Wireless Pers Commun
DOI 10.1007/s11277-008-9608-9

Privacy Protected Spatial Query Processing for Advanced Location Based Services

Wei-Shinn Ku · Yu Chen · Roger Zimmermann

Abstract Due to the popularity of mobile devices (e.g., cell phones, PDAs, etc.), location-based services have become more and more prevalent in recent years. However, users have to reveal their location information to access location-based services with existing service infrastructures. It is possible that adversaries could collect the location information, which in turn invades user's privacy. There are existing solutions for query processing on spatial networks and mobile user privacy protection in Euclidean space. However there is no solution for solving queries on spatial networks with privacy protection. Therefore, we aim to provide network distance spatial query solutions which can preserve user privacy by utilizing K-anonymity mechanisms. In this paper, we propose an effective location cloaking mechanism based on spatial networks and two novel query algorithms, PSNN and PSRQ, for answering nearest neighbor queries and range queries on spatial networks without revealing private information of the query initiator. We demonstrate the appeal of our technique using extensive simulation results.

Keywords Privacy protection · Mobile data management · Location-based services

1 Introduction

As a result of recent technological advances, mobile devices with significant computational abilities, gigabytes of storage capacity, and wireless communication capabilities have

W.-S. Ku (✉)
Department of Computer Science and Software Engineering, Auburn University, 345 W. Magnolia Ave, Shelby Center 3101, Auburn, AL 36849, USA
e-mail: weishinn@auburn.edu

Y. Chen
Department of Electrical & Computer Engineering, Binghamton University, Binghamton, NY 13902, USA

R. Zimmermann
Department of Computer Science, National University of Singapore, Singapore 117590, Singapore

become increasingly popular. In addition, positioning techniques are embedded into more and more mobile devices. Based on these advances, new mobile applications allow users to issue location-dependent queries in a ubiquitous manner. Examples of such location-dependent searches include '*find the nearest gas station*' and '*find the top three closest French restaurants*'. To get location-dependent data, users have to reveal their current locations when launching location-dependent queries. From the server query logs of location-based service providers (LBSP), it is possible that adversaries could collect the location history and monitor the behavior of some users, which in turn invades their privacy [20]. Therefore, with the popularity of location-based services (LBS), mobile user privacy protection becomes a very important research issue for future generation communication and networking.

Recent research has explored the K-anonymity concept [18][1] in which one trusted server is needed to cloak at least K users' locations for protecting location privacy. In order to implement K-anonymity, one trusted server is set up to collect user location information and perform cloaking procedures in which the exact location of the query requester is blurred as a cloaked spatial area whose boundary is defined by the locations of $K-1$ other users. Then, the trusted server will send the location-dependent query along with the cloaked spatial area to location-based service providers to retrieve the relevant data. Note that since the query location is an area instead of a single query point, location-dependent service providers should fetch those query results based on the cloaked spatial region. Prior work in [14] proposed a framework for location-based services without compromising location privacy. However, only a free space environment is considered, which is not fully applicable in real world environments. On the other hand, recent research has produced novel mechanisms to compute location-dependent queries on spatial networks. For example, executing nearest neighbor queries based on the spatial network distance provides a more realistic measure for applications where mobile user movements are constrained by underlying networks. Though devising spatial query schemes in spatial networks, the prior works in [11,15] did not consider location privacy issues. Thus, we aim to provide spatial query (nearest neighbor query and range query) solutions that address privacy protection concerns in this study. With our techniques, location-based service users can enjoy high quality results without sacrificing their privacy. The contributions of our study are as follows.

- We design an improved cloaking mechanism for spatial network search space.
- We propose a novel algorithm for solving privacy protected nearest neighbor queries on spatial networks.
- We extend our nearest neighbor query solution to answer range queries with protection of privacy.
- We demonstrate the feasibility and efficiency of our approaches through extensive simulations.

2 Related Work

In this section, we introduce the background information and related research regarding spatial query processing, location-based services, and location privacy preservation.

[1] Note that we use the symbol K for the degree of anonymity and k for k nearest neighbor queries.

