Research Article

A Voronoi-Based Location Privacy-Preserving Method for Continuous Query in LBS

Chunguang Ma, Changli Zhou, and Songtao Yang

School of Computer Science and Technology, Harbin Engineering University, Harbin 150001, China

Correspondence should be addressed to Chunguang Ma; machunguang@hrbeu.edu.cn

Received 24 April 2014; Accepted 7 October 2014

Academic Editor: Jianwei Niu

Copyright © 2015 Chunguang Ma et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Most of the mobile phones have GPS sensors which make location based service (LBS) applicable. LBS brings not only convenience but also location privacy leak to us. Achieving anonymity and sending private queries are two main privacy-preserving courses in LBS. A novel location privacy-preserving method is proposed based on Voronoi graph partition on road networks. Firstly, based on the prediction of a user’s moving direction, a cooperative \( k \)-anonymity method is proposed without constructing cloaking regions which may lead to efficiency decline in continuous query. And then, a query algorithm is proposed without providing any user’s actual location, replaced by continuous anchor sequence, to LBS provider. This algorithm can work out precise results according to candidate sets returned by LBS provider and it also solves uneven distribution problem in SpaceTwist. Performance analysis and experiments show that our method achieves a preferable tradeoff between QoS and location privacy preserving; it has obvious advantages compared with other methods.

1. Introduction

Nowadays, GPS sensors are available on many smart phones, which can report accurate location of a user, and location based service (LBS) has been widely used. Query service for places of interests (PoIs) based on a user’s location is one of the most frequently used service types in LBS, such as nearest neighbor queries and range queries. Typical query is “get \( K \) nearest restaurants around me” and “get all restaurants within the scope of \( R \) kilometers,” respectively. And a user may send a snapshot or a continuous query request [1] according to his current moving condition. For all these query methods ask a user to provide his actual locations before getting query service. But it leads to a deep-going privacy exposure as the location data is leaked due to the correlation between location information and user’s identity, hobbies, home address, and other privacy data. The location privacy must be protected effectively in the process of LBS.

There are 3 kinds of privacy in LBS, location privacy, query privacy [2–5], and identity privacy [6]; we focus on location privacy preserving. How to cloak a user’s location has been widely studied [7–13] and most of them choose a central server to achieve \( k \)-anonymity for users. The architecture of location privacy preserving in LBS is central or centerless. In Figure 1, the centerless architecture has two kinds of entities: users and LBS providers (LSP), but in this architecture privacy-preserving burden is heavy at the user end. The central architecture is added with a central server (CS) providing stronger privacy-preserving methods. Location \( k \)-anonymity normally adopts the latter architecture to construct cloaking regions at CS end, as shown in Figures 1 and 2, a user sends his actual location to a CS, and the anonymity module of CS obscures the location before sending the query to LSP; the untrusted LSP cannot get any user’s precise location. But constructing cloaking regions for each user causes lots of burden and low success rate of anonymity when CS confronts with a large number of anonymity requests.

In view of that defect, Chow et al. [14] have proposed P2P cooperative anonymity method without CS. In that method, the head node which comes from users organizes other users to form an anonymity group without constructing any cloaking region. It is a simple structure but too much burden for a user to be a head node, and the users’ credibility is hard to guarantee.
However, using snapshot cloaking region is not security in continuous query because adversaries are supposed to get the distribution of users and cloaking regions all the time, so when a user constructs continuous cloaking regions for a user, as shown by the solid line rectangles in Figures 2(a)–2(c), the intersection A of three regions is the actual user using anonymity service. To solve this problem, researchers have proposed lots of methods for continuous query [15–24], such as constructing continuous minimum cloaking regions [15–17], as shown by the dash line rectangles in Figure 2. But constructing continuous regions needs to find neighbor users who have similar moving trend and makes sure most of them will not depart from the cloaking regions. That increases the difficulty of constructing and has a side effect which restricts the users in a small specified region. Query with these regions requires processing function of imprecise coordinates at server end, too. Moreover, most of these methods are based on Euclidean space, not based on the actual road networks.

There are other protecting methods, such as dummies [25, 26], using pseudo locations instead of actual location. These methods have the defects of imprecise query results or too much overhead. Lung Yiu et al. [27] have proposed SpaceTwist which gets candidate sets of Pols from LSP. SpaceTwist achieves comparatively accurate query result. Gong et al. [28] and Huang et al. [29] have, respectively, improved SpaceTwist. We also consult SpaceTwist to make it applicable for continuous query with an anchor sequence. The main contributions and innovations of this paper are as follows.

1. Considering the defects of constructing continuous cloaking regions, we propose a location privacy-preserving method without any cloaking region. We use Voronoi graph partition for the first time to divide road networks into cells, which facilitates the prediction of a user’s moving direction, and a user’s registration algorithm to CS is proposed with the help of Voronoi partition. When a mobile user registers to a CS, he has a high accuracy to an appropriate one. After registration, a cooperative anonymity algorithm at CS end is proposed. This algorithm satisfies each user’s k-anonymity degree.

2. We propose an accurate query algorithm, which is called Singoes running on CS, without providing any user’s actual location to LSP. Singoes takes fixed anchors between two cells to replace the user’s actual locations in continuous query and get precise query results using anchors. According to the returned candidate Pols from LSP, Singoes deduces precise results by expanding demand space again. This algorithm also solves the problem of Pols uneven distribution in SpaceTwist. As we know, we have found the problem for the first time.

2. Related Work

In order to provide strong location privacy preservation at the data generated end, Gruteser and Grunwald [7] have proposed k-anonymity to obscure a user’s actual location. After that lots of location privacy-preserving works have been presented [8–14, 24], Chow et al. [10] and Li and Zhu [13] have proposed partition method for cloaking region, and Xue et al. [2] have proposed cloaking region constructing method in road networks, which is appropriate for practical situation. Chow et al. [14] have proposed a P2P cloaking algorithm without a CS. We take these ideas as a reference in this paper.

The above researches were proposed in a static situation; Dewri et al. [17] make a formal analysis on the privacy leaking problems in continuous LBS, which points out that the static protecting method is not appropriate for mobile situation. Now most works are caring about dynamic privacy preserving. Pan et al. [15] take users’ moving speed and direction as affecting factors and a distortion function is defined to measure the temporal query distortion of a cluster in continuous queries, and they also have proposed a method which maintains maximal cliques to deal with location-dependent attacks [16]. Lee et al. [18] have proposed a grid-based cloaking region creation scheme for continuous LBS, and Wang et al. [19] consider the similarity of velocity and acceleration as factors to construct cloaking regions. Pingley et al. [23] use dummies to protect privacy in continuous query; this method does not need a trust third party. Hashem et al. [20] have presented a mobile K nearest neighbor query against overlapping rectangle attack in continuous query. In road networks, Palanisamy et al. [21] and Yang et al. [24] have proposed location protecting methods in continuous query, which have more practical significance.

