for preserving the privacy of single request. DLS algorithm chooses $2k$ dummy locations, among which k probabilities of dummy locations are before p_{d1} and k after p_{d1}. Then DLS algorithm selects $k-1$ candidate probabilities form the $2k$ dummy locations. Thus, the DLS algorithm loops C_{2k}^{k-1} times and the time complexity increased exponentially with the growth of k. However, the SWK algorithm, proposed in this work, finds the $k-1$ candidate probabilities through sliding window. According to the experiment in Sect. 6, for successfully getting the right sliding window the SWK algorithm generally repeats more than three times. Thus, the time complexity of SWK algorithm is lower than that of the DLS algorithm. ⊓

The TSM algorithm

When a user sends continuous queries, the $flag$ is set as $flag=1$. If it is not the first request for the continuous queries, the time is $t_i > 0$. In this work, we suppose that the LBS system has memory, and it can remember all the candidate locations selected by users. So we can get k locations $L^r = \{d_{i-1,1}, d_{i-1,2}, \ldots, d_{i-1,j}, \ldots, d_{i-1,k}\}$ at the last query at time t_{i-1}. Except considering that the probability of a user's real trajectory, TSM algorithm will preferentially select the dummy trajectories whose endpoint is in the set L^r of last time request. For example, the location $d_{i-1,j}$ may be selected two times. Figure 8 shows why we select the dummy trajectories whose endpoint may be repeatedly chosen.

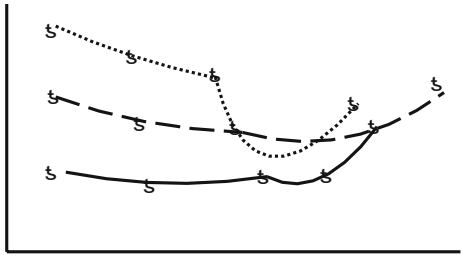


Fig. 8 Repeatedly choose the locations

From Fig. 8, we can analyze that there are more than three trajectories (the black line is the real trajectory). Because of the repeat locations, it can be concluded there are 13 paths. Then the attacker infers the user's real path with a probability 1/13, which is much less than the probability 1/3 (using the traditional k-anonymity for protecting trajectory). So this kind of selecting mechanism greatly avoids the risk of leaking trajectory privacy.

Algorithm 3 shows the pseudo code of the TSM algorithm. The TSM algorithm is suitable for continuous request when the time $t_i > 0$. The input q is represented the historical probability table of trajectory-edge, and d_1 is represented user's real location at t_i. The set L^{rr}, $L^{rr} = \{d_{i-1,1}, \ldots, d_{i-1,k}\}$, denotes the k candidate locations of last query at t_{i-1}. The output is also the k locations, denoted as $L = \{d_1, d_2, \ldots, d_k\}$. After determining the scope of anonymous area A, we sort the probability of all trajectories TR between the candidate locations of the last request L^{rr} and the locations in anonymous area A (Line 3). The set TR contains k edges at least, one of which must be user's real path probability q_1. The TSM algorithm will preferentially select m trajectories whose endpoint is in L^r from the set TR (Line 4–10). So we only find the other $k-1-m$ trajectories except the m selected trajectories and user's real trajectory. Every location of $L^r = \{d_{i-1,1}, d_{i-1,2}, \ldots, d_{i-1,k}\}$, except from the $m+1$ selected locations, has a trajectory set tr_j with the endpoint in the anonymous area A. We store the probability of trajectory-edge tr_j into the corresponding storage $str(q_i)$ (Line 14). Then we randomly choose one probability of trajectory-edge from the each storage $str(q_i)$ every time, and get $k-1$-m probabilities (Line 17–18). Finally, we compute entropy for the k selected locations. If it meets the requirement of anonymity degree D (Line 19), the request is successful. Otherwise, we reset the time $t=0$ and back to the process of *Algorithm 2* (Line 20). This process can protect user's trajectory privacy efficiently. And the malicious attacker infers the user real location/trajectory with a lower probability.

