Collaborative trajectory privacy preserving scheme in location-based services

Tao Peng\textsuperscript{a}, Qin Liu\textsuperscript{b}, Dacheng Meng\textsuperscript{a}, Guojun Wang\textsuperscript{c, a, *}

\textsuperscript{a} School of Information Science and Engineering, Central South University, Changsha, Hunan Province, 410083, P. R. China
\textsuperscript{b} School of Information Science and Engineering, Hunan University, Changsha, Hunan Province, 410082, P. R. China
\textsuperscript{c} School of Computer Science and Educational Software, Guangzhou University, Guangzhou, Guangdong Province, 510006, P. R. China

\textbf{A R T I C L E   I N F O}

\textbf{Article history:}
Received 24 December 2015
Revised 16 July 2016
Accepted 5 August 2016
Available online 5 August 2016

\textbf{Keywords:}
K-anonymity
Location-based service (LBS)
Location privacy
Trajectory privacy preserving
Continuous query

\textbf{A B S T R A C T}

Location-based services (LBSs) have been gaining considerable popularity and are becoming the fastest growing activity-related services that people use in their daily life. While users benefit from LBSs, the collection and analysis of participants’ location data and trajectory information may jeopardize their privacy. Existing proposals focus mostly on snapshot queries. However, privacy preservation in continuous LBSs is more challenging than in snapshot queries because adversaries could use the spatial and temporal correlations on the user trajectory to infer the user’s private information. In this paper, we propose the collaborative trajectory privacy preserving (CTPP) scheme for continuous queries, in which trajectory privacy is guaranteed by caching-aware collaboration between users, without the need for any fully trusted entities. The main idea of our scheme is to obfuscate the actual trajectory of a user by issuing fake queries to confuse the LBS adversary. We first present a multi-hop caching-aware cloaking algorithm to collect valuable information from multi-hop peers based on collaborative caching. Then, we describe a collaborative privacy preserving querying algorithm that issues a fake query to confuse the location service provider (LSP). Extensive experimental results verify the effectiveness and efficiency of our scheme in terms of processing time and communication cost.

© 2016 Elsevier Inc. All rights reserved.

1. Introduction

By virtue of the rapid advances in the development of modern smart devices, wireless communication and positioning technologies, over the past decade location-based services (LBSs) have been gaining very considerable popularity and are becoming the fastest growing activity-related services utilized by mobile device users. Users carrying mobile devices loaded with positioning capabilities, e.g., GPS, are able to query location service providers (LSPs) and enjoy corresponding service data [17]. Typical examples of these data include navigation information and live traffic reports, the nearest restaurant providing the user’s favourite cuisine, coupons from a nearby market, location-based advertisements and so on. Since the LSP is potentially untrustworthy, the submitted queries may lead to some sensitive information about a user being revealed, such as his/her real identity, exact location and queried interest. For instance an adversary who has compromised the LSP may analyse the data and track a user’s trajectory to infer his/her identity [16]. LBSs offer both opportunities and challenges for users and LSPs. The main problem lies in the preservation of the users’ privacy when they utilize the LBSs.

* Corresponding author.
E-mail address: csgjwang@gmail.com (G. Wang).
Among the concerns about LBSs that need to be addressed, trajectory privacy preservation [4] is the most salient because of the vulnerability of the spatial and temporal information contained in the continuous queries received by the LSP, which may expose users’ whereabouts and other personal private information. However, the traditional privacy preservation approaches in snapshot LBSs, e.g., the \( K \)-anonymity principle [13], may suffer a correlation attack [8,14], which prevents them from protecting the privacy of continuous queries on a user trajectory. Fig. 1(a) demonstrates how this is possible. The stars on the trajectory represent a series of query points of the user at different time points \( (t_1, t_2, t_3, t_4) \). As the user moves, he/she blurs the exact locations into \( K \)-anonymizing spatial regions (\( K \)-ASRs) that include at least \( K-1 \) other users, i.e., \( K = 5 \). When the adversary can obtain all the sequential submitted queries and analyse their spatial and temporal correlations, he/she can infer that a user who appears in all the cloaking regions may potentially be the query issuer. The adversary can also reconstruct the trajectory of the user according to the sequence of the query issuing time. We can see that in this scenario, although each snapshot query is protected based on the \( K \)-anonymity principle, the trajectory of the user may be easily disclosed.

Some approaches based on centralized architecture have been proposed for preserving trajectory privacy in the case of continuous queries. In these approaches a trusted third party (TTP), called the Anonymizer, is introduced into the system, acting as an intermediate tier between the users and the LSPs. In some research studies the location \( K \)-anonymity principle in snapshot LBS was extended, expanding the initial ASR to a region that includes the same \( K \) users in all continuous queries. However, the built cloaked area in these approaches becomes excessively large as the \( K \) users move in different directions. Hwang et al. [14] proposed a time-obfuscated approach to avoid this problem. The main idea is to issue redundant queries with \( K \)-ASRs and randomize the sequence of the query time in order to protect trajectory privacy. This method had some limits. (i) It selects other similar \( r-1 \) trajectories from the trusted server’s database for issuing redundant queries, and the greatly increased number of queries incurs considerable additional overheads. (ii) It is not sufficiently flexible to handle the case where the user dynamically changes his/her route, since this method is based on the predicted trajectory generated by the user-predefined start and end points of the journey. Moreover, similarly to other TTP models, these techniques inherit the drawback of the centralized architecture according to which the TTP stores the exact location information of all users. It thus becomes an attractive target for an attacker and the user information is jeopardized when it is attacked by an adversary. Furthermore the TTP constitutes the central point of failure and the performance bottleneck, since all the submitted queries have to pass through it.