In the aspect of using dummies to achieve privacy preserving, Hong and Landay [25] use a significant object to replace a user’s location, but the query result is not accurate enough. Reference [26] sends several dummies accompanied with actual location to LSP to get query results, which brings extra burden to LSP. Based on these researches, Lung Yiu et al. [27] have proposed SpaceTwist running on user side, LSP performs INN (Incremental Nearest Neighbor) searching process for Pols and returns the candidate sets to the user, and the user works out the accurate results himself. But SpaceTwist has not achieved k-anonymity, so Gong et al. [28] and Huang et al. [29] have proposed improved methods. And there is still an issue in SpaceTwist which can be improved.

As shown in Figure 3, we cite the figure of SpaceTwist to specify the issue that can be improved. There are 3 courses in SpaceTwist, a user staying at q takes q’ as an anchor, then the user sends an INN query request with anchor q’ to LSP, as the
second PoI $P_2$ is found, supply space which is the dark grey circle with a radius $r$ expands, and demand space which is the light grey circle with a radius $y$ shrinks, the query procedure will not terminate until supply space covers demand space, and the user at $q$ gets $K$ PoIs.

The above process has a problem called PoIs uneven distribution. After $P_3$ is found, if a $P_x$ exists, which is closer to $q'$ than $P_4$, as shown in Figure 4(a), the query algorithm will terminate immediately since the demand space shrinks and it is covered by supply space; the other PoIs will not be found anymore. That is caused by the appearance of a PoI $P_x$ which is near to $q$ in the opposite direction of $q'$. The demand space will shrink quickly and is covered by expanded supply space. Moreover, all the found PoIs are distributed
around the anchor $q'_1$ and not around the user at $q_1$; as shown in Figure 4(b), no Pols in bottom left of the dash line have been found. All the found Pols are above the dash line around anchor $q'_1$; the user does not get his $K$ nearest neighbor Pols. We call this case Pols uneven distribution problem. We will solve this problem and guarantee that the found Pols are distributed around the user rather than the anchor, which is more precise. SpaceTwist is proposed for snapshot query; it is not related to continuous query. In this paper, we will make it appropriate to get continuous query results.

3. Relevant Knowledge

In this section, we will introduce system structure, cooperative $k$-anonymity, and road network partition based on Voronoi graph.

3.1. System Structure. The structure of privacy-preserving method contains three parts: as shown in Figure 5, a mobile user is denoted as $u_k$. Each user is equipped with an intelligence terminal which contains a GPS sensor and velocity and direction sensors. CS contains three modules; the anonymity module deals with user’s registration, makes moving direction prediction for a user, and forms cooperative $k$-anonymity groups; the communication module broadcasts the current users’ registration situation of its cell periodically and communicates with users. After constructing an anonymous group, query processing module sends query requests with anchors to LSP; this module refines Pols candidate sets and deduces $K$ NN Pols a user requires. CS is credible, and LSP and users are not.

There are also adversaries who aim to find a user’s actual location and a correlation between a user identifier and a query request. We suppose that an adversary has three abilities: (1) obtaining the distribution of mobile users without their identities; (2) obtaining cloaking regions; (3) correlation analysis ability and some background knowledge.

3.2. The Concept of Cooperative Anonymity. Anonymity method can be divided into single anonymity and cooperative anonymity; for example, constructing cloaking regions for each user is single anonymity, and cooperative anonymity is different users sharing one cloaking region. The sharing idea decreases CS’s burden. In this paper, we do not construct cloaking regions but organize users to form a cooperative anonymous group just like a P2P anonymous group [14, 29]. But the P2P cooperative method has to select a user as a cluster head, which is not suitable for energy constrained mobile users. So we modify this method and add powerful CS to organize users in its cell to achieve cooperative $k$-anonymity. Cooperative $k$-anonymity in this paper should meet the following two criterions.

(1) Each user belongs to a sole cooperative group.

(2) A cooperative $k$-anonymity group meets any user’s $k$-anonymity degree.

3.3. Road Network Division Based on Voronoi Graph Partition. Voronoi graph [30] is a geometry of flat space; as shown in Figure 6, it is a way of dividing space into a number of regions. A set of spots is specified beforehand and for each spot there will be a corresponding region consisting of all points closer to that spot than to any other. One way is connecting all CSs around CS$_1$ with dash line segments and doing perpendicular bisectors to all these line segments. The polygon with connected bisectors is a Voronoi cell.

Similarly, we choose places to deploy central servers (CSs); the deployment follows 3 criterions.

(1) A CS is deployed in crowded and traffic convenient places, which insures a lot of users in its cell avoiding constrained space identity attack. Users help CS to transmit messages.
(2) The distance between neighbor CSs meets $\zeta_a < \text{dist}(CS_i, CS_k) < \zeta_b$, and $\zeta_a$ and $\zeta_b$ are thresholds, which insure the area of a cell is reasonable.

(3) The connected paths of neighbor cells are not less than a threshold $\zeta_c$, as shown in Figure 8, which insures a user is not specified in a sole road when he is moving from one cell to another.

After a certain density deployment of CSs, we divide the entire area $\Omega$ using Voronoi graph partition into many cells centered with different CSs. Each cell is used to achieve cooperative anonymity. The Voronoi graph partition is defined as follows.

**Definition 1** (Voronoi graph partition). The spots of $\{CS_1, CS_2, \ldots, CS_n\}$ are distributed in a 2-dimension space $\Omega$, for any $V(CS_1)$ meeting $V(CS_i) = \{A | \text{dist}(A, CS_i) \leq d(A, CS_k), A \in \Omega, i \neq k\}$, where dist$(A, B)$ is the Euclidean distance between $A$ and $B$, and all the $V(CS_i)$ forms a Voronoi graph, and the subregion $V(CS_i)$ is a cell.

The whole city area can be denoted as $V(\Omega) = \{V(CS_1), V(CS_2), \ldots, V(CS_n)\}$. Obviously, neighbor cells have no overlap area and each user belongs to a sole cell. All the users in the same cell can form different cooperative $k$-anonymity groups.

4. Cooperative Anonymity and Continuous Query with Anchors

There are 3 phases in our method: initialization phase, cooperative $k$-anonymity phase, and continuous query Pols with anchors phase. After initialization phase, a user registers to a CS and sends his request of joining in a cooperative anonymous group, CS organizes $k$ users to form a cooperative $k$-anonymity group according to their moving trends and sends $k$ queries to LSP, a user’s identity is removed, and his actual location is replaced by continuous anchors in the query procedure. After registration, a user should report his latest locations to a CS periodically to achieve continuous query until he gets in a new cell or finished a query. In query phase, the CS takes its own location and a neighbor CS location which the user is heading to as two elementary anchors and chooses other continuous anchors in the line segment between them. Then CS sends each snapshot query of continuous query request with an anchor which the user has not passed yet. In a snapshot query, CS performs algorithm Singoes to compute results for a user according to candidate sets returned from LSP.

