Preventing Location-Based Identity Inference in Anonymous Spatial Queries

Panos Kalnis, Gabriel Ghinita, Kyriakos Mouratidis, and Dimitris Papadias

Abstract— **The increasing trend of embedding positioning capabilities (e.g., GPS) in mobile devices facilitates the widespread use of Location Based Services. For such applications to succeed, privacy and confidentiality are essential. Existing privacy-enhancing techniques rely on encryption to safeguard communication channels, and on pseudonyms to protect user identities. Nevertheless, the query contents may disclose the physical location of the user.**

In this paper, we present a framework for preventing location-based identity inference of users who issue *spatial queries* to Location Based Services. We propose transformations based on the well-established *K*-anonymity concept to compute exact answers for range and nearest neighbor search, without revealing the query source. Our methods optimize the entire process of anonymizing the requests and processing the transformed spatial queries. Extensive experimental studies suggest that the proposed techniques are applicable to real-life scenarios with numerous mobile users.

Index Terms— **Privacy, Anonymity, Location Based Services, Spatial Databases, Mobile Systems.**

I. Introduction

In recent years, positioning devices (e.g., GPS) have gained tremendous popularity. Navigation systems are already widespread in the automobile industry and, together with wireless communications, facilitate exciting new applications. General Motor's OnStar system, for example, supports on-line rerouting to avoid traffic jams and automatically alerts the authorities in case of an accident. More applications based on the users' location are expected to emerge with the arrival of the latest gadgets (e.g., iPAQ hw6515, Mio A701), which combine the functionality of a mobile phone, PDA and GPS receiver. For such applications to succeed, the privacy and confidentiality issues are of paramount importance.

Consider that Bob uses his GPS-enabled mobile phone to find the nearest betting office. This query can be answered by a Location Based Service (LBS) in a publicly available web server (e.g., Google Maps). Since Bob does not want to disclose to Alice his gambling habits, instead of directly sending the query to the LBS, he uses an *anonymizer*, which is a trusted server (services for anonymous web surfing are commonly available nowadays). He establishes a secure connection (e.g., SSL) with the anonymizer, which removes the user id from the query and forwards it to the LBS. The answer from the LBS is also routed to Bob through the anonymizer.

Nevertheless, the query itself unintentionally reveals sensitive information. In our example, the LBS requires the coordinates of the user in order to process the nearest neighbor (NN) query. Since the LBS is not trusted, Alice can collaborate with the LBS and acquire the location of Bob and his query result (i.e., betting office). The next step is to relate the coordinates to a specific user. Alice may choose from a variety of techniques such as physical observation of Bob, triangulating his mobile phone's signal[1], or consulting publicly available databases. If, for instance, Bob uses his phone within his residence, Alice can easily convert the coordinates to a street address (most on-line maps provide this service) and relate the address to Bob by accessing an on-line white pages service.

For a broad discussion on the risks of revealing sensitive information in location-based services see [7]. In practice, users would be reluctant to access a service that may disclose their political/religious affiliations or alternative lifestyles. Furthermore, given that the LBS is not trusted, users might be hesitant to ask innocuous queries such as “find the closest gas station” or “which are the restaurants in my vicinity” since, once their identity is revealed, they may face unsolicited advertisements, e-coupons, etc. Motivated by this fact, we develop methods to protect the privacy of users issuing spatial queries against location-based attacks. Specifically, we prevent an attacker from inferring the identity of the query source by adapting the well established *K*-anonymity technique to the spatial domain.