Inspired by the scheme presented in [14] we propose the collaborative trajectory privacy preserving (CTPP) scheme, which spatiotemporally breaks the correlations of continuous LBS queries to obfuscate the user’s actual trajectory. As shown in the Fig. 1(b), by disturbing the query time and shifting the anonymity regions, our method prevents an adversary from tracking or reconstructing a user trajectory. However, unlike that presented in [14], which requires a TTP to generate redundant cloaking regions, our CTPP scheme constructs the \( K \)-ASRs based on the gathered information obtained from multi-hop peers and the trajectory privacy is guaranteed by user collaboration. We designed a multi-hop caching-aware cloaking (MCC) algorithm to share desired data received from multi-hop peers with which the user can construct a cloaking area and locally obtain the results for future queries. Notice that the value of the hop distance in MCC can be adjusted dynamically by the user. If the user’s requirement can be satisfied directly by the gathered service data, the user can issue a fake query to
confuse the adversary. As compared with that of [14], our scheme is more flexible, since it needs only the next location of the user’s track and not the entire predicted trajectory. Moreover, it can be expected to be more efficient as some of the user LBS queries can be locally solved rather than submitted to the remote LSP. Our CTPP scheme protects users’ location privacy by the following means. (i) Users’ collaborative caching can reduce the number of queries sent to the LSP and thus reduce the amount of private information exposed to the server. (ii) The obfuscated and fake queries can confuse the adversary’s attempt to identify the issuer or to reconstruct the user trajectory. (iii) The privacy preservation of queries sent to the LSP is guaranteed by the K-anonymity principle. The main contributions of this paper are as follows.

1. We describe the development of a novel framework for protecting trajectory privacy in continuous LBSs, eliminating the traditional TTP and without changing the structures of the current LBSs. To the best of our knowledge the CTPP scheme is the first user-collaboration technique to provide trajectory privacy for continuous LBSs using non-centralized architectures.

2. We propose an MCC algorithm, which utilizes the shared information received from multi-hop peers to form a K-ASR and to find the answer of the LBS.

3. We design a collaborative privacy preserving querying (CPPQ) algorithm, which enables the user to issue a fake query to confuse the LSP or adversary.

The remainder of this paper is organized as follows. In Section 2 we present the proposed system architecture of the CTPP scheme. In Section 3 we describe the motivation and overview of our scheme. Then we present the MCC algorithm and CPPQ algorithm in detail in Section 4. In Section 5 we describe a set of simulations used to evaluate the effectiveness of our proposal. In Section 6, we present the related work. Finally, we conclude this paper and present future work in Section 7.

2. System architecture and assumption

In this section, we first present the system architecture of our CTPP scheme and then provide the attack model and security requirements.

2.1. System architecture

Fig. 2 illustrates the system architecture of the CTPP scheme. We employ two roles, the mobile user and the LSP, in our system.

Mobile users: Mobile users carry mobile devices that have positioning capabilities, e.g., GPS, to determine their current location information. Users can communicate with the LSP through a cellular network, e.g., 3G/4G, or WiFi access points. Moreover they also have the ability to communicate with other mobile peers by a short-range communication system, e.g., WiFi or Bluetooth. They can form a collaborative group, such as a mobile P2P network, via wireless transmission protocols or ad hoc network routing protocols [27,28]. Mobile users participate in the system collaboratively and share the information in their caches with each other to protect their privacy.

LSPs: LSPs are online location-based service providers, for instance, Google Maps or Yelp!. LSPs deploy location-based database servers storing map resources and the information of some particular types of points of interest (POIs), e.g., nearby restaurants and gas stations, and also other location-related data, e.g., traffic reports and market coupons. LSPs process users’ snapshots or continuous queries based on an exact location or a cloaked region, searching for the answers in their database, and then provide the location-based information in which the users are interested, such as the POIs within a certain area.
2.2. Assumption

\((K, T_u, H_{max})\)-User privacy profile. In order to protect privacy in an LBS, each mobile user can specify three parameters to accommodate personalized privacy requirements. Users are able to arbitrarily change the settings of their privacy profiles at any time.

- Anonymization degree, \(K\): This setting indicates the anonymity level when the location \(K\)-anonymity cloaking area is generated, which should include at least \(K\) different users. In general the larger the value of the \(K\), the more privacy is provided.
- Temporal threshold, \(T_u\): This setting determines the freshness of the stored data. Users define a time threshold, \(T_u\), to specify the length of the timeout for each record in their cache. When the dwell time of a record has exceeded \(T_u\), it is deleted.
- Maximum hop distance, \(H_{max}\): This setting specifies the maximum tolerable value of hop distance from peers to the user. The larger is the value of \(H_{max}\), the wider is the coverage area of the peers; the time delay of the user receiving a response from other peers also increases with the hop distance.

Knowledge of attackers. When the LSP receives a user-submitted query it has all the knowledge about the user’s LBS requirements, which include the user’s identity, the sequence of ASRs, the type of POI etc. We assume that the attacker has the following capabilities.

- The ability to collect all the user’s anonymity regions and their time points when the queries are received.
- The ability to observe the \(K\) users in the anonymity region, without knowing which location belongs to whom.
- The ability to learn the algorithms that are used to offer privacy in the LBSs because the common algorithms are typically publicly available.

Attack model. An adversary can be the owner of LBS servers that collect all the users’ information and may release some sensitive data to other third parties. Alternatively the adversary may be able to compromise and control the LSP, gaining access to the user data stored on it. We assume that the attacker attempts to breach the trajectory privacy of users using snapshot or continuous LBS data.

- In the snapshot LBS the adversary analyses a single snapshot LBS query of a user to infer more information about the position or the identity that the user intended to hide.
- In a continuous LBS the adversary knows a set of multiple or series sequence queries collected over time. We consider the adversary successfully intrudes on the trajectory privacy of a user if he/she can reconstruct the entire trajectory from continuous LBS queries and associate a user’s real identity with the corresponding trajectory.

In our threat model we do not address the threat that observers may sniff the communication channels to infer users’ sensitive information, as such a threat can be alleviated by some of the existing security schemes, e.g., secure sockets layer (SSL), and conventional solutions, e.g., cryptography and hashing [26,30,31]. We assume that mobile users are trusted, which means that they neither provide malicious information to their peers nor use the gathered data obtained from peers to attack our system. This assumption has also been made in previous decentralized architectures in LBSs developed by other researchers (see, e.g., [7,19,22]), in which the mobile users protect their privacy in a trusted and collaborative manner.