4.1. Initialization Phase. A Trust Authority (TA) picks appropriate CSs’ deploying locations according to the 3 criterions and generates Voronoi graph partition cells. Then TA sends road segments and neighbor CSs situation of each cell to the corresponding CS; the details will be discussed in Section 4.2. Any CS can verify validity of a user’s pseudonym [31].

A $CS \in CS$ initializes a registering table according to registered time sequence of each user $Table_R = (UID, loc, dest, t_r, \bar{V}, Con, k, Pset, n_i)$, where UID is a pseudonym of a user, and $t_r$ is the registered time, loc and dest are the location of a user and his destination, respectively, $\bar{V}$ represents a user’s history velocity vectors, Con represents a query content, $k$ is his anonymity degree, $Pset$ is a prediction of the next cells, and $n_i$ is registered users’ amount. $CS_i$ broadcasts his registration situation $B_{ci} = (CS_{i}, loc_{ci}, n_i, t_r, r_{\text{min}}, r_{\text{max}})$ at a regular interval; $t_r$ is broadcasting time, and $r_{\text{min}}$ and $r_{\text{max}}$ are defined in Definition 2 and in Figure 7.

**Definition 2** (repartition of a cell). If a user’s location meets dist$(u_k, CS_i) < r_{\text{min}}$, he is in a convinced region, if it meets dist$(u_k, CS_i) > r_{\text{max}}$, he is in an excluding region, and if it meets $r_{\text{min}} \leq \text{dist}(u_k, CS_i) \leq r_{\text{max}}$, he is in an unconvinced region.

As shown in Figure 7, a user in a convinced region of a cell is certainly in the cell, such as $C$, and in an excluding region, the user is certainly not in the cell, such as $D$, and in an unconvinced region, a user has both situations, such as $A$ and $B$.

4.2. Registration and Cooperative $k$-Anonymity Phase. In road networks, registration and anonymity are affected by user’s moving direction, so we specify a cell prediction method first.

4.2.1. Road Networks Voronoi Graph Partition. Direction prediction is important to all the following procedures. We consider a user is in restrained road networks; his moving
Direction is affected by roads. The Voronoi graph partition is on the road networks, as shown in Figure 8. In initialization phase, TA marks all the roads with numbers according to its sole actual road name, the same road may go through several cells, and the roads are directed connected graph and form road networks. Then TA sends two tables to every corresponding CS, Table_{c1} records all the road segments in a cell, and Table_{c2} records all of the neighbor CSs of a cell.

Definition 3 (Table_{c1}). The road segments table is denoted as Table_{c1} and represented by its attributes Table_{c1} = \langle \text{RoadID} | \text{Connect RoadID} \rangle, \text{RoadID} is the identification number of a road in cell \( V(\text{CS}) \), and Connect RoadID is all the directly connected roads to RoadID in the cell.

As shown in Figure 8, in a cell \( V(\text{CS}_i) \), Road Segment 1 is recorded in Table\_{c1} as a row record \((1 | 2,3,4)\). Each road segment in \( V(\text{CS}_i) \) has the same kind of row record.

Definition 4 (Table_{c2}). All the neighbor CSs of a cell are in Table_{c2} and represented by its attributes Table_{c2} = \langle \text{CS} | \text{loc}_{\text{cs}} | \text{Connect RoadID} \rangle, \text{CS} is neighbor's identities, loc_{\text{cs}} is the location of CS, and Connect RoadID represents directed connected roads to a neighbor cell.

As shown in Figure 8, a neighbor CS_k of cell \( V(\text{CS}_i) \) is recorded as \( \langle \text{CS}_k | \text{loc}_{\text{cs}k} | 2,3,4 \rangle \); each neighbor CS has the same kind of row record in Table_{c2}.

The correlation of Table_{c1} and Table_{c2} can deduce paths which leads a user in a cell to another neighbor cells. Due to the constraints of roads, that is, a user has to travel along the roads rather than arbitrary locations in Euclidean space, CS can predict next neighbor cell for a user according to his history movements and destination. The result of prediction is also provided to the user, which helps the user to register to next CS.

4.2.2. Next Neighbor Cell Prediction. Although there are roads constraints, a user always chooses the paths which most of its road segments approximately point to his destination, or else the user will not get to his destination. And most of his history velocity vectors, constrained by the roads, have the same feature. CS uses Formula (1) to predict the weights of each neighbor cell for a user according to his registering message \( \text{Register}_{uk} = (u_k, \text{loc}_{\text{cs}k}, \text{dest}, t_r, V, \text{Con}, k_{uk}) \), and then it stores the prediction results in Pset. Suppose \( u_k \) is in a cell \( V(\text{CS}_j) \); CS\_j computes the weight of one of his neighbor cells CS\_i for \( u_k \):

\[
P_{cai} = \delta_d \cdot \theta(\overrightarrow{\text{CS}_j \text{dest}}, \overrightarrow{\text{CS}_i \text{CS}_j}) + \delta_v \cdot \frac{\sum_{x=1}^{w} \theta(\overrightarrow{\text{CS}_i}, \overrightarrow{v_x})}{w}.
\]

\( \overrightarrow{\text{CS}_j \text{dest}} \) is the vector from \( \text{CS}_j \) to dest, \( \theta \) is used to compute the angle between two vectors, \( \overrightarrow{v_x} \in V \) is one of the user's history velocity vectors, and \( w \) is its amount; \( \delta_d \) and \( \delta_v \) are weights which are 0.4 and 0.6, respectively. According to Formula (1), we can find that a smaller \( \theta(\overrightarrow{\text{CS}_j \text{dest}}, \overrightarrow{\text{CS}_i \text{CS}_j}) \) means a small deviation to a user's destination if the user chooses a cell \( V(\text{CS}_j) \) to pass through, and more history velocity vector pointing to \( \text{CS}_j \) means entering the cell \( V(\text{CS}_j) \) will not bring large deviation comparing with history directions. Therefore, the lower \( P_{cai} \) is, the higher probability of a user to go through \( \text{CS}_i \). \( \text{CS}_i \) computes all the neighbor cells' weights for a user and stores them in Pset = \{\( P_{cai}, P_{cik}, P_{cjr}, \ldots \)\} in ascending order.

After the registration for a user, CS checks the road number which the user travels on by reverse geocoding techniques and computes Pset; the neighbor cell with minimum weight is the most likely cell which the user will enter into.

The prediction result also affects the anchors selection in continuous query. It also can be used to plan routes for users. The connecting paths of next cell are deduced by Definitions 3 and 4; CS can select the best one for a user according to path length and road congestion situation.