To realize *k*-anonymity and maximum entropy, TSM algorithm select $k-1$ dummy trajectories, whose probability of trajectory-edge is similar to the probability of the user's real trajectory. Furthermore, the *k* trajectories selected by TSM algorithm may have some crossing points. The LP may believe that there are more than *k* trajectories. So the TSM algorithm can effectively protect trajectory privacy. For each request of continuous queries, TSM algorithm also achieves *k*-anonymity and protects location privacy of user.

Algorithm 3: Trajectory Select Mechanism (TSM)

Input: $q, K, \theta_{max}, \theta_{min}, d_1, t, flag,$
$L' = \{d_{i-1,1}, d_{i-1,2}, \dots, d_{i-1,k}\}$

Output: $L = \{d_1, d_2, \dots, d_k\}$

1: **if** ($t_i > 0$ && $flag$ ==1)
2: Determine the scope of anonymous area A by the θ_{max}, θ_{min} and d_1.
3: Sort the probability of all trajectories TR between the candidate locations of the last request L' and the locations in the anonymous area A in ascending order.
4: **if (**the location is included in $L$and in area A**)**
5: $L.prior = 1$
6: **end if**
7: Find the user's real path probability q_1 marking the historical probability table of trajectory-edge q.
8: **if** (the $L.prior = 1$of endpoint of TR &&the probability of edge is similar with q_1)
9: Choose the edges as the candidate trajectory.
10: Let the corresponding m positions as the part of output $d_2', d_3', \dots, d_{m+1}'$.
11: **for** ($d_{i-1,2} : d_{i-1,k}$)
12: **if** ($d_{i-1,i} \notin$ {the endpoint of m trajectories})
13: Find all the trajectories between $d_{i-1,i}$ and the location $l_i \notin d_1, d_2, \dots, d_{m+1}$ in the area A.
14: Store the probability of the trajectories in $str(q_i)$.
15: Let Q = 0, where Q represents the set of k selected probability of trajectory.
16: **for** ($str(q_i)$)
17: Select a probability from $str(q_i)$ and the number is *k*-1-*m*.
18: Compute the D of the *k*-1-*m* probabilities, the probability of m trajectories, and user's real q_1.
19: **if** ($D \in [K - \varepsilon, K]$)
20: Q = {*k* probabilities of trajectory-edge}.
21: break;
22: **if** (Q)
23: Return L (the corresponding endpoint of the *k* trajectories).
24: **else**
25: Reset $t = 0$ and submit request again.

6 Simulation results and analysis

For evaluating and the effectiveness of our proposed *k*-anonymity trajectory algorithm (KAT), we have conducted extensive simulations. In this section, we first describe the simulation environment, and then present the simulation results and analysis.

Simulation environment

We first construct a graph to represent a large real geographic area. The horizontal axis of the graph represents the latitude of the geographic location and the vertical axis denotes the longitude. We then divide the graph into 10×10 small cells. Label each cell with a ordered pair (i, p_i), where i represents the location and p_i is the corresponding historical location probability. As described before, the historical probability must meet two constraints:

$$0 < p_i < 1 \tag{10}$$

$$\sum_{i=1}^{100} p_i = 1 \tag{11}$$

To simulate the roads in the real geographic area we randomly add edges q_{ij} between these locations. The triple i, j, q_{ij} denotes the historical trajectory probability between location i and location j. The probability q_{ij} should meet the following two conditions.

$$0 \le q_{ij} < 1 \tag{12}$$

$$\sum_{j=1}^{100} \sum_{i=1}^{100} q_{ij} = 1 \tag{13}$$

Finally, we construct the undirected graph $G(V, E)$, where V is the set of points, which are the center locations of all of the cells and E is the set of edges in the graph.

In our simulations, we compare the performance of our proposed KAT algorithm and the DLS algorithm proposed in [12] under the scenarios of single request and continuous request.