3. Motivation and system overview

Suppose the user \(u\), attempting to search for the nearest clinic on the road where he/she is located, submits a series of continuous queries, which include the identity, location and interest, to the untrustworthy LSP in order to enjoy the services of the LBS. The user wants to obtain his/her desired information and, at the same time, preserve his/her trajectory and location privacy. Under the principle of traditional \(K\)-anonymity the user blurs his/her exact locations into a sequence of ASRs (assuming \(K = 3\) in the recurrent queries. We assume that the set of users inside the ASRs and their time points are as shown in Table 1. The attacker has the ability to access all the data on the LSP and extract the information corresponding to a specific user. The attacker can then easily infer that the user, \(u\), who appears in all the cloaking regions is potentially the query issuer. The attacker can further reconstruct the user trajectory by correlating the spatial and temporal information in

<table>
<thead>
<tr>
<th>Time</th>
<th>ASR</th>
<th>User set</th>
</tr>
</thead>
<tbody>
<tr>
<td>(t_1)</td>
<td>(R_1)</td>
<td>({a, b, u})</td>
</tr>
<tr>
<td>(t_2)</td>
<td>(R_2)</td>
<td>({b, c, u})</td>
</tr>
<tr>
<td>(t_3)</td>
<td>(R_3)</td>
<td>({d, e, u})</td>
</tr>
</tbody>
</table>
the ASRs. As we can see, in this scenario the traditional K-anonymity approach does not suffice to prevent trajectory and location privacy breaches. Therefore a secure privacy preservation scheme is imperative to protect the sensitive information of each user when utilizing a continuous LBS. The scheme is also supposed to consider the issuing time of queries and the correlation of the sequential ASRs in the continuous LBS in order to protect user trajectory privacy. Some existing approaches are based on a centralized architecture. However these TTP-based methods have several weaknesses, as described in Section 1. The preservation of user privacy in continuous LBSs and the elimination of the TTP has become a problem. In this paper we propose a user-centric K-anonymity privacy-preserving scheme for both snapshot and continuous queries. In our scheme user location and trajectory privacy is guaranteed by caching-aware collaborations between users rather than by the participation of a TTP. We first present the MCC algorithm, which causes the user to collect valuable information from multi-hop peers and attempt to fulfil his/her requirement locally. Then we describe the design of the CPPQ algorithm, which enables the user to issue either a fake query to confuse the adversary or a real query while privacy is preserved.

Fig. 3 shows an example of our scheme. The red stars (L1–L6) on the trajectory indicate the footprints of the user, \( u \); black solid dots represent the positions of other mobile peers. The peer shares the requested ASR and corresponding anonymity sets with \( u \), which are represented by circles and dots within the circles. This figure illustrates that the continuous queries of the user are issued together with the cloaked regions \((R_1 – R_6)\) from six positions against the original moving direction.

**Step 1:** At the beginning of the journey, the user \( u \) performs the MCC algorithm to gather valuable information from his/her peers. \( u \) first broadcasts a message to seek cooperation. Assume that his/her 1-hop neighbours \( a, b, c, d \) receive requests from \( u \) and share their valid information with \( u \), which includes the required position of a POI, the K-ASR and other information related to the LBS. (Black circles represent the shared ASRs from these 1-hop neighbours’ cache).

**Step 2:** The user \( u \) gathers information from these 1-hop peers and checks whether his/her next location, \( L_2 \), is included in the gathered ASRs. If it is, the user \( u \) directly proceeds to Step 3. If not, he/she refers to the 2-hop peers or multi-hop peers, until the hop count has increased to the maximum value, \( H_{max} \), and then proceeds to Step 3. In Fig. 3, the red circles represent shared ASRs from the user’s 2-hop peers \( e, f, g \) and \( h \); \( u \)’s next location, \( L_2 \), is located in the anonymity region received from peer \( g \).

**Step 3:** \( u \) selects \( K-1 \) nearest points from the gathered points to generate his/her K-ASR by using the minimum bounding rectangle (MBR) [11] (represented by dashed rectangle \( R_1 \), assuming \( K = 4 \)). Using \( R_1 \), \( u \) submits his/her query to the LSP.

**Step 4:** When \( u \) travels to location \( L_2 \), he/she first searches the LBS answer locally before querying the LSP. The user traverses the stored data and finds that \( L_2 \) is in the area of the ASR received from 2-hop peer \( g \). Thus \( u \) can obtain the service data of his/her LBS requirement from the stored record received from \( g \) instead of querying the LSP.

**Step 5:** In order to obfuscate his/her trajectory and confuse the adversary, \( u \) selects a dummy ASR, e.g., \( R_2 \), from his/her cache to issue a fake query.

**Step 6:** A similar situation occurs when \( u \) arrives at \( L_3 \). He/she traverses the stored data in his/her cache and finds \( L_3 \) is covered by the ASR received from the 2-hop peer \( f \). Thus, \( u \) can enjoy the stored service data directly from this cache and issue a fake query using the dummy cloaking area \( R_3 \).

**Step 7:** If the user cannot locally settle his/her LBS request, for instance, when \( u \) moves to point \( L_4 \), he/she should perform the MCC algorithm again to pursue collaboration from peers and then issue a query together with \( R_4 \).
Step 8: In the shared valid information of Step 7, the ASRs received from peers $i$ and $j$ cover points L5 and L6. Thus when $u$ moves to L5 and L6 he/she can locally obtain the LBS answer and issues fake queries with $R_5$ and $R_6$ in the same fashion.

This figure shows that the user can search for the POIs locally, if his/her current location is included in the ASRs stored in his/her cache. The user trajectory is completely obfuscated because of the fake ASRs (dashed rectangles $R_3$, $R_4$, $R_5$, $R_6$). Our CTPP scheme not only satisfies the user’s K-anonymity requirement for each snapshot query but also uses the obfuscating technique to protect user trajectory privacy in continuous LBSs.