4.2.3. User Registration. A user will receive several neighbor CSs broadcasting messages \( B = \{B_{\text{CS}1}, B_{\text{CS}2}, \ldots, B_{\text{CS}n}\} \), and he needs to register to an appropriate CS that he is in or about to. As a user's query request will be sent to LSP through a CS, enough users in the same cell make a user avoid being the only target to be attacked. According to these messages, only if there are enough users in the appropriate cell, a user will register to the CS or else wait for the next message. \( u_k \)'s other related information will be sent within Register\_uk = \{(u_k, loc_{\text{cs}k}, \text{dest}, t_r, V, \text{Con}, k_{uk})\}. The key point is how to register to an appropriate CS according to \( B = \{B_{\text{CS}1}, B_{\text{CS}2}, \ldots, B_{\text{CS}n}\}\).

As shown in Figure 7, according to repartition, \( u_k \) estimates the cell which he is in or heading to: for each \( B_{\text{cs}i} \in B \), \( u_k \) drops the old message and computes the distance between each CS\_i and himself; if \( \text{dist}(\text{loc}_{\text{cs}k}, \text{loc}_{\text{cs}i}) < r_{\text{min}}(\text{CS}_i) \) is found, \( u_k \) is definitely in \( V(\text{CS}_i) \) according to Definition 2, the user registers to the \( \text{CS}_i \) and drops all the other messages then or else drops all messages which meet \( \text{dist}(\text{loc}_{\text{cs}k}, \text{loc}_{\text{cs}i}) > r_{\text{max}} \) because these messages are from excluding regions. And then \( u_k \) computes minimum distance \( \min(\text{dist}(\text{loc}_{\text{cs}k}, \text{loc}_{\text{cs}i})) \) of these neighbor CSs, because \( u_k \) is in their unconvincing regions at the same time, and registers to the minimum distance CS. Since the deployment of CS, lots of users can help to transmit broadcasting messages and each user is more possible to receive the message from his cell quickly. Even if \( u_k \) have not received the message from the appropriate cell during a time period, that means there are not enough users around the CS; it cannot achieve k-anonymity for any user either. \( u_k \) should not register to that CS. The static registration algorithm is as shown in Algorithm 1.

Algorithm 1 is only suitable to static or slow users, but when a user moves quickly, such as B in Figure 8, if he moves upward of the figure and registers to CS\_j, he will register wrong even he is still in V(\text{CS}\_j), because he is about to get out of that cell. So we modify the algorithm with cell prediction to make sure that a user registers to an appropriate CS (see Algorithm 2).

After successful registration to a CS, a user will report his latest locations periodically to the CS and drop broadcasting message from it until the user gets out of the cell or registers
Algorithm 1: A static user estimates the Voronoi cell he is staying in and registers to the CS.

(1) Procedure: \( u_k \) receives \( n \) broadcasting messages \( B = \{ B_{c1}, B_{c2}, \ldots, B_{cn} \} \) from different CSs, a \( B_{ci} \in B \) is \( B_{ci} = (CS_i, loc_{ci}, ni, t_i, r_{min}, r_{max}) \).
(2) \( u_k \) generates empty arrays \( Array \) and \( RegisTime \).
(3) for each \( B_{ci} \in B \) do
(4) While \( t_{now} - t_i < \delta \) and do // \( t_{now} \) is current time, \( \delta \) is a threshold, the formula helps to drop old messages
(5) compute \( dist(loc_{uk}, loc_{ci}) \)
(6) if \( dist(loc_{uk}, loc_{ci}) < r_{min} \) then
(7) discard all the other messages
(8) else if \( dist(loc_{uk}, loc_{ci}) > r_{max} \) then
(9) discard \( B_{ci} \)
(10) else \( Array \leftarrow B_{ci} \)
(11) compute \( min(dist(loc_{uk}, loc_{ci})) \) \( Array \)
(12) if \( min(dist(loc_{uk}, loc_{ci})) \) exists then
(13) register to CS \( i \)
(14) \( RegisTime \leftarrow T_{reg}(CS_i) \) // record the register time to CS \( i \)
(15) End Procedure.

Algorithm 2: Registration algorithm with cell prediction for dynamic users.

(1) Procedure: \( u_k \) receives \( n \) broadcasting messages \( B = \{ B_{c1}, B_{c2}, \ldots, B_{cn} \} \) from different CSs, a \( B_{ci} \in B \) is \( B_{ci} = (CS_i, loc_{ci}, ni, t_i, r_{min}, r_{max}) \).
(2) \( u_k \) generates empty arrays \( Array \), \( Angle \) and \( Register \).
(3) for each \( B_{ci} \in B \) do
(4) While \( t_{now} - t_i < \delta \) do // \( t_{now} \) is current time, \( \delta \) is a threshold, the formula helps to drop old messages
(5) compute \( dist(loc_{uk}, loc_{ci}) \)
(6) if \( dist(loc_{uk}, loc_{ci}) < r_{min} \) then
(7) discard all the other messages
(8) else if \( dist(loc_{uk}, loc_{ci}) > r_{max} \) then
(9) discard \( B_{ci} \)
(10) else \( Array \leftarrow B_{ci} \)
(11) compute \( min(dist(loc_{uk}, loc_{ci})) \) \( Array \)
(12) if \( min(dist(loc_{uk}, loc_{ci})) \) exists then
(13) register to CS \( i \)
(14) \( RegisTime \leftarrow T_{reg}(CS_i) \) // record the register time to CS \( i \)
(15) End Procedure.
to next cell. If a user is in an unconvinced region, he has a probability to enter into a neighbor cell, so prediction to the next cell is significant; the prediction has two factors, as shown in Line 17: one is the current moving direction of the user himself, the other one is cell prediction results from CS he had just registered to, referring to Formula (1), and the former one we have not discussed yet, and it is a little complicated that we divide it into 3 cases.

(1) \( u_k \) in a convinced region need not estimate next cell. As shown in Figure 9(a), \( u_k \) staying at \( C \) decides whether to register to \( CS_i \) according to his motion state. We divide motion state into two kinds: (1) when \( \theta_1 = \langle \overrightarrow{V_{uk}}, \overrightarrow{u_k CS_i} \rangle < 90^\circ \), \( u_k \) has just gotten into a cell and received \( B_{cai} \), so \( u_k \) registers to \( CS_i \) directly without estimating next cell; (2) when \( \theta_2 = \langle \overrightarrow{V_{uk}}, \overrightarrow{u_k CS_i} \rangle > 90^\circ \), \( u_k \) has just left from the convinced region, which indicates \( u_k \) has entered into the cell and registered to \( CS_i \) for a certain time; \( B_{cai} \) is a repeat message now and discarded by \( u_k \). To sum up, if \( u_k \) finds that he is in a convinced region, he need not compute angle \( \theta = \langle \overrightarrow{V_{uk}}, \overrightarrow{u_k CS_i} \rangle \) or predict next cell.

(2) \( u_k \) in an excluding region need not estimate next cell. As shown in Figure 9(b), \( u_k \) stays at \( D \) and receives \( B_{cai} \); if he has just left the cell, he will discard it, or if he is about to enter into the cell he will receive \( B_{cai} \) again with high probability when he is much closer to \( CS_i \). So in this case, \( u_k \) is far from \( CS_i \); he discards \( B_{cai} \) without estimation.