2.1 Spatial Query Processing

We focus on two common types of spatial queries, namely k nearest neighbor queries and range queries. With R-tree [9] based spatial indices, depth-first search (DFS) [16] and best-first search (BFS) [10] have been the prevalent branch-and-bound techniques for processing nearest neighbor (NN) queries. The DFS method recursively expands the index nodes for searching nearest neighbor candidates. At each newly visited non-leaf node, DFS computes the ordering metrics for all its child nodes and applies pruning strategies to remove unnecessary branches. When a leaf node is reached, the data objects are retrieved and the nearest neighbor candidates are updated. In addition, the BFS technique utilizes a priority queue to store nodes to be explored through the search process. The nodes in the queue are sorted according to their minimum distance (MINDIST) to the query point. During search, the BFS repeatedly dequeues the top entry in the queue and enqueues its child nodes with their MINDIST into the queue. When a data entry is dequeued, it is inserted into the result set. In order to increase the NN query accuracy, recent research proposed solutions based on spatial networks. Kolahdouzan and Shahabi [11] presented a novel approach to efficiently evaluate kNN queries in spatial network databases using a first order Voronoi diagram. Papadias et al. [15] proposed two algorithms, the Incremental Euclidean Restriction (IER) algorithm and the Incremental Network Expansion (INE) algorithm to solve nearest neighbor queries on spatial networks.

2.2 Location Privacy Preservation

With the popularity of LBS, privacy protection for mobile users has become an important issue [8, 17]. Gruteser and Grunwald [7] proposed a middleware architecture and algorithms for maintaining location K-anonymity. Their algorithms adjust the resolution of location information along spatial or temporal dimensions to fulfill the required anonymity constraints. Based on the work in [7], a unified privacy personalization framework is proposed in [6] to support different levels of anonymity according to the requests of users. However, these previous research approaches mainly focused on the system architecture and the location anonymizer design rather than query processing. Mokbel proposed to employ a trusted third party, the *location anonymizer*, which expands the user location into a spatial region for protecting user privacy [13]. Mokbel et al. also proposed related privacy-aware query processing algorithms [14]. However their query processing solutions are based on Euclidean metrics. In real life, mobile users cannot move freely in space but are usually constrained by underlying networks (e.g., cars on roads, trains on tracks, etc.). Therefore, we need solutions for processing privacy protected queries on spatial networks [12].

3 System Architecture

We first describe the system architecture for supporting privacy protected spatial queries with underlying spatial networks. Figure 1 depicts our operating environment with three main entities: *mobile users*, the *location cloaker*, and *location-based service providers*. We consider mobile clients such as cell phones, personal digital assistants (PDA), and laptops, that are instrumented with global positioning systems (GPS) for continuously obtaining position information. Furthermore, we assume that there are access points/base stations distributed in the system environment for mobile devices to communicate with the location cloaker. All

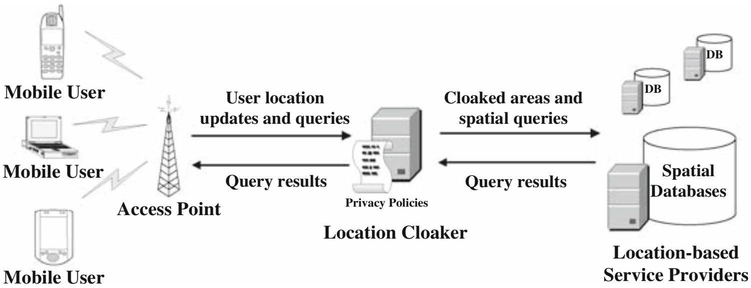


Fig. 1 The system architecture

users are mobile and travel on the underlying network and they also hold *privacy policies* which specify the privacy requirements of each user. In this research we focus on three parameters, K-anonymity degree, the minimum cloaked region size—R_{min}, and the minimum covered network segments—S_{min}, that are included in the user privacy policies. A user can demand the cloaked area to cover the locations of $K-1$ closest peers for anonymizing its exact location. In order to keep a reasonable size of the cloaked area in highly user-dense regions, the user decides the minimum acceptable size of R_{min}. Since mobile users are moving on underlying spatial networks, the cloaked region has to cover a minimum number of road segments, S_{min}, to avoid location tracking by adversaries. Based on the privacy requirements at different locations or time slots, a user can update his/her privacy policies at any time.

3.1 The Location Cloaker

Compared with location-based service providers, the location cloaker is an intermediate agent which can be trusted by mobile users. The location cloaker receives continuous location updates from mobile users and blurs their exact locations into cloaked areas A_c according to individual user privacy policies before forwarding the information to location-based service providers (e.g., for buddy searching services). In addition, the location cloaker also anonymizes the location of any query requesting user q to a cloaked region before forwarding the query to related location-based service providers. Note that any user identity related information in the query is also removed by the location cloaker during the cloaking process.