### 4. Collaborative trajectory privacy-preserving scheme

In our scheme, each user holds a cache in which to store the information received from the LSP or other peers. The data are recorded in the format $\{AS, POI, T, ID, \bar{h}\}$, where AS indicates the ASR and the anonymity set in it, POI is the point of interest, $T$ is the timestamp, ID is the identity of the peer who shares the data with the user, $\bar{h}$ is the hop value of the data sharing user. Specifically $\bar{h} = 0$ means that this record is generated by the user him/herself. The user submits an LBS query in the form of $\{R_u, T_{poi}, ID_u, others\}$, where $R_u$ is the ASR of user $u$, $T_{poi}$ means the type of point in which the user is interested, $ID_u$ is the identity of the user $u$, and others is some information related to this LBS. The notations used in the CTPP scheme are summarized in Table 2.

#### 4.1. Multi-hop caching-aware cloaking

The main idea of the MCC algorithm is to allow a mobile user to communicate with multi-hop neighbours and share valid information. With the shared data, the user can (i) generate a cloaking area to blur his/her current location to enjoy an LBS from the remote LSP and (ii) when moving to the next location, locally search for the LBS answer instead of querying the LSP. The built ASR should satisfy the K-anonymity requirement of the user. However, the approach of K-anonymity suffers a serious problem in high density areas, where the cloaking region tends to be very small. Thus the spatial tolerance, $R_{min}$, is considered in our scheme, which can be specified by the user, so that each caching region is sufficiently large to protect the user’s location privacy. If the cloaked area is $R_u < R_{min}$, it can be enlarged by a parameter, $\alpha$. The derivation is as follows [7]:

\[
\begin{align*}
(w + 2\alpha)(l + 2\alpha) &= R_{min} \\
4\alpha^2 + 2(w + l)\alpha + w \cdot l - R_{min} &= 0 \\
4\alpha^2 + 2(w + l)\alpha + R_u - R_{min} &= 0
\end{align*}
\]

where $w$ and $l$ are the width and length of the $R_u$ respectively. Since $R_u - R_{min} < 0$ the discriminant of Eq. 1 is non-negative. Hence $\alpha$ equals the non-negative root of Eq. 1, which is

\[
\alpha = \frac{-2(w + l) + \sqrt{4(w + l)^2 - 16(R_u - R_{min})}}{8}
\]

If $R_u$ is represented by the bottom-left vertices, $(x_l, y_l)$ and the top-right vertices, $(x_r, y_r)$, the extended $R_u$ is $(x_l - \alpha, y_l - \alpha)$ and $(x_r + \alpha, y_r + \alpha)$.

Algorithm 1 depicts the pseudo code of our MCC algorithm. We assume that the anonymity degree is $K$, the required minimum area is $R_{min}$, $H_{max}$ indicates the maximum hop distance of the peers, $l_{next}$ is the next location of the user and $T_{poi}$ is the type of POI. User $u$ performs the MCC algorithm and initializes hop distance $\bar{h} = 1$ in line 2. The user broadcasts a message requesting collaboration to nearby neighbours. The message consists of five parts: a unique message ID, the request for collaboration, a user ID, e.g., an IP address, the hop distance $\bar{h}$ and $T_{poi}$. Then $u$ listens to the network. The peer who is willing to collaborate with $u$ replies and shares the valid data from his/her cache with $u$. The shared information $CachedData$ includes $K$-ASR, the timestamp, $T$, when the ASR is generated, and the service data in which $u$ is interested, as well as other related information. When he/she has received the $CachedData$ from his peers, $u$ labels the value of $\bar{h}$ on the data and stores

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASR</td>
<td>Anonymizing spatial region</td>
</tr>
<tr>
<td>POI</td>
<td>Point of interest</td>
</tr>
<tr>
<td>$T_{poi}$</td>
<td>Type of POI</td>
</tr>
<tr>
<td>$R_{min}$</td>
<td>Spatial tolerances</td>
</tr>
<tr>
<td>$R_u$</td>
<td>ASR of user $u$</td>
</tr>
<tr>
<td>AS</td>
<td>ASR and the anonymity set within it</td>
</tr>
<tr>
<td>$T_u$</td>
<td>Time threshold specified by user $u$</td>
</tr>
<tr>
<td>$T$</td>
<td>Timestamp of a record assigned when ASR is generated</td>
</tr>
<tr>
<td>$H_{max}$</td>
<td>Maximum value of hop distance specified by user</td>
</tr>
<tr>
<td>$\bar{h}$</td>
<td>Hop distance from user to a peer</td>
</tr>
</tbody>
</table>


**Algorithm 1** Multi-hop caching-aware cloaking: requester.

**Input:** $U, K, H_{\text{max}}, L_{\text{next}}, T_{\text{poi}}$

**Output:** $R_u$

1: function \textsc{MultiHopCloak}($U, K, H_{\text{max}}, L_{\text{next}}, T_{\text{poi}}$)

2: \hspace{0.5cm} $\tilde{h} = 1$. List $\leftarrow \emptyset$

3: \hspace{0.5cm} \textbf{while} $\tilde{h} \leq H_{\text{max}}$ \textbf{do}

4: \hspace{1cm} Broadcast request to peers with $\tilde{h}, T_{\text{poi}}$

5: \hspace{1.5cm} //Alg. 2 gives the response of the request receiver

6: \hspace{1cm} List $\leftarrow$ List $\cup \{\text{the received CachedData}, \tilde{h}\}$

7: \hspace{1cm} \textbf{if} $L_{\text{next}}$ is included in the cached ASRs \textbf{then}