(3) \( u_k \) in an unconvinced region must estimate next cell. As shown in Figure 9(c), \( u_k \) staying at \( B \) may receive several messages, from \( CS_j \) or \( CS_k \). (1) When \( \theta_3 = \langle \overrightarrow{V_{uk}}, \overrightarrow{u_k CS_j} \rangle > 90^\circ \), \( u_k \) is leaving from \( V(CS_j) \); he discards \( B_{cai} \) even he is still in the cell. (2) When \( \theta_1, \theta_2 < 90^\circ \) and \( \theta_1 = \theta_2 \), it is hard for \( u_k \) to estimate which cell he is heading to; \( u_k \) needs a further estimation. In Algorithm 2 Lines 16–22, all the angle
4.2.4. Cooperative Anonymity Algorithm. After receiving registrations from users, CS organizes them to form several cooperative $k$-anonymity groups which meet each member's $k$-anonymity degree in a group. As discussed before, cooperative $k$-anonymity is just like the way P2P achieves anonymity, and the function of cluster head is undertaken by a CS. As shown in Algorithm 3, CS$_i$ checks the road number of the registering user staying at and computes $P_{set}$ for the user. CS$_i$ searches registration table $Table_p$ for all the users who have the same next cell prediction result and stores them in array Sim. That considers user's moving direction similarity and insures proportion of users using the same anchor sequence to avoid correlating attacks. Then CS$_i$ picks users in Sim on neighbor roads by preference; if users in a group are not enough, the group will be combined with another group. It is worth noticing that the way of constructing cloaking regions always finds the minimum boundary rectangle containing neighbor uses, which insures the security of a query. In our method, CS organizes users registered to the same cell to form anonymity groups without constructing cloaking regions, so we do not consider finding neighbor users but the similar direction users in the cell. The cell is similar to a cloaking region and several users in it are chosen to form a group (see Algorithm 3).

In Algorithm 3, Lines 3–5 pick users who have the same next neighbor cell to form the group; Lines 6 and 9 make the constructed group satisfy each user’s anonymity degree. The combination in Line 10 guarantees quality of anonymity service. After successful anonymity, CS removes users’ identities and sends the group of queries to LSP with continuous anchors instead of any user's actual location. An item of query is denoted as $(CS_i, loc_{anchor}, Con)$, $loc_{anchor} \in$ Anchor; Anchor is an anchor sequence and Con is a query content.

### Algorithm 3: Cooperative anonymity group construction.

|Line| Description |
|---|---|
|1| **Procedure**: CS$_i$ received registration message $Register_{ak}$ from $u_k$ and computes $P_{set_{ak}}$ for him according to Formula (1), CS$_i$ stores $P_{set_{ak}}$ in $Table_p$. The CS owns two tables: $Table_1$ and $Table_2$ as in Definitions 3 and 4, respectively. |
|2| CS$_i$ generates empty arrays $Sim$, $Group$ |
|3| for each $u_i \in Table_p$ do |
|4| if $\min(P_{set_{ak}}) = \min(P_{set_{ak}})$ then |
|5| $Sim \leftarrow u_i$ |
|6| $Group_{ak} \leftarrow$ find users in $Sim$ clockwise whose anonymity degree meets $k_{ai} \leq k_{ak}$ |
|7| if $|Group_{ak}| < k_{ak}$ then |
|8| for other $Group_{ak}$ do |
|9| if $|Group_{ak}| + |Group_{ak}| \geq \max(k_{ai}, k_{ak})$ then |
|10| $Group \leftarrow$ combine $Group_{ak}$ with $Group_{ak}$ |
|11| return $Group$ |
|12| End procedure |

less than 90° may be the next cell, for each of the $u_k$ computes $M_i$ which is affected by prediction results $P_{set}$ from CS$_i$; the minimum $M_i$ is the predicted result, which makes the estimation more accurate.

In conclusion, only the users in an unconvinced region need to predict next cell by computing $M_i$, those are the users who are leaving from the current cell to another, and it is consistent with actual situation. Therefore the repartition of a Voronoi cell is befitting for registration and simplifies the procedure for a user.

4.3. Query Phase

4.3.1. Anchor Picking. When a user has gotten out of one cell he will enter into another cell inevitably. Therefore, a user picks anchors according to the cell prediction result in continuous query. The anchor sequence is generated by two elements: one is the location of CS$_i$ which the user is staying in now; the other one is the location of CS$_{m}$ which the user is heading to. The anchor sequence is on the line segment from CS$_i$ to CS$_m$; the anchor amount in a sequence depends on user query frequency. Midpoint of the segments is picked as an anchor:

$$loc_{midm} = \left(\frac{X_{csi} + X_{csm}}{2}, \frac{Y_{csi} + Y_{csm}}{2}\right). \quad (2)$$

An anchor sequence with three anchors for $u_k$ is denoted as $Anchor_{uk} = (loc_{csl}, loc_{midm}, loc_{con})$; CS computes anchor sequences for different users. The users who enter into the same next cell have the same anchor generating elements.

4.3.2. Query Algorithm. CS picks an anchor from anchor sequence to replace user’s location and send a snapshot query of a continuous query request [23]. The anchor is the one which the user has not passed in his anchor sequence. So CS asks a registered user to report his latest location in a continuous query procedure. When LSP receives a query, it uses the anchor to perform an INN search. After candidate results are returned from LSP, CS uses Algorithm 4 to work...
(1) Procedure: $K \in Con, q \leftarrow$ get $u_k$’s current location
(2) $q' \leftarrow$ pick an anchor which $u_k$ has not passed yet in the anchor sequence $\text{Anchor}_{u_k}$
(3) the registered central server generates a min heap $H_{\min}$
(4) insert $K$ pairs of $\langle \text{NULL}, \infty \rangle$ in previous snapshot query into $H_{\min}$
(5) $\gamma \leftarrow$ the bottom distance in $H_{\min}$
   // initialize radius of demand space
(6) $\tau \leftarrow 0$ // initialize radius of supply space
(7) Send INN search request to LSP with anchor $q'$
(8) while $\gamma + \text{dist}(q,q') > \tau$ and $\tau < r_{\text{threshold}}$ do
(9)   POI $\leftarrow$ continue to get an INN package of PIs from LSP
(10) $\tau \leftarrow \max_{\text{P}_i \in \text{POI}} \text{dist}(q',\text{P}_i)$
(11) for each $\text{P}_i \in \text{POI}$ do
(12)   if $\text{dist}(q,\text{P}_i) < \gamma$ then
(13)     update $H_{\min}$ and $\gamma$ by using $\text{dist}(q,\text{P}_i)$
(14) $\gamma \leftarrow \text{dist}(q,\text{P}_k)$ of $K$th PIs in $H_{\min}$
   // expand demand space and don’t change $\gamma$ anymore
(15) while $\gamma + \text{dist}(q,q') > \tau$ and $\tau < r_{\text{threshold}}$ do
(16)   POI $\leftarrow$ continue to get an INN package of PIs from LSP
(17) $\tau \leftarrow \max_{\text{P}_h \in \text{POI}} \text{dist}(q',\text{P}_h)$
(18) if $\text{dist}(q,\text{P}_h) < \gamma$ then
(19)   update $H_{\min}$ by using $\text{dist}(q,\text{P}_h)$
(20) terminate INN search request
(21) return $K$ PIs from the first one in $H_{\min}$
(22) End Procedure.