According to previous research [6–8,14] there are several different mechanisms to support location anonymization. Compared with existing solutions, the location anonymizing technique proposed by Mokbel et al. [14] has prominent efficiency (low cloaking time) and flexibility (user defined privacy profile) for the cloaking process in free space environments. However, the proposed location anonymizer does not consider the characteristics of spatial network environments. Figure 2 demonstrates an example. The cloaking region in Fig. 2a is generated without considering the underlying spatial networks and there is only one road segment (i.e., the W 35th Street) contained in A_c. Advisories with access to the map of this city can easily infer that the mobile user is currently driving on the W 35th Street. Consequently, the effect of the location anonymizing mechanism proposed in [14] may be drastically weakened in spatial network environments.

In our system, we adopt the grid-based pyramid data structures proposed in [1,19] for managing the locations of mobile users. The pyramid structure is dynamically maintained for monitoring the current number of mobile users within each unit space of the whole spatial

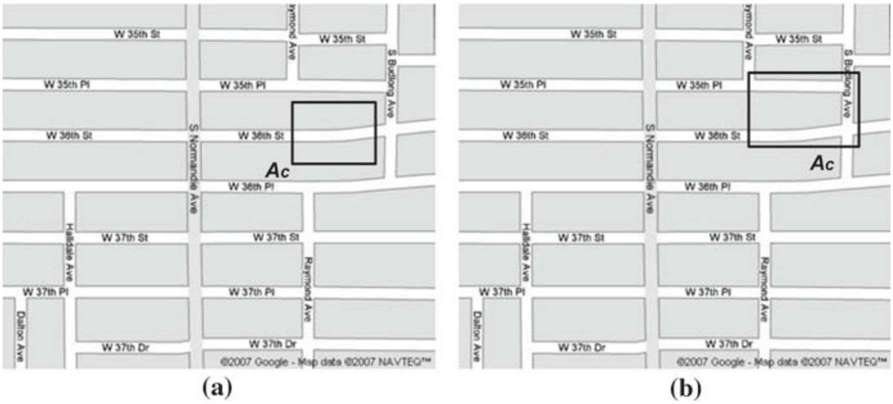


Fig. 2 Location cloaking in spatial network environments. (**a**) Location cloaking without considering of underlying spatial networks. (**b**) Location cloaking with considering of underlying spatial networks

area [14]. Consequently, the size of the cloaked region can be efficiently computed. In order to avoid the aforementioned drawbacks, users have to define a minimum number of road segments which should be covered in the cloak region. The revised cloak region after taking S_{min} into account is demonstrated in Fig. 2b ($S_{min} = 3$). With a minimum number of road segments within A_c, mobile user privacy can be positively secured.

3.2 Location-Based Service Providers

Location-based service providers play the role of spatial data maintainers and spatial query processors in our system. In order to handle privacy protected spatial queries, location-based service providers implement *privacy protected query processors* in their databases. The privacy protected query processor has the ability to process cloaked spatial queries efficiently and retrieves an *inclusive result set* (i.e., the minimal set which covers all the possible answers) for query requesters. After receiving the result set, mobile users can distill the exact answers from their locations in linear time. The privacy policies of a user determine the computational complexity of his/her spatial queries. Strict privacy requirements (i.e., large K, R_{min}, and S_{min} values) increase the complexity of processing the query.

In addition to a privacy protected data processor, a LBSP also needs to maintain spatial databases for storing (cloaked) user locations, spatial data and road networks. The stored spatial data can be categorized as public data and private data. Public data covers static objects such as restaurants, hotels, and gas stations and also the dynamic information (e.g., real-time bus locations) which are directly open to public queries. In contrast, private data mainly comprise cloaked mobile user locations from the location cloaker. Based on the two data categories, the spatial queries submitted to a LBSP can be classified into four types: (1) *public queries over public data*, (2) *public queries over private data*, (3) *private queries over public data*, and (4) *private queries over private data*. For the first query type, there were already existing solutions proposed in [11,15]. Because the movement of mobile users is limited by the underlying road networks, we can easily extend the mechanisms proposed in [11,15] for solving queries of the second query type (e.g., Fig. 3a) with probability density functions [4]. To the best of our knowledge, there is no existing solution for the third query type. Similarly the fourth query type (e.g., Fig. 3b) can be answered by extending the

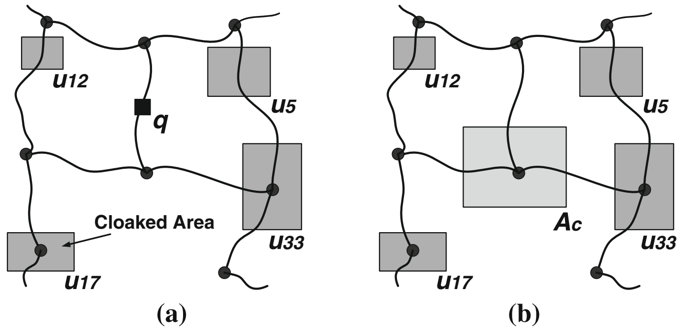


Fig. 3 Two novel query types. (**a**) Public query over private data. (**b**) Private query over private data

algorithms of the third query type. Therefore, we propose our novel techniques for solving private queries over public data on spatial networks in Sect. 4.