8: \hspace{1.5cm} AnonySet $\leftarrow U \cup \{K - 1 \text{ nearest}\}$

9: \hspace{1.5cm} $R_u \leftarrow \text{MBR of points in AnonySet}$

10: \hspace{1.5cm} \textbf{if} $R_u \leq R_{\text{min}}$ \textbf{then}

11: \hspace{1.75cm} Adjust $R_u$ using Equation 2

12: \hspace{1cm} \textbf{end if}

13: \hspace{1cm} \textbf{else}

14: \hspace{1.75cm} $\tilde{h} = \tilde{h} + 1$

15: \hspace{1cm} \textbf{end if}

16: \hspace{0.5cm} \textbf{end while}

17: \hspace{0.5cm} return $R_u$

18: \hspace{0.5cm} \textbf{end function}

it in a list List. Then in lines 7–9 $u$ checks whether his/her next location is included in these gathered ASRs or not. If it is, the user can stop gathering information from his/her peers and build a K-ASR to blur his/her current location in order to enjoy the LBS service. The user chooses $K-1$ nearest points from the stored points and determines a cloaking area, $R_u$ (represented by its bottom-left vertex and top-right vertex), using the minimum bounding rectangle (MBR), which encloses $K$ users. In lines 10–12, if $R_u < R_{\text{min}}$, the ASR is extended by a parameter, $\alpha$, using Eq. 2 to achieve the spatial requirement of the user, $R_{\text{min}}$. Together with this ASR, $u$ issues a live query to the LSP using a pseudo ID and utilizes the LBS. Upon receiving results from an LSP, the user stores this LBS information to answer future queries or to share with other collaborative requesters. If the user cannot find that his/her next location is included in the shared ASRs of 1-hop peers, $u$ increases the value of $\tilde{h}$ to meet the condition until $\tilde{h} = H_{\text{max}}$. Finally, when $\tilde{h} > H_{\text{max}}$, the process of collaboration data collection is completed and then the algorithm returns an ASR, $R_u$, to blur the user’s current location for an LBS query.

Algorithm 2 shows how a peer $p$ responds to the collaboration request together with $T_{\text{poi}}$ and hop distance $\tilde{h}$. Upon receiving the request, $p$ first checks the message ID; if it is a duplicate, he/she replies with an ACK message. Then $p$ decrements $\tilde{h}$ and verifies whether $\tilde{h} \neq 0$; he/she will be either a forwarder or an executor based on the value of $\tilde{h}$:

Case 1: $\tilde{h} \neq 0$, $p$ is a forwarder. $p$ randomly chooses a neighbour (not the requester) to whom to forward the collaboration request with the $T_{\text{poi}}$ and modified $\tilde{h}$. Then $p$ continues to listen to the network until he/she receives the response from the neighbour peer and then returns the results to the message requester; otherwise this forwarder becomes an executor.

Case 2: $\tilde{h} = 0$, $p$ is an executor. $p$ searches the cache with key/value = $T_{\text{poi}}$. If $p$ can find the desired data, he/she directly replies with the information CachedData in the form $\{AS, POI, T, ID\}$. Otherwise, if the user’s requirements cannot be satisfied by the cached service data, $p$ has to build an ASR with his/her 1-hop neighbour and sends the query to the remote LSP together with this ASR. When $p$ receives the results from LSP, he/she generates a timestamp for these data and then returns the data of $\{AS, POI, T, ID\}$ to the message forwarder.

### 4.2. Collaborative privacy-preserving querying

We designed a CPPQ algorithm, which enables the user to either locally search for his/her LBS answers or query the remote LSP. The pseudo code of the CPPQ algorithm is shown in Algorithm 3. In our scheme, the user uses a list List to store his/her history information or keep the shared data received from other peers. The information may have remained in this cache for a period of time, the older the cached data, the lower the QoS that the system provides. Thus, in line 2, the user can check the freshness of data and records. When an ASR is generated, a timestamp, $T$, is assigned to it. The expiration time of stored data is determined by the interval between the timestamp, $T$, and the current system time. If the expiration time of a record exceeds the user-specified threshold value, $T_{\text{thr}}$, it is deleted from the cache. When the user starts to travel toward his/her destination, for example, he/she moves to location $L_u$, he/she first traverses the List to check whether $L_u$ is included in the stored ASRs or not (line 3). If it is, this means that the current location is in either one of the gathered ASRs received from the user’s neighbours or the user’s own stored history cloaking regions. In this case the user can obtain the LBS answers locally by searching for his/her stored service data in his/her own cache (line 4). Then, in lines 5–8, if there are valid ASRs in his/her cache, the user chooses one from which to issue a fake query. The user should try to select a cloaking area at the greatest distance possible from the cached data to confuse the adversary. In general we
Algorithm 2  Multi-hop caching-aware cloaking: receiver.

Input: $T_{pol}, \tilde{h}$
Output: $\text{CachedData}$

1: function $\text{RESPONSE}(T_{pol}, \tilde{h})$ /*This function gives the response of the request receiver $p$.*/
2: if the request is a duplicate then
3: Reply with an ACK message
4: return;
5: end if
6: // when $\tilde{h} \neq 0$, $p$ is a forwarder
7: $\tilde{h} \leftarrow \tilde{h} - 1$
8: if $\tilde{h} \neq 0$ then
9: Randomly choose a neighbour to forward a cooperation request with $(\tilde{h}, T_{pol})$
10: Forward results to $u$ or $r$, when reply received.// Assume $r$ is the forwarder
11: else
12: // when $\tilde{h} = 0$, $p$ is an executor
13: Search cache with keyvalue= $T_{pol}$
14: if $\text{Find}(T_{pol})$ then
15: Send $\text{CachedData}$ back to $r$ or $u$
16: else
17: $R_p \leftarrow \text{Get ASR using Alg. 1(Set } \tilde{h} = 1)$
18: Query LSP with ASR and $T_{pol}$
19: Get results from LSP and return related data to $r$ or $u$
20: end if
21: end if
22: end function

Algorithm 3  Collaborative privacy-preserving querying.