Simulation results

In our experiment, we assume that a user is in the location $i = 20$ and sends a single query. Table 1 shows the comparison for entropies of the KAT, DLS and MAXS (we use MAXS to denote the maximum entropy defined in Sect. 4.1). From Table 1, we can see that our KAT algorithm can achieve *k*-anonymity in single query. Because the entropies of DLS and KAT are identical when the privacy degree K varies from 3 to 5, it means that they may select the same locations as the candidate location. With the increasing of K, the entropy of algorithm KAT is a little less than DLS. The entropy gained

Table 1 Comparison for the entropies of MAXS, DLS and KAT

Entropy	Privacy degree K							
	3	4	5	6	7	8	9	10
MAXS	1.58496	2.00000	2.32190	2.58500	2.80740	3.00000	3.16990	3.32190
DLS	1.58435	1.99899	2.31964	2.58226	2.80409	2.99535	3.16402	3.31354
KAT	1.58435	1.99899	2.31964	2.57885	2.80033	2.98704	3.15548	3.30315

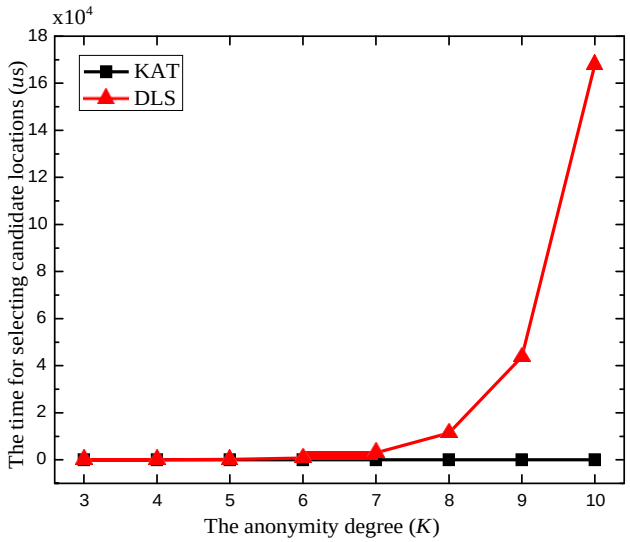


Fig. 9 The time consumption for single query

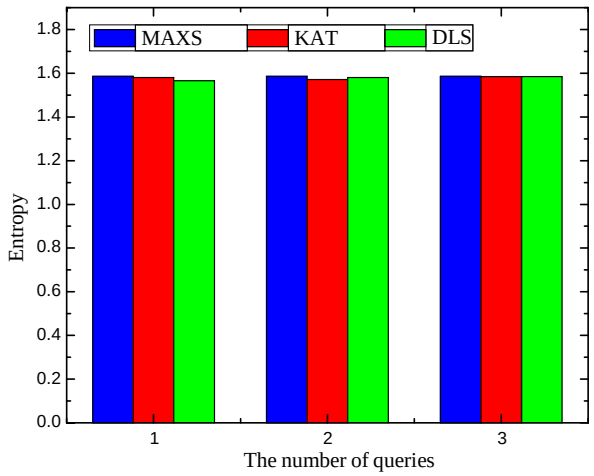


Fig. 10 Comparison on entropies for continuous queries

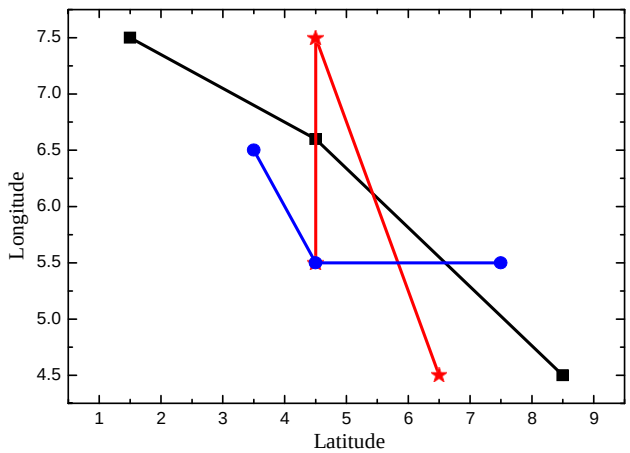


Fig. 11 The trajectory map of KAT algorithm

from both DLS and KAT algorithms are very closed to the maximal value (e.g., MAXS).

Figure 9 shows the time complexity of the KAT and DLS algorithms with the different K in a single query. We can see that the time consumptions of the KAT algorithm and DLS algorithm are very closed to each other when the privacy degree $K < 7$. However, the time for selecting candidate locations grows exponentially when $K > 7$, whereas the time consumed by KAT algorithm is stable. Therefore, the KAT algorithm greatly reduces the time complexity compared to DLS.