K-anonymity [25], [27] has been used for publishing microdata, such as census, medical and voting registration data. A dataset is said to be *K-anonymized*, if each record is indistinguishable from at least *K*-1 other records with respect to certain identifying attributes. In the context of location based services, the *K*-anonymity concept translates as follows: given a query, guarantee that an attack based on the query location cannot identify the query source with probability larger than 1/*K*, among other *K*-1 users. Most of the existing work adopts the framework of Figure 1a. In this framework, a user sends his location and query to the anonymizer through a secure connection. The anonymizer removes the id of the user and transforms his location through a technique called *cloaking*. Cloaking hides the actual location by a *K-anonymizing spatial region* (*K*-ASR or ASR), which is an area that encloses the client that issued the query, as well as at least *K*-1 other users. The anonymizer then sends the ASR to the LBS, which returns to the anonymizer a set of *candidate results* that satisfy the query condition for any possible point in the ASR. The

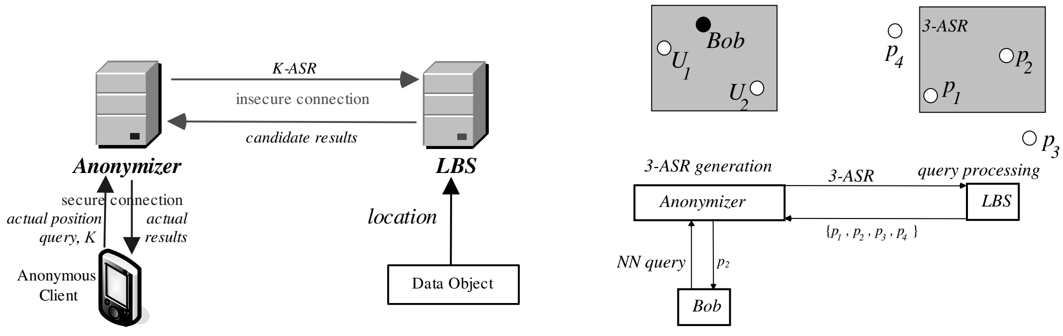
P. Kalnis and G. Ghinita are with the Department of Computer Science, National University of Singapore, COM1-Law Link, Singapore 117590. Email: {kalnis, ghinitag}@comp.nus.edu.sg.

K. Mouratidis is with the School of Information Systems, Singapore Management University, 80 Stanford Road, Singapore 178902. E-mail: kyriakos@smu.edu.sg.

D. Papadias is with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong. E-mail: dimitris@cs.ust.hk.

[1] Phone companies can estimate the location of the user within 50-300 meters, as required by the US authorities (E911).

Digital Object Indentifier 10.1109/TKDE.2007.190662 1041-4347/$25.00 © 2007 IEEE



(a) General framework

(b) Example of NN query

Fig. 1. Framework and example for spatial K-anonymity

LBS may be compromised, i.e., an adversary may have complete knowledge of all queries received by the LBS.

Figure 1b illustrates this process in detail, by continuing the running example. Bob forwards his request to the anonymizer, together with his anonymity requirement K. Assuming that K=3, the anonymizer generates a 3-ASR (shaded rectangle) that contains Bob and two other users U_1, U_2 (the anonymizer knows the exact locations of all users). Then, it sends this 3-ASR to the LBS, which finds all betting offices that can be the NN of any point in the 3-ASR (recall that the LBS does not know where Bob is). This candidate set (i.e., p_1, p_2, p_3, p_4) is returned to the anonymizer, which filters the false hits and forwards the actual NN (in this case p_2) to Bob. Even if Alice knows the location of Bob and the other users, she can only ascertain that the query originated from Bob with probability $1/3$.

Existing methods for spatial K-anonymity (reviewed in Section II) have at least one of the following shortcomings: *(i)* They compromise the query issuer's identity for certain user location distributions. *(ii)* They sacrifice quality of service (QoS), i.e., some queries must be delayed or dropped. *(iii)* They are inefficient, i.e., they generate large ASRs. *(iv)* They focus exclusively on cloaking mechanisms and lack algorithms for query processing at the LBS. In this paper we aim at solving these problems through a comprehensive set of techniques. Specifically, we propose two cloaking algorithms: *Nearest Neighbor Cloak* that significantly outperforms the existing techniques in terms of efficiency but has similar anonymity problems for some distributions, and *Hilbert Cloak* that never reveals the query source, independently of the user location distribution. Moreover, we address the issue of anonymized query processing at the LBS. Specifically, we adopt an existing algorithm to compute the k nearest neighbors[2] (kNN) of rectangular regions, as opposed to points and develop a novel algorithm to compute the kNN of circular regions, which reduces the number of redundant results, hence the communication cost between the anonymizer and the LBS.