Input: $\text{List}, L_u, T_u$
Output: $\text{POI}$

1: function $\text{CPPQUERYING}(\text{List}, L_u, T_u)$
2: $\text{DelTimeoutResult}((\text{List}, T_u)$
3: if $\text{Find}(L_u, \text{List})$ then
4: $\text{POI} \leftarrow \text{getPOI}(L_u, \text{List})$
5: if Num of fake ASR $\neq 0$ then
6: $\text{ASR} \leftarrow \text{Choose an ASR in List}$
7: $\text{Delete this record from List}$
8: $\text{issueFakeQuery}($ ASR $)$
9: else
10: $\text{Build an ASR by using Alg. 1 with } \tilde{h} = 1$
11: $\text{Issue a live query}$
12: end if
13: else
14: $\text{ASR} \leftarrow \text{Get ASR by Alg. 1}$
15: $\text{issueQuery}($ ASR $)$
16: end if
17: end function

consider that the hop value, $\tilde{h}$, in the record determines the distance between the user and peers. The larger the value of $\tilde{h}$, the greater is the distance. Thus the user should select a record with a larger $\tilde{h}$ value from his/her cache and use the ASR to issue a fake query. In this way, we shift the user’s real cloaking region to a dummy area to break the correlation of the submitted continuous ASR. In the process of sending a query to the server, the requirement belongs to the user, while the ASR belongs to a faraway peer, and the querying user appears to be the virtual user. In lines 9–11, if the cache is empty or no dummy cloaking area that the issuer can use exists, he/she can build a K-ASR by using Algorithm 1 with $\tilde{h} = 1$, so that the LBS query can conform with the K-anonymity principle to preserve privacy. Otherwise, in lines 13–16, if the current location cannot be found in the cached data, the user must run the MCC algorithm to seek cooperation from his/her peers. This mechanism not only reduces the computation and communication cost of the user, but also decreases the amount of private information exposed to the server. Moreover, the disordered queries and fake queries can confuse adversaries and prevent them from correlating sequential ASRs and reconstructing the user’s trajectory.
Table 3
Parameters and their values.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
<th>Ranges</th>
<th>Default values</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>Number of users</td>
<td>2000–5000</td>
<td>5000</td>
</tr>
<tr>
<td>R</td>
<td>Transmission range</td>
<td>150–300</td>
<td>200</td>
</tr>
<tr>
<td>Q</td>
<td>Maximum number of queries</td>
<td>10.20</td>
<td>10</td>
</tr>
<tr>
<td>K</td>
<td>Anonymization degree</td>
<td>3–15</td>
<td>5</td>
</tr>
<tr>
<td>(H_{\text{max}})</td>
<td>Maximum value of hop count</td>
<td>1–7</td>
<td>3</td>
</tr>
<tr>
<td>(T_u)</td>
<td>Time threshold</td>
<td>2–14</td>
<td>6</td>
</tr>
<tr>
<td>(\eta)</td>
<td>Confusion degree</td>
<td>0–1</td>
<td>–</td>
</tr>
</tbody>
</table>

![Confusion degree vs. K.](image)

5. Evaluation

In this section, the experimental evaluation of the efficiency and effectiveness of our proposed CTPP scheme under various system settings is presented. We first describe the experimental setup in Section 5.1, followed by the performance evaluation results presented in Section 5.2 – Section 5.4.

5.1. Experimental setup

Our experiments were implemented with the Java Development Kit (JDK) – 1.7 and Eclipse Integrated Development Environment (IDE), running on a local machine with an Intel Core-i5 2.5 GHz, 2 GB RAM and Microsoft Windows 7 OS. We used the simulator Network-based Generator of Moving Objects [2] to generate mobile nodes and simulate their movement on the real road map of Oldenburg, Germany, a city of approximately 15*15 km². Mobile users were initially distributed among the roads moving at medium speed (three speed profiles can be set for moving objects on the Network-based Generator of Moving Objects: slow, medium and fast). We used various parameters in our evaluation. In each query \(R_{\text{min}}\) was set to 0.5–1%, the value of \(K\) was set to 3–15, \(H_{\text{max}}\) was set to 1–7 and \(T_u\) was set to 2–14 min. We considered a heterogeneous network environment where the transmission range of each mobile user was uniformly set to 100–300 m. Without loss of generality we assumed the mobile user continues to search for the nearest clinic on road on which he/she is located and thus only one POI is considered in our simulations. Table 3 summarizes the parameters and their values used in the experiment. All default parameter values are included in this table unless otherwise noted.

5.2. Confusion degree

The main idea of our scheme is to locally meet the LBS requirement by user collaborations and obfuscate the user trajectory by issuing fake queries. A greater number of fake queries means less private information is exposed to the server and a lower possibility of an attacker reconstructing the trajectory exists. In other words, the success of our scheme depends to a large extent on the number of fake queries. Thus, we computed the confusion degree (\(\eta\)) using Eq. 3 to show whether our privacy preservation technique provides effective protection for the users and then evaluated the impact of the user privacy profile on it. For each set of experiments we randomly chose 20 different users to obtain the average value of \(\eta\).

\[
\eta = \frac{\# \text{fake queries}}{\# \text{total queries}}
\]

Confusion degree vs. \(K\). Fig. 4 shows the results of the confusion degree with varying \(K\), which changes from 3 to 15. Here we set \(T_u = 6\) and \(H_{\text{max}} = 3\). When mobile clients increase their \(K\) anonymity parameter, they need to find more peers to
build a larger cloaking region and thus their trajectory is covered by these regions with a greater probability. Thus mobile users have an improved possibility of locally meeting their LBS service requirements and issuing fake queries. This is verified by the two sets of data shown in Fig. 4, where the maximum number of queries is set to 10 and 20 respectively. The value of $\eta$ grows with the increased $R$, and $\eta$ can be 0.69 and 0.716, when $K=15$.

**Confusion degree vs. $H_{\text{max}}$.** When a user specifies a greater maximum hop distance, more peers can participate in the collaboration and the coverage area become wider. In general the confusion degree increases with $H_{\text{max}}$. We present an experiment in which this was evaluated, where the maximum hop count was increased from 1 to 7 and the transmission range, $R$, was 150 or 300. The results are shown in Fig. 5. When $R = 150$ the confusion degree slowly increases from 0.45 to 0.66 with $H_{\text{max}}$, whereas it increases from 0.63 to 0.78 when $R = 300$. It is clear that the wider the range, the larger is the value of $\eta$.

**Confusion degree vs. $T_u$.** Fig. 6 displays the confusion degree of the scheme when the time threshold ranges from 2 to 14 min. We take the number of users as 2000 and 5000 respectively. When $N = 5000$ the confusion degree increases from 0.43 to 0.8 with the increasing value of $T_u$. If $N$ is fixed at 2000, the increase rate of $\eta$ clearly slows and the confusion degree is almost always no greater than 50%. The reason is that it is more difficult for mobile users to search for peers with whom to collaborate in an area where users are sparse.