Figure 10 shows the simulation results comparing entropy of the compared algorithms. In this set of simulations, we also assume that the user is in the location $i = 20$ before sending continuous queries. In our experiments, the user submits three queries in total with the privacy degree $k = 3$. Accordingly the KAT algorithm selects 3 candidate locations ($L = \{d_{i1=20}, d_{i2=33}, d_{i3=44}\}$) at time t_0. At time t_1 the candidate locations denoted as $L_1 = \{d_{i1=34}, d_{i2=44}, d_{i3=24}\}$. The location $i = 44$ is selected two times, because the location is in the both anonymous areas of the two queries. At time t_2 the candidate locations are the $L_2 = \{d_{i1=58}, d_{i2=47}, d_{i3=56}\}$. The entropies for KAT and DLS algorithm are both closed to the maximal entropy (MAXS). Thus, we can see that each continuous query request gets the near maximal entropy and also achieves k-anonymity in our proposed KAT algorithm.

Figure 11 shows the trajectory achieved by the KAT algorithm. The black line is the user's real trajectory, and the other two are the dummy trajectories selected by the KAT algorithm. Because of the intersection point $i = 44$ $(x = 4.5, y = 5.5)$, we can get 5 trajectories. Thus, the malicious attacker may infer the user's real trajectory with probability of 1/5 rather than the probability of 1/3.

Figure 12 shows the trajectory achieved by the DLS algorithm. The black line represents the user's real trajectory and

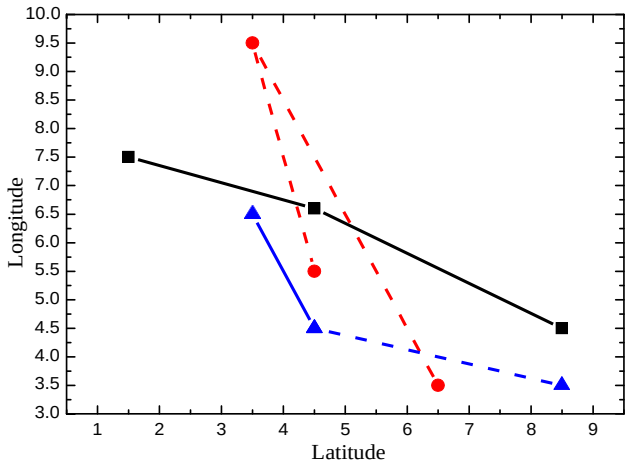


Fig. 12 The trajectory map of DLS

the other two are the dummy trajectory selected by the DLS algorithm. The dashed line implies that there are no roads in the geographic map. Figure 12 shows that there are two trajectories that have no roads. So the attacker can easily filter out the dummy trajectories (no road) and infer user's real trajectory. By comparing the results shown in Figs. 11 and 12, we can see that our proposed KAT algorithm can better protect trajectory privacy than the existing DLS algorithm for continuous LBS requests.

Figure 13 illustrates the simulation results comparing the time complexity of the KAT and DLS algorithms under different privacy degree k (k = 3, 5, 7, 9) with continuous queries. For the DLS algorithm, the time for selecting k candidate trajectories between each request of continuous query is stable and increases quickly with the growth of the value of k. On the other hand the time for selecting the k trajectories increases slowly in the KAT algorithm. Thus, the KAT algorithm is more time efficient than the DLS algorithm in the continuous query scenario.

7 Discussion

In this paper, we propose the k-anonymity trajectory algorithm (KAT) for preserving privacy for the continuous query scenarios in IoT-cloud systems, rather than protecting the user's privacy single query scenario in our previous work [41–43]. KAT is superior to the dummy location selection (DLS) algorithm proposed in [12], as shown in Fig. 9. The DLS algorithm is time consuming in selecting other k 1 candidate locations form the 2k dummy locations. In contrast, the time complexity of KAT algorithm is substantially