The rest of the paper is organized as follows: Section II presents the related work. Next, Section III deals with the construction of the K-ASR at the anonymizer, followed by Section IV where we describe the query processing algorithms at the LBS. The results of our experiments are illustrated in Section V. Finally, Section VI concludes the paper and presents directions for future work.

II. Related work

Section II-A discusses K-anonymity in relational databases and Section II-B presents privacy-preserving methods for location-based services. Section II-C overviews related spatial query processing techniques.

A. *K-anonymity in Relational Databases*

Anonymity was first discussed in relational databases, where published data (e.g., census, medical) should not be linked to specific persons. Adam and Wortmann [1] survey methods for computing aggregate functions (e.g., *sum, count*) under the condition that the results do not reveal any specific record. Agrawal and Srikant [3] compute value distributions, suitable for data mining, in confidential fields. Recent work has focused on K-anonymity as defined in [25], [27]: a relation satisfies K-anonymity if every tuple is indistinguishable from at least K-1 other tuples with respect to a set of *quasi-identifier* attributes. Quasi-identifiers are attributes (e.g., date of birth, gender, zip code) that can be linked to publicly available data to identify individuals. Records with identical quasi-identifiers form an anonymized group. Two techniques are used to transform a relation to a K-anonymized one: *suppression*, where some of the attributes or tuples are removed and *generalization*, which involves replacing specific values (e.g., phone number) with more general ones (e.g., only area code). Both methods lead to information loss. Algorithms for anonymizing an entire relation, while preserving as much information as possible, are discussed in [4], [19]. Xiao and Tao [31] consider the case where each individual requires a different degree of anonymity, whereas Aggarwal [2] shows that anonymizing a high-dimensional relation leads to unacceptable loss of information due to the dimensionality curse. Machanavajjhala et al. [20] propose ℓ-diversity, an anonymization method that prevents sensitive attribute disclosure by providing diversity among the sensitive attribute values of each anonymized group. Finally, [14] employs multi-dimensional to 1-D transformations to solve efficiently the K-anonymity and ℓ-diversity problems.

B. *K-anonymity in Location-Based Services*

Most previous work on location-based services adopts the concept of K-anonymity using the framework of Figure 1: a user sends his position, query and K to the anonymizer, which removes the id of the user and transforms his location through cloaking. The generated K-ASR is forwarded to the LBS which processes it and returns a set of candidates, containing the actual results and

[2] Note that k, the number of nearest neighbors is different than K, the degree of anonymity.

false hits. The first cloaking[3] technique, called *Interval Cloak* [15] is based on quadtrees. A quadtree [26] recursively partitions the space into quadrants until the points in each quadrant fit in a page/node. Figure 2 shows the space partitioning and a simple quadtree assuming that a node contains a single point. The anonymizer maintains a quadtree with the locations of all users. Once it receives a query from a user U, it traverses the quadtree (top-down) until it finds the quadrant that contains U and fewer than K-1 users. Then, it selects the parent of that quadrant as the K-ASR and forwards it to LBS.