#### 5.3. Overhead of collaborative trajectory privacy-preserving scheme

We evaluated the overhead of our proposed CTPP scheme from two aspects: processing time and communication cost. In this experiment we tracked 120 users observing all the footprints on the user’s entire trajectory and computed the average communication and computational overhead of a user at each location to process the LBS query. For comparison purposes, we implemented a baseline method. In the baseline algorithm the user requests his/her nearby neighbours to share their anonymity sets with him/her. Through using the shared points, the user blurs his/her exact location into $K$-ASR and then sends the query to the LSP. Note that this baseline method satisfies only the $K$-anonymity protection of snapshot queries, without considering the leakage of trajectory data in a continuous LBS.
**Processing time.** In the evaluation of the processing time we ignored the transmission delay between mobile users, considering only the processing time of the algorithm itself. The time period is from the moment at which the user constructs an \( K \)-ASR for an LBS request to the moment he/she receives the answer of the LBS. We set \( K = 5, H_{\text{max}} = 3 \), and \( T_u = 6 \). Fig. 7 shows that the time cost of the scheme is inversely proportional to the value of \( \eta \). In the entire journey, if the user has a greater \( \eta \), the number of fake queries becomes larger. In this case, the user can locally settle his/her LBS requirement by searching for the service data in the cached data instead of the LSP. Thus the average processing time of the LBS query at each location decreases.

We compared the average processing time of our scheme with that of the baseline method for various \( K \)-anonymity levels from 5 to 15. Fig. 8 shows the results when \( \eta \) was 0, 50% and 70% respectively. As compared with the baseline method, the computational cost of our proposed scheme is greater when \( \eta = 0 \). The main reason is that it takes a longer time to conduct the MCC algorithm to gather more information from peers. While the average time decreases as the confusion degree increases, since a larger confusion degree means that the user has a greater possibility of locally obtaining the LBS answer rather than having to query the remote LSP. For example, when \( \eta = 70\% \), \( K = 5 \), the average processing time of CTPP is 608 ms, which is considerably less than that of the baseline method.

**Communication cost.** We analysed the average communication overhead of a user at each location in terms of the number of messages. For each message, the maximum size is no greater than 64 bytes. (Among all messages, in our opinion, the shared information \( \text{CachedData} \), which includes the ASR and the service data, has the largest size of a packet; it is smaller than 64 bytes.) The communication cost of a mobile user is composed of two parts. First the user broadcasts a collaboration request to neighbours and receives responses to collect the desired data. Second, the user submits a query to the untrusted LBS server and receives the service data. In general, the first part is the major factor in traffic generation. If a user has to perform the MCC algorithm to seek collaboration many times, the overall communication cost increases. Fig. 9 shows the communication overhead results of a user under different confusion degrees \( \eta \). We set \( K = 5, H_{\text{max}} = 3 \) and \( T_u = 6 \). The average number of messages was 17 and 20 respectively, when \( \eta = 0 \) and 0.1. The communication cost reached a peak when \( \eta = 0.2 \). In the entire trajectories of a user, if \( \eta \) is 0 or 0.1, the user should conduct the MCC algorithm on almost each query point, which generates flow-out messages. We observe that the user whose \( \eta \) is very small is often located
in sparsely populated outlying districts and therefore cannot easily receive a response from nearby neighbours and thus generates very few flow-in messages. The message volume is largest when $\eta = 0.2$. The reason is that user should perform the MCC algorithm many times and meanwhile obtains many responses from peers, which increases the volume of traffic considerably. As the value of $\eta$ increases, there is a greater possibility that the user can locally settle the LBS query and this can reduce the user’s communication cost significantly. Thus the curve trends down as the confusion degree increases and reaches its lowest point when $\eta = 0.5$. Then the average number of messages is approximately 17, when the value of $\eta$ is greater than 0.5.

We compared the average communication overhead of our scheme with that of the baseline method while varying the $K$-anonymity level from 5 to 15. Fig. 10 shows the results when $\eta$ was 0, 50% and 70% respectively. The communication cost of both methods grows with the increasing value of $K$. With a higher anonymity level, mobile clients have to broadcast more messages to their peers and receive more responses and therefore the number of messages increases as the $K$ anonymity parameter becomes larger. In general the communication cost of our CTPP is larger than that of the baseline method. However Fig. 10 shows that the increase is not large when $\eta$ is 50% and 70%. Notice that the baseline method provides only the $K$-anonymity protection of snapshot queries. We can state that our proposed method achieves a better performance in terms of trajectory privacy protection at a small communication cost.

The results of the evaluation show that, if the mobile clients increase the value of $K$, $H_{\text{max}}$ and $T_u$, they can gain a better confusion degree. However with a larger $K$ and $H_{\text{max}}$ parameter, mobile users have to contact more peers who are farther away from them, which results in an increase in response time and traffic. The larger $T_u$ value also means that more data are stored and the information is older. Therefore, we should consider the balance between the performance of confusion degree and the system overhead.

5.4. Privacy level comparisons

Our proposed method provides a spatiotemporal confusing method to protect the user’s trajectory by obfuscating the query time and shifting the anonymity region. We use the distribution of grayscale cells to represent the privacy level of a continuous LBS. A chaos distribution illustrates a higher privacy level. The black arrow represents the user trajectory and
travel direction in Fig. 11. The grid cells are filled with grey shades from light to dark to represent the query time sequence. The time point of the query in cells with a lighter grey shade is earlier than that in cells with a darker grey shade. We compare three privacy methods: (a) the baseline method, (b) the traditional trajectory $K$-anonymity method [5] and (c) our proposed method. As shown in the Fig. 11(a), each snapshot query of the baseline method satisfies the $K$-anonymity principle, but the time order of the queries in this method shows a trend of almost the same grey enhancement. Thus the trajectory of the user may be easily disclosed according to the distribution of the query cells. In Fig. 11(b) the cloaking area becomes larger, since the traditional trajectory $K$-anonymity method expands an initial cloaking region to include at least the same $K$ of users who may move in different directions. Moreover, the query sequence in cells is consistent with the users’ movement direction, which may provide some valuable information for the adversary for inferring the user’s trajectory. However, our CTPP scheme not only satisfies the user privacy requirement but also uses the obfuscation technique to send fake queries to confuse the LBS. Importantly it considers the spatiotemporal factor of queries and utilizes the shifted cloaking region and disordered issuing time to increase the complexity of inferring the user’s direction or tracing the user trajectory. As we can see in Fig. 11(c), the query cell distribution in our proposed method is more chaotic than that of the other methods, showing that our method achieves strong privacy preservation in a continuous LBS.