In addition, as discussed in [15], we assume that the spatial network database supports the following primitive operations:

- *inside_segments*(A_c): returns a set of subsegments of a network N which intersects with the cloaked area A_c.
- *find_objects*($segment_x$): returns the data objects which fall on the input network segment $segment_x$.
- $Dist(p_1, p_2)$: calculates the network distance of two input points, p_1 and p_2, in the underlying network by applying an algorithm (e.g., Dijkstra's algorithm [5]) to compute the shortest path between p_1 and p_2.

4 Privacy Protected Query Processing

We now illustrate our mechanisms for solving private queries over public data on road networks. We focus on two popular query types, nearest neighbor queries and range queries.

4.1 Privacy Protected Nearest Neighbor Query on Spatial Networks

Given a query point q and an object data set $\mathbb{S}$, a network k nearest neighbor query retrieves the k objects of $\mathbb{S}$ closest to q based on the network distance. Papadias et al. [15] proposed two algorithms (*incremental Euclidean restriction* and *incremental network expansion*) to efficiently solve nearest neighbor queries with spatial network databases. However in order to protect user privacy, a location-based service provider can only receive cloaked spatial areas from users. Therefore, LBSPs need to have a competent mechanism for retrieving an inclusive query result set based on the input cloaked area and the underlying spatial network. We design a *privacy protected spatial network nearest neighbor query* (PSNN) algorithm by extending the incremental network expansion algorithm [15].

PSNN first locates all the intersection points between the edges of the input cloaked area A_c and the spatial network as a point set $\mathbb{T}$ by executing primitive database operations. If $\mathbb{T}$ is not empty, A_c covers at least one network segment. We denote these network segment(s) within A_c as Seg_i and the segments outside A_c as Seg_o. According to the network topology, the network edges inside A_c can be fully connected or comprise several separate subgroups.

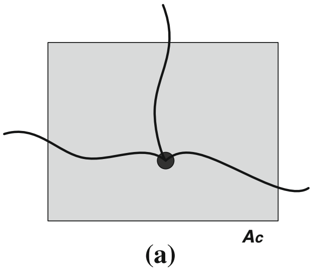


Fig. 4 The two possible connection statuses of the network segments inside a cloaked area A_c. (**a**) The network segments in A_c are totally connected. (**b**) The network segments in A_c comprise two separate subgroups

Figure 4 illustrates the two cases. Since we designed efficient solutions for the case in which the network edges inside A_c are fully connected, we can utilize the *divide and conquer strategy* to handle the case demonstrated in Fig. 4b. Consequently, we perform a pre-process for splitting the input cloaked area until each subregion contains only one connected network segment set. Then we execute our algorithms on each subregion separately and merge their results after the complete computation. For ease of presentation, we assume the pre-process has been executed in the following sections.

We observe that the number of data objects inside a cloaked area A_c can meet one of two conditions: (i) there are at least k objects inside A_c or (ii) there are fewer than k objects within A_c.

4.1.1 The Cloaked Area Contains at Least k Objects

Since Seg_i contains a limited number of network segments, PSNN starts the search on Seg_i for retrieving data objects inside A_c. If Seg_i covers more than or equal to k data objects, the search can be finished by expanding the intersection points in $\mathbb{T}$. First PSNN includes the data objects within A_c into the result set $\mathbb{R}$. Next, for each point t_i in $\mathbb{T}$, PSNN calculates the distance from t_i to its kth nearest object inside A_c as $Dist(t_i, n_k)$, and then expands outward from point t_i to search for data objects on Seg_o within distance $Dist(t_i, n_k)$ (the search upper bound). Consequently, we can cover the special case when q is located exactly at t_i. All the newly discovered data objects are inserted into the result set $\mathbb{R}$.

Figure 5 demonstrates an example where the circles represent the nodes in the modeling graph, triangles indicate data objects, and the gray rectangle stands for the cloaked area. Assuming that only one nearest neighbor is queried ($k = 1$), PSNN first retrieves n_1 by searching the network segments inside A_c. Since the number of data objects found in A_c is equal to 1, PSNN computes the network distance from t_1, t_2, and t_3 to n_1 as 2, 5, and 6, respectively. Afterwards, PSNN expands the search space outbound from the three intersection points according to their distance to n_1. No data object is found from the expansion of t_1 and t_3. The expansion of t_2 reaches n_2 and it is inserted into the result set. Consequently, the final search result set covers objects n_1 and n_2.