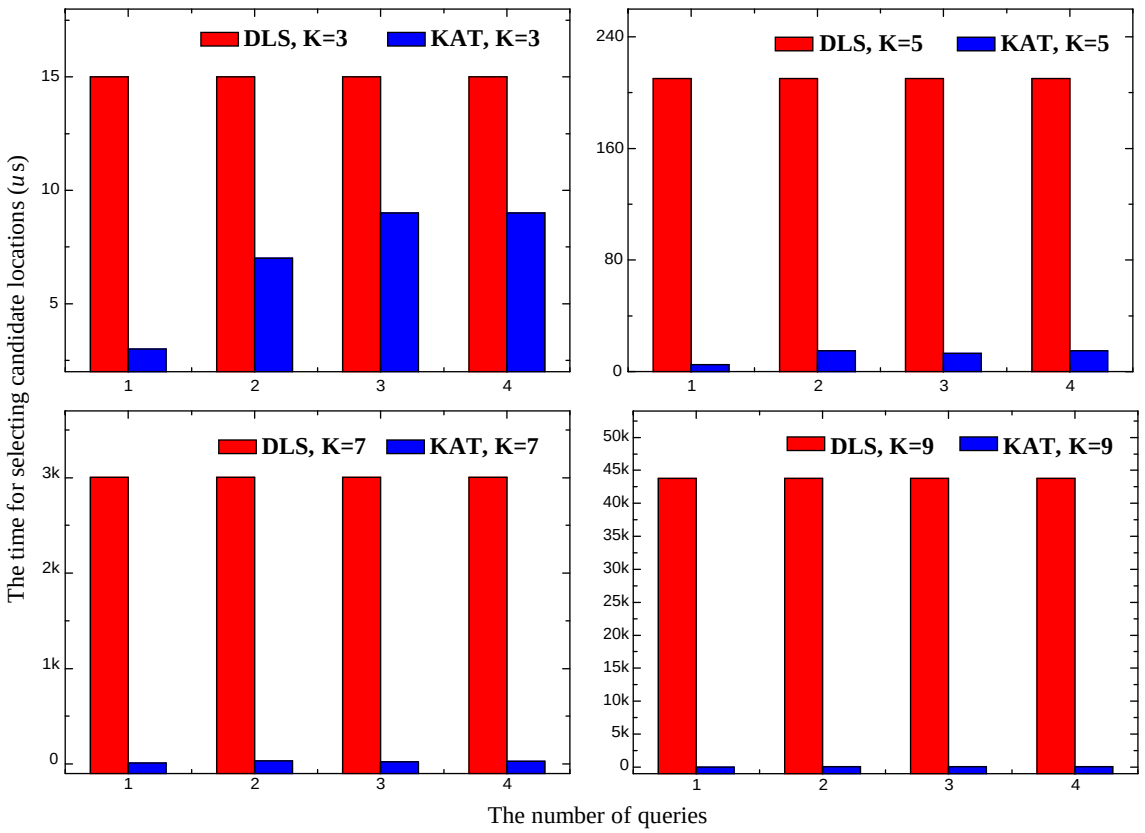


Fig. 13 The time consumption in continuous query

reduced by collecting other k_1 candidate locations by using the *sliding window* based *k*-anonymity mechanism.

To protect trajectory privacy, we introduce the *maximum entropy* and the *trajectory select mechanism* into our KAT algorithm for choosing other k_1 dummy trajectories to resist attacks in the continuous query. We validated the trajectory anonymity through simulations in Figs. 11 and 12. It is evident that the malicious LP can easily infer the actual customer trajectory if the traditional DLS algorithm is used. It is also evident that the time consumption for the proposed KAT algorithm in continuous query is far less than the traditional DLS algorithm for peer anonymity.

8 Conclusion

This paper has addressed privacy issues in location-based services for IoT-cloud systems and also identified a list of research issues in privacy. In this paper we have studied the problem of protecting trajectory and location privacy in LBS. We considered three attack models where a user applies traditional *k*-anonymity technology for trajectory privacy preserving and the LP may be malicious attacker. We design an efficient *k*-anonymity trajectory algorithm for preserving users' location privacy in single query and trajectory privacy in continuous queries. In our proposed algorithm, we design the *k* sliding window for selecting the dummy locations and the Trajectory Select Mechanism (TSM) for selecting the dummy trajectories. Simulation results show that our algorithm reduces the time complexity compared to the existing solutions for single query and effectively preserves users' trajectory privacy for continuous queries.