6. Related work

During the past decades, many promising approaches for preserving location privacy in LBS have been proposed. We roughly divide them into two categories: centralized and non-centralized architecture.

In centralized architecture [11,12,15,20,25] a centralized entity is introduced into the system to protect the location privacy. The principle of $K$-anonymity is the most popular means used for protecting users’ privacy in LBSs. Gruteser and Grunwald [13] originally employed this concept in LBSs to protect location privacy. Inspired by their work, the authors of [11,15,25] proposed efficient spatial cloaking mechanisms to construct a $K$-anonymity cloaking region. In general, all these centralized schemes share some drawbacks. (i) It is not easy to find a third party that can be fully trusted. (ii) The Anonymizer has all the knowledge about users’ locations as well as queries; thus it becomes an attractive target for attacker and therefore the user’s information is jeopardized, when it is attacked by an adversary. (iii) All users have to continuously send their queries and update their locations to the Anonymizer, which causes the Anonymizer to be a performance bottleneck and thus the central point of failure in the entire system. Our previous work [21] addressed these shortcomings at some level, in which the Anonymizer does not have any knowledge about a user’s real location.

In the non-centralized architectures, users cloak their location without trusting a TTP. Some approaches, such as obfuscation-based methods [1,9], cryptographic-based methods [3,23,32] and collaboration-based methods [6,18,19,24], were
proposed to protect the user’s privacy. Obfuscation is achieved by adding noise, quantizing locations or enlarging the location area, without revealing the exact location to the LBS servers. The authors of [1] presented a solution aimed at preserving the location privacy of users by perturbing location information. The main drawback of obfuscation-based methods is that the QoS is degraded because of the low-level accuracy of the answers. Cryptographic tools are used to protect privacy data in the LBS. The authors of [23] employed a ciphertext-policy anonymous attribute-based encryption technique and designed a framework, called FINE, to achieve confidentiality of the LBS data. Cryptographic-based methods are not practical for mobile devices, since they require a powerful computational capability and incur a large pre-processing overhead on the user side. In collaboration-based methods each user communicates with his/her peers and collects their location data to generate the cloaking region. The first collaborative TTP-free algorithm for location privacy in an LBS was proposed in [6]. The main idea is that, before sending a request to an LSP, the mobile user forms a group with his/her peers via single-hop communication and/or multi-hop routing and generates a cloaking area including K users. Shokri et al. [24] designed a distributed location privacy preserving algorithm for a collaborative group, called MobiCrowd, which allows users to answer LBS queries from neighbouring peers so that querying users can protect their location privacy from the LSP. These approaches focused mainly on snapshot queries and the problem of protecting location privacy in the continuous LBSs is not considered in the TTP-free methods.

In a few privacy-preserving techniques, an attempt was made to use the TTP model for continuous LBSs [5,10,14]. The authors of [5] presented an algorithm for trajectory K-anonymity in LBSs, the main idea of which is to continuously expand an initial cloaked area to include at least the same K users. This means that while a request for an LBS is in progress, no grouped user who participated in the original anonymity set of the requester is allowed to leave the group, since this action would jeopardize the privacy of the requester. The authors of [14] proposed a time-obfuscated approach to randomize the sequence of the query issuing time and confuse the LSP and then presented a cloaking method combining ambient conditions to prevent an adversary from reconstructing the user trajectory. In this scheme a user must provide a TTP with a future movement trajectory for each LBS request when he/she starts to travel. Other similar r-1 trajectories are selected in the trusted server’s database to issue redundant queries and therefore the overhead approximately linearly increases because of the greatly increased number of queries. Xu et al. [29] exploited historical locations to construct trajectory K-anonymity and then presented algorithms for spatial cloaking. However when a user moves on the cloaked path, the LBS can still easily identify the user’s actual location, if no other user exists on that path. These techniques rely on a TTP to protect the location privacy of continuous LBSs. In contrast to other previous schemes, the scheme proposed in this paper is the user-centric technique to provide location privacy for continuous LBSs using non-centralized architectures.

7. Conclusion

In this paper, we proposed an efficient collaborative scheme, CTPP, to protect users’ trajectory privacy in continuous LBSs. Our scheme does not require any fully trusted third party; instead trajectory privacy is preserved by the users’ collaboration based on cached information. Our goal is to confuse the LSP and by using an obfuscating strategy to prevent the adversary from reconstructing a user’s actual trajectory from continuous queries. We presented the design of two heuristic algorithms to achieve our goal. In the MCC algorithm users share information with multi-hop peers. Specifically, when the user’s location can be found in the cached data, the scheme utilizes a CPPQ algorithm to locally meet the LBS requirement by searching for the stored service data and issues a fake query based on the dummy cloaking area, which is constructed by a faraway neighbour. On the one hand the method can reduce the amount of private information exposed to the server and on the other hand can break the correlation of the continuous queries in order to confuse the LSP and adversary. In our scheme, it is assumed that the collaborative users are honest and can trust each other. In future work we will consider the situation where the user is potentially untrustworthy and present an enhanced security scheme in continuous LBSs.

Acknowledgement

This research was supported in part by the High Level Talents Program of Higher Education in Guangdong Province under grant 2016ZJ01, the National Natural Science Foundation of China under grant numbers 61472451&61272151&61402161, the International S&T Cooperation Program of China under grant 2013DFB10070, the Hunan Provincial Science Technology Program of China under grant 2012GG4106, the Hunan Provincial Natural Science Foundation of China under grant 2015JJ3046, and the CERNET Innovation Project under grant NGII20150408.

References