4.1.2 The Cloaked Area Contains Fewer than k Objects

If there are fewer than k data objects found on Seg_i, PSNN has to search the network segments which are outside of A_c. Since the points in $\mathbb{T}$ are all on the boundary of A_c, they

Algorithm 1 PSNN (q, k, A_c)

initialize R_i
locate the intersection points between A_c and the underlying network as $\mathbb{T} = \{t_1, \ldots, t_m\}$
$Seg_i = inside\_segments(A_c)$
search objects on Seg_i and insert the retrieved objects into $\mathbb{R}$
if Seg_i covers $\geq k$ objects **then**
  for $\forall t_i \in \mathbb{T}$ **do**
    expand t_i outward A_c for searching objects within distance $Dist(t_i, n_k)$
    insert any discovered objects into $\mathbb{R}$
  end for
else
  for $\forall t_i \in \mathbb{T}$ **do**
    $p_m p_n = find\_segment(t_i)$
    $R_i = R_i \cup find\_objects(p_m, p_n)$
    /* $\{n_1, \ldots, n_k\}$ = the k nearest objects in R_i sorted in ascending order, $n_j, n_{j+1} \ldots, n_k$ may be $\emptyset$, if $|R_i| < k$ */
    $Dist_{max} = Dist(t_i, n_k)$
    /* $Dist_{max} = \infty$, if $n_k = \emptyset$ */
    $Q = \langle (p_m, Dist(p_m, t_i)), (p_n, Dist(p_n, t_i)) \rangle$
    de-queue the node p in Q with smaller $Dist(p, t_i)$
    while $Dist(p, t_i) < Dist_{max}$ and the data set contains at least k objects **do**
      for each non-visited adjacent vertex p_x of p **do**
        $R_i = R_i \cup find\_objects(p_x, p)$
        update $Dist_{max}$ with sorted R_i
        en-queue$(p_x, Dist(p_x, t_i))$
      end for
      de-queue the next node p in Q
    end while
    $\mathbb{R} = \mathbb{R} \cup R_i$
  end for
end if
Sort $\mathbb{R}$ for removing duplicates
return $\mathbb{R}$

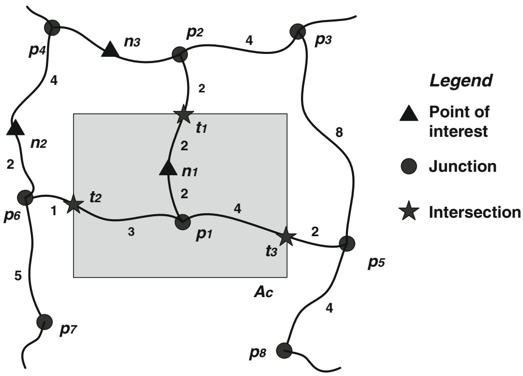
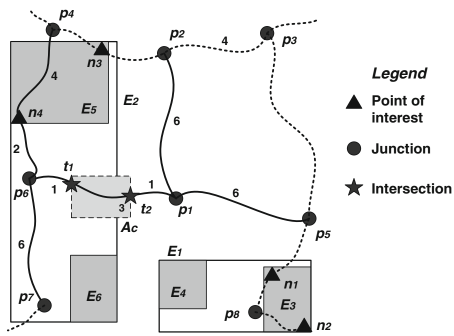


Fig. 5 Searching k network nearest neighbors with PSNN where the cloaked area contains k objects ($k = 1$ in this example)

determine the network search expansion upper bound. Consequently, PSNN executes the network expansion from all the intersection points for retrieving an inclusive result set. For every point t_i in $\mathbb{T}$, the PSNN algorithm first retrieves the network segment $p_m p_n$ which passes through t_i and searches all data objects on this segment. In the mean time, the two end points p_m and p_n are inserted into a queue Q with their distance to t_i. Afterward, if the search found fewer than k objects on $p_m p_n$ or the search retrieved no fewer than k objects but one end point of $p_m p_n$ whose distance to t_i is shorter than $Dist(t_i, n_k)$ (n_k is the kth

Fig. 6 Network range query with PSRQ



nearest neighbor of t_i), the end point p_x which is closer to t_i will be de-queued from Q and expanded. For each non-visited adjacent point p_y of p_x, PSNN searches $p_x p_y$, updates the result set, and inserts p_y with its distance to t_i into Q. Then the point in Q with the shortest distance to t_i is de-queued. The procedure repeats until k nearest neighbors of t_i are found and the k objects are inserted into the result set $\mathbb{R}$. PSNN repeats this process until it has expanded all the points in $\mathbb{T}$.