References

1. Truong, H., Dustdar, S.: Principles for engineering IoT cloud systems. IEEE Cloud Comput. **2**(2), 68–76 (2015)
2. Barcelo, M., Correa, A., Llorca, J., et al.: IoT-cloud service optimization in next generation smart environments. IEEE J. Sel. Areas Commun. **34**(12), 4077–4090 (2016)
3. Kalmar, E., Kertesz, A., Varadi, S., et al.: Legal and regulative aspects of IoT cloud systems. In: IEEE International Conference on Future Internet of Things and Cloud Workshops, pp. 15–20 (2016)
4. Cai, H., Xu, B., Jiang, L., et al.: IoT-based big data storage systems in cloud computing: perspectives and challenges. IEEE Internet Things J. **387**, 75–87 (2016)
5. Botta, A., de Donato, W., Persico, V., et al.: Integration of cloud computing and internet of things: a survey. Future Gener. Comput. Syst. **56**, 684–700 (2016)
6. Landwehr, C.: Privacy research directions. Commun. ACM **59**(2), 29–31 (2016)
7. Xin, M., Lu, M., Li, W.: An adaptive collaboration evaluation model and its algorithm oriented to multi-domain location-based services. Expert Syst. Appl. **42**, 2798–2807 (2015)
8. Wang, Y., Xu, D., He, X., et al.: L2p2: location aware location privacy protection for location-based services. In: IEEE INFOCOM, 1996- 2004 (2012)
9. Li, Y., Yiu, M.: Route-saver: leveraging route APIs for accurate and efficient query processing at location-based services. IEEE Trans. Knowl. Data Eng. **27**(1), 235–249 (2015)
10. Schlegel, R., Chow, C., Huang, Q., et al.: User-defined privacy grid system for continuous location-based services. IEEE Trans. Mob. Comput. **14**(10), 2158–2172 (2015)
11. Zhou, T.: Understanding user adoption of location-based services from a dual perspective of enablers and inhibitors. Inf. Syst. Front. **17**(2), 413–422 (2015)
12. Niu, B., Li, Q., Zhu, X., et al.: Achieving K-anonymity in privacy-aware location-based services. In: IEEE INFOCOM, pp. 754–762 (2014)
13. Niu, B., Li, Q., Zhu, X., et al.: Enhancing privacy through caching in location-based services. In: IEEE INFOCOM (2015)
14. Liu, X., Zhao, H., Pan, M., et al.: Traffic aware multiple mix zone placement for protecting location privacy. In: IEEE INFOCOM, pp. 972–980 (2012)
15. Zhu, X., Chi, H., Niu, B., et al.: Mobi Cache: when k-anonymity meets cache. In: IEEE GLOBECOM, pp. 820–825 (2013)
16. Shu, T., Chen, Y., Yang, J.: Multi-lateral privacy preserving localization in pervasive environments. In: IEEE INFOCOM, pp. 2319–2327 (2014)
17. Li, H., Sun, L., Zhu, H., et al.: Achieving privacy preservation in WiFi fingerprint-based localization. In: IEEE INFOCOM, pp. 2337–2345 (2014)
18. Beresford, A., Stajano, F.: Mix zones: user privacy in location-aware services. In: The Second IEEE Conference on Pervasive Computing and Communications Workshops, pp. 127–131 (2004)
19. Liu, X., Liu, K., Guo, L., et al.: A game-theoretic approach for achieving k-anonymity in location based services. In: IEEE INFCOM, pp. 2985–2993 (2013)
20. Sweeney, L.: k-anonymity: a model for protecting privacy. Int. J. Uncertain. Fuzziness Knowl. Based Syst. **10**(5), 557–570 (2002)
21. Yang, D., Fang, X., Xue, G.: Truthful incentive mechanisms for k-anonymity location privacy. In: IEEE INFOCOM, pp. 2994–3002 (2013)
22. Sharma, V., Shen, C.: Evaluation of an entropy-based k-anonymity model for location based services. In: International Conference on Computing, Networking and Communications (ICNC), pp. 374–378 (2015)
23. Shokri, R., Troncoso, C., Diaz, C.: Unraveling an old cloak: K-anonymity for location privacy. In: ACM Workshop on Privacy in the Electronic Society, pp. 115–118 (2010)
24. Lu, R., Lin, X., Shi, Z., et al.: PLAM: a privacy-preserving framework for local-area mobile social networks. In: IEEE INFOCOM, pp. 763–771 (2014)
25. Li, X., Miao, M., Liu, H., et al.: An incentive mechanism for k-anonymity in LBS privacy protection based on credit mechanism. Soft Computing (2016). 10.1007/s00500-016-2040-2
26. Caballero-Gil, C., Molina-Gil, J., Hernández-Serrano, J., et al.: Providing k-anonymity and revocation in ubiquitous VANETs. Ad Hoc Netw. **36**, 482–494 (2016)
27. Andrews, M., Wilfong, G., Zhang, L.: Analysis of k-anonymity algorithms for streaming location data. In: IEEE Conference on Computer Communications Workshops, pp. 1–6 (2015)
28. Gedik, B., liu, L.: Protecting location privacy with personalized k-anonymity: architecture and algorithms. IEEE Mob. Comput. **7**, 1–18 (2007)