Algorithm 4: Singoes running on CS for KNN PIs.

out each snapshot query result which is precise KNN PIs to a user. We propose Algorithm 4 at CS end consulting SpaceTwist, but SpaceTwist has the uneven distribution problem we have specified in Section 2. We will solve this problem to make Pol results distribute around a user but not around the anchor. We call Algorithm 4 the “Singeos.”

As proposed in [27], SpaceTwist terminates when supply space covers the demand space, as shown in Figures 3(c) and 4(a). That is why the PIs are around anchor $q'$. One solution to deal with this problem is to make the query procedure continue to get more PIs and terminate when KNN PIs around the user are found. Singoes is as shown in Algorithm 4.

As shown in Figure 3(c), when $\text{P}_4$ is found, supply space covers the demand space, and SpaceTwist terminates and gets $K = 4$ PIs $\{\text{P}_3, \text{P}_2, \text{P}_1, \text{P}_4\}$, but they are around anchor $q'$. In Singoes, demand space expands for the last time and covers $K$ PIs in the region, which guarantees $K$ PIs at least are gotten by the user, as shown in Figure 10(a). As each Pol is found gradually, supply space will not stop expending until it covers demand space. As shown in Figure 10(c), LSP returns 8 PIs in total. Singoes gets $K = 4$ PIs $\{\text{P}_3, \text{P}_2, \text{P}_1, \text{P}_4\}$, distributing around $q$, which is more precise than SpaceTwist.

After finishing a snapshot query with an anchor, the other PIs will not be discarded, because the user may get closer to these PIs in next snapshot query of the same continuous query request with anchor sequence. As shown in Figure 10(c), because a snapshot anchor is always picked in front of user, when a user is at $q$ and heading to anchor $q'$, $\{\text{P}_3, \text{P}_2\}$ is closer to the user, so CS can use them to refine the result with a user’s latest location to ensure continuous query quality.

4.3.3. Brief Summary. The procedure of our method is as follows.

After initialization, a user who has LBS query request registers to a CS according to his current location and cell prediction results from previous CS. If he registers to a CS for the first time, he will use Algorithm 1 to achieve registration. Then CS predicts the next cell for him according to his history velocity vectors and current direction; CS also returns the prediction results to the user to help him achieve next cell registration. After registration the user should report his latest location and CS constructs a cooperative $k$-anonymity group for users, and then CS sends a group of queries to LSP with anchors. Each anchor for a user is generated according to the CS locations of his current cell and next cell. At last, CS performs Singoes to compute KNN PIs for the user according to Pol candidate sets returned from LSP. For each snapshot query belonging to a continuous query request, CS always picks an anchor in front of the user to ensure the query quality according to Singoes.
4.4. Performance Analysis

4.4.1. Analysis of Anonymity Procedure

**Accuracy of Prediction and Registration.** Cell prediction is affected by a user’s history velocity vector and the current direction. A user always chooses a path in which most of its road segments approximately point to his destination; otherwise the user will not get to his destination. And most of his history velocity vectors, constrained by the road, have the same feature. Since CS's neighbor cells are not many, the direction range of each neighbor cell around the user is obvious; prediction with angular separation has a high success rate. According to Algorithm 2, only the user who is in an unconvinced region needs to predict his next cell, and the prediction has a high success rate. It is a rare event that the user changes his direction in an unconvinced region since the road constraints. According to criterion \( \varsigma_a < \text{dist}(\text{CS}_i, \text{CS}_k) < \varsigma_b \) in Section 3.3, it will not happen that an unconvinced region covers its neighbor convinced regions, so there is no registration in advance by mistake.

**Success Rate of Cooperative k-Anonymity.** CSs are picked in crowded locations around by lots of users. After Voronoi partition, global map with \( U \) users uniform distribution is divided into \( b \) cells, each cell containing about \( U/b \) users in average. The Voronoi partition is analogous to divide all the users into \( U/b \) anonymous groups in advance, so Voronoi partition helps users to achieve \( k \)-anonymity quickly. In Algorithm 3 Lines 7–10, the combination of groups guarantees the anonymity quality. Suppose a mobile user whose velocity is 70 km/h enters into a cell with \( r_{\text{max}} = 500 \) meters; the passing time is about \( t \approx 0.5 \times 2/70 \times 3600 \approx 51 \) seconds; experiments in Section 5.2 show the response time of query is less than 1 second. So LBS will last for a while in a cell; a user has enough time to achieve \( k \)-anonymity without a jitter.
Anonymity and Query Security. Suppose users are distributed uniformly; if an adversary knows $n$ users distribution in a cell, a user will be identified with a probability $p(x_i) = 1/n$ when he issues a query; the entropy of issuing a query is

$$H(q) = \sum_{i=1}^{n} p(x_i) \log \frac{1}{p(x_i)} = \sum_{i=1}^{n} \frac{1}{n} \log n = \log n. \quad (3)$$

Our method does not utilize cloaking regions so there are no overlapping region attacks. A trusted CS provides anonymity of each query from the same user, and most of the users in a cooperative anonymous group querying with anchors only indicate that these users have passed the cell. An adversary cannot know any user's actual location or the correlation with any query. Then $m$ queries from the same user are mutual independence, so the information entropy of a continuous query is

$$H(q_1 \cdots q_m) = \sum_{i=1}^{m} H(q_i) = m \log n. \quad (4)$$

So our method achieves a less information release which increases an adversary's uncertainty.

Deployment Cost. Constructing a Voronoi graph depends on a large number of CS deployments, which brings cost increasing. That is a defect of our method. But comparing with mobile communication base station whose deployment generally follows shouting distance about 1000 m, our method is acceptable. Distribution service from CSs helps to reduce service bottleneck. And CS can be embedded and maintained as a part of communication base station.