4.2 Privacy Protected Range Query on Spatial Networks

We define a spatial network range query as follows: given a query point q, a range value r, and an object data set $\mathbb{S}$, the query retrieves all elements of $\mathbb{S}$ that are within network distance r from q. For executing a *privacy protected spatial network range query* (PSRQ), first we have to locate the intersection points of the cloaked spatial area A_c and the underlying road network as a point set $\mathbb{T} = \{t_1, \ldots, t_m\}$. Then, PSRQ searches the network segments inside A_c and inserts the retrieved objects into $\mathbb{R}$. For each point t_i in $\mathbb{T}$ we can compute a set of *candidate segments* within network range r from t_i and then retrieve the data objects falling on these segments. Because the search range r could be a large number and it may cover many candidate segments, it is inefficient to check each candidate segment with the primitive operation *find_objects*(*segment*). Therefore, we utilize the intersection join function [3] for retrieving all intersection object pairs from the spatial network R-tree and the object R-tree. When reaching the leaf node level, PSRQ executes the plane-sweep method with the object R-tree nodes which intersect with the MBR of at least one candidate segment and stores the qualified objects into $\mathbb{R}$.

An example is demonstrated in Fig. 6. Assuming r is equal to a 7 unit distance, PSRQ joins the candidates segments (solid lines) with the object R-tree and retrieves leaf node E_5 intersecting with segment $p_4 p_6$. After executing the intersection test (plane-sweep method), PSRQ retrieves n_4 as the query result. The complete algorithm of PSRQ is shown in Algorithm 2.

5 Experimental Validation

In this section, we present extensive simulation results of our location cloaker and query processing algorithms. We implemented our cloaking technique and privacy protected query algorithms in a simulator to evaluate the performance of our approach. Our main objectives are to observe the influence of performance related factors (e.g., cloaked region size) on the system and to test the feasibility of our approach. Performance is measured in terms of the

Algorithm 2 PSRQ (q, r, A_c)

locate the intersection points between A_c and the underlying network as $\mathbb{T} = \{t_1, \ldots, t_m\}$
$Seg_i = inside\_segments(A_c)$
search objects on Seg_i and insert the retrieved objects into $\mathbb{R}$
for each point t_i in $\mathbb{T}$ **do**
  Compute all the candidate segments within distance r from t_i as C
  Intersection join C with the object R-tree for finding intersection leaf nodes
  for each retrieved leaf node E_i **do**
    R_{ti} = intersection test of the intersected segments with data objects in E_i
  end for
  $\mathbb{R} = \mathbb{R} \cup R_{ti}$
end for
Sort $\mathbb{R}$ for removing duplicates
return $\mathbb{R}$

cloaked region size, result set size, and CPU time. All simulation results were recorded after the system model reached steady state.

5.1 Simulator Implementation

Our simulator consists of three main components, the *mobile environment*, the *location cloaker*, and the *location-based service provider*. For the mobile environment, we utilized the *network-based moving objects generation framework* [2] to generate a set of mobile users and the underlying road network inside a geographical area, measuring 10 miles by 10 miles. Each mobile user is an independent object which encapsulates all its related parameters (e.g., its current speed and destination). We implemented the location cloaker as a new module for interacting with mobile users to anonymize spatial queries in the framework. Our privacy protected query approaches (Sects. 4.1 and 4.2) were also implemented inside the framework as new functions and play the role of the LBSP. We obtained our road network data of the City of Los Angeles and Riverside County from the TIGER/Line street vector data available from the U.S. Census Bureau.

5.2 Exploring Performance Influencing Factors

We are interested in the effect of three major performance influencing factors: the cloaked region size, the number of k, and the Point of Interest (POI) number, of spatial queries. We experimented with both nearest neighbor queries and range queries with these factors as follows.

Effect of the Cloaked Region Size. We increased the cloaked region size from 2% to 10% of the whole experimental region and the cloaked region size also reflects the influences of the two privacy policy settings—K and R_{min}. Figure 7 demonstrates the result set size and query processing time with different cloaked region sizes. Since a bigger cloaked region usually intersects with more underlying network segments, it generates a larger candidate result set and takes longer to process. As shown in Fig. 7a, the curve of PSRQ remarkably increases because the result set covers all the POIs within the cloaked region and the search range r. In contrast, PSNN removes duplicated objects from $\mathbb{R}$. In addition, the query processing time of PSNN increases notably with an enlarged cloaked region size as illustrated in Fig. 7b.