29. Buttyan, L., Holczer, T.: A practical pseudonym changing scheme for location privacy in VANETs. In: IEEE Vehicular Networking Conference, pp. 1–8 (2009)
30. Du, J., Xu, J.: IPDA: supporting privacy-preserving location-based mobile services. In: International Conference Mobile Data Management, pp. 212–214 (2007)
31. Yao, L., Lin, C., Liu, G., et al.: Location anonymity based on fake queries in continuous location-based services. In: The 7th International Conference on Availability, Reliability and Security, pp. 375–382 (2012)
32. Feng, Y., Liu, P., Zhang, J.: A mobile terminal based trajectory preserving strategy for continuous querying LBS users. In: IEEE International Conference on Distributed Computing in Sensor Systems, pp. 92–98 (2012)
33. Mohammed, N., Fung, B., Debbabi, M.: Walking in the crowd, anonymizing trajectory data for pattern analysis. In: The 18th ACM Conference on Information and Knowledge, pp. 1441–1444 (2009)
34. Xu, T., Cai, Y.: Exploring historical location data for anonymity preservation in location-based services. In: IEEE INFOCOM, pp. 547–555 (2008)
35. Yigitoglu, E., Luisa, M.: Privacy-preserving sharing of sensitive semantic locations under road-network constraints. In: IEEE 13th International Conference on Mobile Data Management, pp. 186–195 (2012)
36. Chow, C., Mokebel, M.F.: Trajectory privacy in location-based services and data publications. ACM SIGKDD Explor. Newsl. **13**(1), 19–29 (2011)
37. Schmid, F., Richter, K.: Semantic trajectory compression. Advances in spatial and temporal databases. In: Lecture Notes in Computer Science, pp. 411–416 (2009)
38. Peng, T., Liu, Q., Meng, D., et al.: Collaborative trajectory privacy preserving scheme in location-based services. Inf. Sci. **387**, 165–179 (2016)
39. Li, X., Wang, E., Yang, W., et al.: DALP: a demand-aware location privacy protection scheme in continuous location-based services. Concurr. Comput. Pract. Exp. **28**(4), 1219–1236 (2016)
40. Niu, B., Gao, S., Li, F., et al.: Protection of location privacy in continuous LBSs against adversaries with background information. In: IEEE International Conference on Computing, Networking and Communications, pp. 1–6 (2016)
41. Sun, G., Liao, D., Li, H., et al.: L2P2: a location-label based approach for privacy preserving in LBS. Future Gener. Comput. Syst. **74**, 375–384 (2016)
42. Sun, G., Chang, V., Ramachandran, M., et al.: Efficient location privacy algorithm for Internet of Things (IoT) services and applications. J. Netw. Comput. Appl. **89**, 3–3 (2016)
43. Sun, G., Xie, Y., Liao, D., et al.: User-defined privacy location-sharing system in mobile online social networks. J. Netw. Comput. Appl. **86**, 34–45 (2017)