4.4.2. Analysis of Query Procedure

Query Efficiency. Suppose PoIs are randomly distributed on a 2-dimensional space; CS can obtain $K$ PoIs if it searches a circle no more than a radius of $\gamma$, so PoIs on a unit area can be computed as $K/\pi \gamma^2$, and the number of PoIs on a region with a radius $\tau$ is about $N_1 = \pi \tau^2 \cdot (K/\pi \gamma^2) = K \cdot (\tau/\gamma)^2$, in Figure 10(c). When Algorithm 4 terminates, $\tau = \gamma + \text{dist}(q,q^1)$, so the number of PoIs $N_1$ becomes

$$N_1 = K \cdot \left(1 + \frac{\text{dist}(q,q^1)}{\gamma}\right)^2. \quad (5)$$

The fewer PoIs in $N_1$, the higher query efficiency. As we have just supposed, $\gamma$ is a constant value; therefore the query efficiency is affected by the distance between a user and his anchor $\text{dist}(q,q^1)$, which means the anchor picking is significant to query efficiency. As shown in Figure 10(c), if each anchor picking satisfies $\text{dist}(q,q^1) < \gamma$, so $N_1 < 4K$. Compared with using $n$ dummies to issue every PoIs query, which LSP has to return $N_2 = nK$ PoIs, Singoes has a remarkable advantage when $n > 4$. The searching upper limit of Singoes is $4K$, but the query procedure will terminate before reaching it in practical.

Query Accuracy. CS uses a user’s actual location to work out KNN PoIs according to returned candidate sets; in Singoes line 14, demand space expands again and keeps it radius $\gamma$ unchanged, and demand space covers $K$ Pols at least that time, so even if there are no other Pols found in the following procedure, $K$ Pols in the demand space can be found in 100% success rate, which guarantees finding KNN results around a user.

Fixed Anchor. There are many ways to generate anchors; most of them are generated randomly and temporarily. But the anchor sequence in our method is generated only by two neighbor CSs, and other anchors are restrained on the line segments between them; the anchors are not many since the user is moving fast from one cell to another, so the anchors are picked from the integer division points of the line segment; most users may use the same anchor in the anchor sequence, so the anchor is nearly fixed. And the advantages of fixed anchor are as follows.

1. An anchor which is related to a CS’s location is near to users; it guarantees a high efficiency of continuous query as we have just discussed.
2. Different users pick the same fixed anchors and make adversaries confused; it is much more difficult for them to relate a query with a user.
3. Fixed anchor queries are sent to LSP; LSP can use cache to store same queries with fixed anchors, which decreases searching database operations compared with random anchors.

5. Experiments

We discuss 4 main metrics in LBS: anonymity success rate, average response time, data traffic, and accuracy of query results. And the affecting factors of metrics are users amount $U$, user’s anonymity degree $k$, and Pols number $K$. The experiments run on different data sets. Comparative experiments of anonymity and query followed.

5.1. Environment Preferences. Simulation experiment is running on Windows 7, CPU is 3.5 GHz Intel Core i7 processor, and RAM is 16 GB. The algorithms are written by Java. We use two data sets, one is a real data set from BGN (http://geonames.usgs.gov/index.html), denoted as GDS, and it includes 169553 Pols. The other one is simulated data set (http://aapg.jade-hs.de/personen/brinkhoff/generator/), denoted as TBS, and this data set is generated by widely used Thomas Brinkhoff Generator which is based on road networks of Oldenburg in Germany; it generates a city area about 24 km × 27 km. The bandwidth between CS and users is 3 Mbps. CS gets a data package of 128 bytes from LSP each time. Removing 40 bytes package head, if a Pol costs 8 bytes, a package contains at most $\beta = (128 - 40)/8 = 11$ Pols. The parameters and defaults are in Table 1.

5.2. Experiments on Different Data Sets. We estimate anonymity success rate, average response time, average packages amount, and accuracy of query results as the number of LBS users increasing on dataset GDS and TBS. Anonymity
### Table 1: Parameters and defaults.

| Parameters                  | Value range | Defaults |
|-----------------------------|-------------|----------|
| Anonymity degree $k$        | $5 \leq k \leq 30$ | 10       |
| Number of users $U$         | $25000 \leq U \leq 200000$ | 100000   |
| $\text{dist}(q, q')$       | $500 \leq \text{dist}(q, q') \leq 1500$ | 1000 m   |
| Pols capacity $\beta$      | $1 \leq \beta \leq 11$ | 6        |
| Query number of Pols $K$   | $5 \leq K \leq 30$ | 10       |
| Amount of Pols $m$          | —           | 169553   |

**Figure 11:** Success rate of anonymity.

Success rate is the ratio of users achieving anonymity and all LBS users. Response time is the time cost during anonymity and query Pols, packages amount is sending and receiving packages during response time, and accuracy of query results is the rate of precise Pols in a KNN query result and $K$; that is, if 10 NN query results are Pols of $\{A, B, C, \ldots, I, J\}$, and eight of them are the ones which a user wants and the $K$ nearest neighbor Pols to the user, the accuracy of query results is 0.8.

As shown in Figures 11-12, as LBS users are increasing, anonymity success rate is gradually increasing, maintaining ratios more than 80%, and users achieve anonymity effectively and timely. The average response time declines gradually in Figure 12; there are more messages to deal with when confronting with more users; the average response time would have increased, but CS does not construct cloaking regions, and more users make CS facilitated to form cooperative anonymity groups quickly and the Pols are quickly found with fixed anchors in LSP’s cache; therefore the average response time declines and keeps stable at 0.3 seconds.

As shown in Figure 13, although packages increase as the number of users increases, average packages maintain stable, because users achieve anonymity by forming anonymity groups instead of constructing continuous cloaking regions; most of them send the request once in a cell until they get out. In the other hand, packages of query for Pols are increasing, because Singoes needs to search more area via LSP to get KNN Pols to a user, and the communication between CS and users increases more, as shown in Figure 14.

As shown in Figure 15, the accuracy of query result maintains steady. That is because Algorithm Singoes guarantees the demand space covers $K$ Pols in the second expansion and the radius of demand space keeps unchanged; then LSP gradually returns candidate Pols to Singoes; when the supply space covers the demand space, the searching procedure terminates. Due to the reliability of LSP, all the Pols which...
a user queries for are found in the unchanged demand space, so it is easy for Singoes to find the precise $K$ nearest neighbor PoIs in the candidate set around the user who is the center of the demand space.

5.3. Comparative Experiments of Anonymity. The comparative experiments are carried out on simulation dataset TBS. Firstly, we compare Casper* [10], P2P anonymity [14] with cooperative $k$-anonymity method we have proposed. As shown in Figure 16, when users are increasing, packages of P2P anonymity have a significant growth due to communications among users to form anonymous groups. And there is much burden for anonymizers in Casper* to construct cloaking regions for users; therefore the average packages and response time increase both. Therefore the anonymizer is a service bottleneck. Average packages keep stable in our cooperative method without constructing cloaking regions in any cell.

Large anonymity degree of users means more users are needed to achieve $k$-anonymity; as shown in Figure 17, packages are increasing as $k$ becomes larger. The packages of our method are less than the other two, because CS organizes $k$ registered users to form anonymous groups in low communications. But the success rate of anonymity will
decline as \( k \) is increasing. From Figures 16-17, we conclude that average packages of our method will maintain stable even if \( U \) and \( k \) vary.