Effect of k. We tested the impact of varying the number of requested nearest neighbors, i.e., k. We altered k in the range from 4 to 20. As shown in Fig. 8, the result set size grows when we

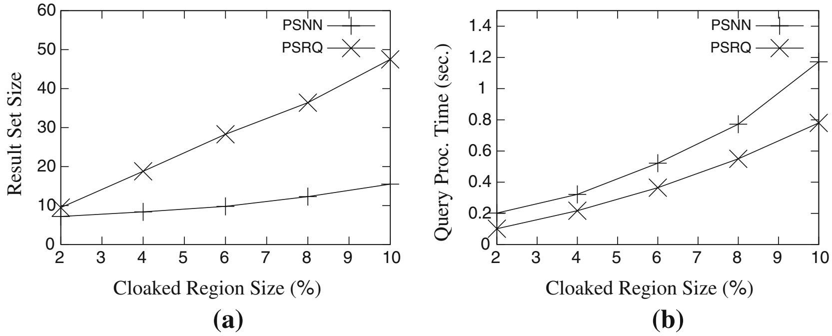


Fig. 7 The effect of the cloaked region size. (**a**) Result set size. (**b**) Query processing time

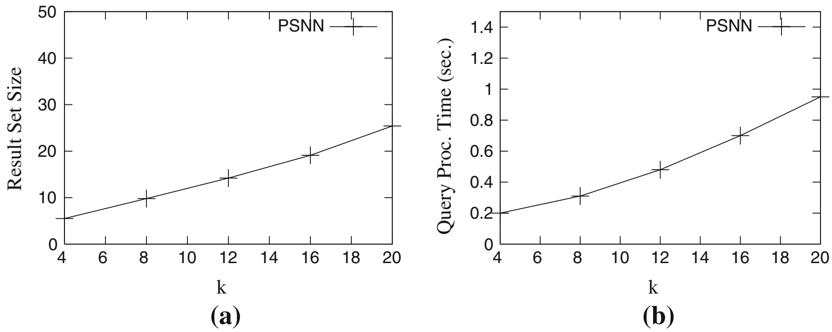


Fig. 8 The effect of k. (**a**) Result set size. (**b**) Query processing time

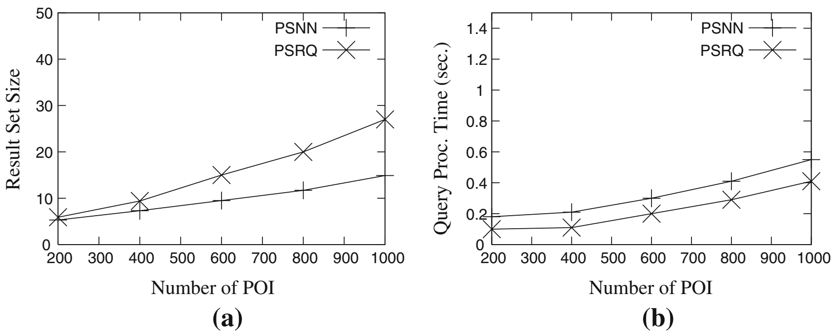


Fig. 9 The effect of the number of POIs. (**a**) Result set size. (**b**) Query processing time

raised k from 4 to 20 and we also observe that the result set increases super-linearly when k is a relatively large number. For example, the result set covers around 25% more objects than the queried k number when k is equal to 20. This factor has a similar super-linear influence on the query processing time as demonstrated in Fig. 8b.

Effect of the Number of POI. To see the effect of varying the total POI number, we increased the total POI number from 200 to 1,000 in the simulation environment. Figure 9 illustrates the result set size and query processing time of PSNN and PSRQ with increasing POI numbers. We note that with a higher POI density the result set of PSRQ increases conspicuously compared with PSNN. This is expected, since PSRQ has to report all the objects inside the cloaked region. In addition, the query processing time of PSNN is always longer than that of PSRQ.

6 Conclusions

In this paper we present an advanced location cloaker and two novel algorithms for processing *k* nearest neighbor queries and range queries on spatial networks with privacy protection. The main idea is to hide the exact mobile user location with a cloaked region. The cloaked region covers the query requester and at least $K-1$ other users based on the K-anonymity concept. The spatial queries are executed based on both the cloaked region and the underlying networks. A candidate result set will be returned to the requesting user who filters out the exact answer. Our comprehensive simulations demonstrate the efficiency of our methods. We plan to extend the proposed solutions to support spatial network based query processing with other privacy protection models (e.g., dummy-based solutions and transformation-based solutions).