5.4. Comparative Experiments of Query. We compare SpaceTwist [27], PrivacyGrid [32], P2P [14] with our query algorithm Singoes in the query procedure. SpaceTwist and our method both use anchors to query for Pols. But Singoes uses fixed anchors in every snapshot of the same continuous query request. SpaceTwist runs on user end; Singoes runs on CS. The other two methods PrivacyGrid and P2P query for Pols both with cloaking regions.

Figure 18 shows that the average packages of query will increase in two algorithms, because both of them have to search more area, which have to return more PoI candidate sets. And in Figure 19, PrivacyGrid has similar packages communication when \( U \) is less than 100 thousand, and Singoes’s communication cost is lower than it when \( U \) is growing larger. P2P has more packages increasement than the other two methods, because the cluster head has to transmit Pols to each user when \( U \) is growing larger.

In continuous query, Singoes does not discard returned Pols which are filtered once in KNN query; most of these Pols in front of the user are used for the next snapshot queries, which makes a snapshot query terminate as soon as KNN Pols are found. So packages of Singoes are less than SpaceTwist. In Figure 20, as fixed anchors are picked, LSP will store these Pols around frequently used anchors in cache, which significantly reduces searching time at LSP end, so the average query time of Singoes does not have a significant growth.

PrivacyGrid and P2P are both higher than Singoes in Figure 20; their query time is growing as \( K \) increases, especially PrivacyGrid which has to construct cloaking regions for each user, and querying with the cloaking regions leads to the time delay, so the query time cost of Singoes is better than these methods.

6. Conclusion

We propose anonymity algorithm and query algorithm to achieve location privacy preserving. Based on Voronoi graph partition on road networks, we propose cell predicting method as a precondition. In anonymity procedure, CS need not construct cloaking regions for each user, which reduces time and communication cost. And then we propose query algorithm Singoes referring to SpaceTwist, which solves the
PoIs uneven distribution problem in SpaceTwist. It uses fixed anchor sequence to issue every snapshot query of the same continuous query request. That protects a user’s actual location, and any adversary is hard to associate a query request with a user. The precise query results of KNN PoIs are obtained around a user’s actual location. Performance analysis and experiments show that our method achieves QoS of query for PoIs and location privacy preserving. It has better performances compared with other methods.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This research is supported by a grant from National Natural Science Foundation of China (nos. 61170241 and 61472097), the Fundamental Research Funds for the Central Universities (HEUCFZ1105), Specialized Research Fund for the Doctoral Program of Higher Education (no. 20132304110017), Excellent Youth Foundation of Heilongjiang Province in China (no. JC 201117), and Science and Technology Research Project of Heilongjiang Education Department (nos. 12513049 and 12541788). The authors also thank the anonymous reviewers and the editors for their valuable comments.

References

[1] M. A. Cheema, W. Zhang, X. Lin, and Y. Zhang, “Efficiently processing snapshot and continuous reverse k nearest neighbors queries,” The VLDB Journal, vol. 21, no. 5, pp. 703–728, 2012.
[2] J. Xue, X.-Y. Liu, X.-C. Yang, and B. Wang, “A location privacy preserving approach on road network,” Chinese Journal of Computers, vol. 34, no. 5, pp. 865–878, 2011.
[3] X. Pan, X. Hao, and X. Meng, “Privacy preserving towards continuous query in location-based services,” Journal of Computer Research and Development, vol. 47, no. 1, pp. 121–129, 2010.
[4] C. Chow Y and F. Mokbel M, “Trajectory privacy in location-based services and data publication,” ACM SIGKDD Explorations Newsletter, vol. 13, no. 1, pp. 19–29, 2011.
[5] J. Krumm, “A survey of computational location privacy,” Personal and Ubiquitous Computing, vol. 13, no. 6, pp. 391–399, 2009.
[6] D. Wu, X. Wang, L. Sun, Y. Ling, and D. Zhang, “Identity privacy-based reliable routing method in VANETs,” Peer-to-Peer Networking and Applications, vol. 7, no. 3, pp. 285–294, 2014.
[7] M. Gruteser and D. Grunwald, “Anonymous usage of location-based services through spatial and temporal cloaking,” in Proceedings of the 1st International Conference on Mobile Systems, Applications and Services, pp. 31–42, ACM, 2003.
[8] B. Gedik and L. Liu, “Location privacy in mobile systems: a personalized anonymization model,” in Proceedings of the 25th IEEE International Conference on Distributed Computing Systems (ICDCS ’05), pp. 620–629, 2005.
[9] M. F. Mokbel, “Towards privacy-aware location-based database servers,” in Proceedings of the 22nd IEEE International Conference on Data Engineering Workshops (ICDEW ’06), p. 93, 2006.
over road networks,” in Proceedings of the IEEE 8th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob ’12), pp. 435–442, October 2012.

[25] J. I. Hong and J. A. Landay, “An architecture for privacy-sensitive ubiquitous computing,” in Proceedings of the 2nd International Conference on Mobile Systems, Applications, and Services, pp. 177–189, ACM, 2004.

[26] H. Kido, Y. Yanagisawa, and T. Satoh, “An anonymous communication technique using dummies for location-based services,” in Proceedings of the International Conference on Pervasive Services (ICPS ’05), pp. 88–97, July 2005.

[27] M. Lung Yiu, C. S. Jensen, X. Huang, and H. Lu, “SpaceTwist: managing the trade-offs among location privacy, query performance, and query accuracy in mobile services,” in Proceedings of the IEEE 24th International Conference on Data Engineering (ICDE ’08), pp. 366–375, 2008.

[28] Z. Gong, G.-Z. Sun, and X. Xie, “Protecting privacy in location-based services using K-anonymity without cloaked region,” in Proceedings of the 11th IEEE International Conference on Mobile Data Management (MDM ’10), pp. 366–371, IEEE, May 2010.

[29] Y. Huang, Z. Huo, and X.-F. Meng, “Coprivacy: a collaborative location privacy-preserving method without cloaking region,” Chinese Journal of Computers, vol. 34, no. 10, pp. 1976–1985, 2011.

[30] R. I. Saye and J. A. Sethian, “Analysis and applications of the Voronoi implicit interface method,” Journal of Computational Physics, vol. 231, no. 18, pp. 6051–6085, 2012.

[31] X. Lin, R. Lu, X. Liang, and X. Shen, “STAP: a social-tier-assisted packet forwarding protocol for achieving receiver-location privacy preservation in VANETs,” in Proceedings of the IEEE International Conference on Computer Communications (IEEE INFOCOM ’11), pp. 2147–2155, IEEE, Shanghai, China, April 2011.

[32] B. Bamba, L. Liu, P. Pesti, and T. Wang, “Supporting anonymous location queries in mobile environments with PrivacyGrid,” in Proceedings of the 17th International Conference on World Wide Web (WWW ’08), pp. 237–246, Beijing, China, April 2008.