References

1. Aref, W. G., & Samet, H. (1990). Efficient processing of window queries in the pyramid data structure. In *PODS* (pp. 265–272).
2. Brinkhoff, T. (2002). A framework for generating network-based moving objects. *GeoInformatica, 6*(2), 153–180.
3. Brinkhoff, T., Kriegel, H.-P., & Seeger, B. (1993). Efficient processing of spatial joins using R-trees. In *SIGMOD Conference* (pp. 237–246).
4. Cheng, R., Kalashnikov, D. V., & Prabhakar, S. (2003). Evaluating probabilistic queries over imprecise data. In *SIGMOD Conference* (pp. 551–562).
5. Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik, 1*, 269–271.
6. Gedik, B., & Liu, L. (2005). Location privacy in mobile systems: A personalized anonymization model. In *25th International Conference on Distributed Computing Systems* (pp. 620–629).
7. Gruteser, M., & Grunwald, D. (2003). Anonymous usage of location-based services through spatial and temporal cloaking. In *MobiSys*.
8. Gruteser, M., & Liu, X. (2004). Protecting privacy in continuous location-tracking applications. *IEEE Security & Privacy, 2*(2), 28–34.
9. Guttman, A. (1984). R-trees: A dynamic index structure for spatial searching. In *SIGMOD Conference* (pp. 47–57).
10. Hjaltason, G. R., & Samet, H. (1999). Distance browsing in spatial databases. *ACM Transactions on Database Systems, 24*(2), 265–318.
11. Kolahdouzan, M. R., & Shahabi, C. (2004). Voronoi-based K nearest neighbor search for spatial network databases. In *VLDB* (pp. 840–851).
12. Ku, W.-S., Zimmermann, R., Peng, W.-C., & Shroff, S. (2007). Privacy protected query processing on spatial networks. In *ICDE Workshops* (pp. 215–220).
13. Mokbel, M. F. (2006). Towards privacy-aware location-based database servers. In *ICDE Workshops* (p. 93).
14. Mokbel, M. F., Chow, C.-Y., & Aref, W. G. (2006). The new casper: Query processing for location services without compromising privacy. In *VLDB* (pp. 763–774).
15. Papadias, D., Zhang, J., Mamoulis, N., & Tao, Y. (2003). Query processing in spatial network databases. In *VLDB* (pp. 802–813).
16. Roussopoulos, N., Kelley, S., & Vincent, F. (1995). Nearest neighbor queries. In *SIGMOD Conference* (pp. 71–79).
17. Schilit, B. N., Hong, J. I., & Gruteser, M. (2003). Wireless location privacy protection. *IEEE Computer, 36*(12), 135–137.
18. Sweene, L. (2002). k-Anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 10*(5), 557–570.
19. Tanimoto, S. L., & Pavlidis, T. (1975). A hierarchical data structure for picture processing. *Computer Graphics and Image Processing, 4*(2), 104–113.
20. Voelcker, J. (2006). Stalked by satellite: An alarming rise in GPS-enabled harassment. *IEEE Spectrum, 47*(7), 15–16.

Author Biographies

Wei-Shinn Ku received his Ph.D. degree in computer science from the University of Southern California (USC) in 2007. He also obtained both the M.S. degree in computer science and the M.S. degree in Electrical Engineering from USC in 2003 and 2006 respectively. He is an Assistant Professor with the Department of Computer Science and Software Engineering at Auburn University. His research interests include spatial and temporal data management, mobile data management, geographic information systems, and security and privacy. He has published more than 30 research papers in refereed international journals and conference proceedings. He is a member of the ACM and the IEEE.

Yu Chen received his M.S. and Ph.D. degrees in Electrical Engineering from the University of Southern California (USC) in 2002 and 2006, respectively. He is an Assistant Professor of Electrical and Computer Engineering at SUNY—Binghamton. His research interests include network security, security and privacy in distributed systems and pervasive computing environments, Internet infrastructure security, and reconfigurable hardware based security solutions. He is a member of the ACM, the IEEE and the SPIE.

Roger Zimmermann received his M.S. and Ph.D. degrees from the University of Southern California (USC) in 1994 and 1998. He is currently an Associate Professor with the Department of Computer Science at the National University of Singapore (NUS). He is also an investigator with the Interactive and Digital Media Institute at NUS. His research interests are in the areas of distributed and peer-to-peer systems, collaborative environments, streaming media architectures, geospatial database integration, and mobile location-based services. He has co-authored a book, two patents and more than 90 conference publications, journal articles and book chapters. He is a Senior Member of the IEEE and a member of ACM.