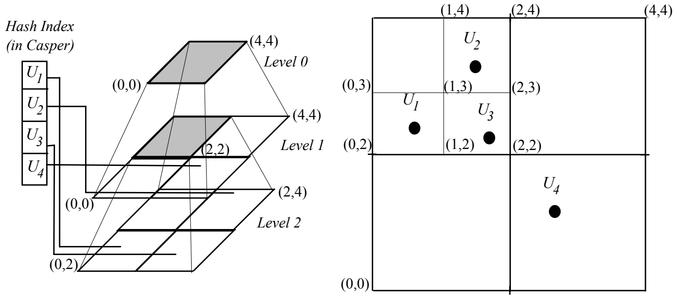


Fig. 2. Example of *Interval Cloak* and *Casper*

Assume that in Figure 2, U_1 issues a query with K=2. Quadrant[4] $\langle(0,2),(1,3)\rangle$ contains only U_1, so its parent $\langle(0,2),(2,4)\rangle$ becomes the 2-ASR. Note that the ASR may contain more users than necessary; in this example it includes U_1, U_2, U_3, although 2 users would suffice for the privacy requirements. A large ASR burdens the query processing cost at the LBS and the network overhead for transferring a large number of candidate results from the LBS to the anonymizer. In order to overcome this problem, Gruteser and Grunwald [15] combine *temporal cloaking* with spatial cloaking, i.e., the query may wait until K (or more) objects fall in the user's quadrant. In our example, the query of U_1 will be executed when a second user enters $\langle(0,2),(1,3)\rangle$, in which case $\langle(0,2),(1,3)\rangle$ is the 2-ASR sent to the LBS.

Similar to *Interval Cloak*, *Casper* [23] is based on quadtrees. The anonymizer uses a hash table on the user id pointing to the lowest-level quadrant where the user lies. Thus, each user is located directly, without having to access the quadtree top-down. Furthermore, the quadtree can be adaptive, i.e., contain the minimum number of levels that satisfies the privacy requirements. In Figure 2, for instance, the second level for quadrant $\langle(0,2),(2,4)\rangle$ is never used for $K \geq 2$ and can be omitted. The only difference in the cloaking algorithms of *Casper* and *Interval Cloak* is that *Casper* (before using the parent node as the K-ASR) also considers the neighboring quadrants at the same level of the tree. Assume again that in Figure 2 U_1 issues a query and K=2. *Casper* checks the content of quadrants $\langle(1,2),(2,3)\rangle$ and $\langle(0,3),(1,4)\rangle$. Since the first one contains user U_3, the 2-ASR is set to $\langle(0,2),(2,3)\rangle$, which is half the size of the 2-ASR computed by *Interval Cloak* (i.e., $\langle(0,2),(2,4)\rangle$).

In *Clique Cloak* [11], each query defines an axis-parallel rectangle whose centroid lies at the user location and whose extents are $\Delta x, \Delta y$. Figure 3 illustrates the rectangles of three queries located at U_1, U_2, U_3, assuming that they all have the same Δx and Δy. The anonymizer generates a graph where a vertex represents a query: two queries are connected if the corresponding users fall in the rectangles of each other. Then, the graph is searched for cliques of K vertices and the minimum bounding rectangle (MBR) of the corresponding rectangles forms the ASR sent to the LBS. Continuing the example of Figure 3, if K=2, U_1 and U_2 form a 2-clique and the MBR of their respective rectangles is forwarded so that both queries are processed together. On the other hand, U_3 cannot be processed immediately, but it has to wait until a new query (generating a 2-clique with U_3) arrives. *Clique Cloak* allows users to specify a temporal interval Δt such that, if a clique cannot be found within Δt, the query is rejected. The selection of appropriate values for $\Delta x, \Delta y, \Delta t$ is not discussed in [11].

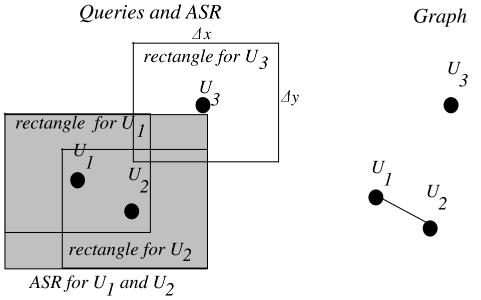


Fig. 3. Example of *Clique Cloak*

Probabilistic Cloaking [8] preserves the privacy of locations without applying spatial K-anonymity. Instead, *(i)* the ASR is a closed region around the query point, which is independent of the number of users inside and *(ii)* the location of the query is uniformly distributed in the ASR. Given an ASR, the LBS returns the probability that each candidate result satisfies the query, based on its location with respect to the ASR. Finally, location anonymity has also been studied in the context of related problems. Kamat et al. [18] propose a model for sensor networks and examine the privacy characteristics of different sensor routing protocols. Hoh and Gruteser [16] describe techniques for hiding the trajectory of users in applications that continuously collect location samples. Ghinita et al. [12], [13] and Chow et al. [10] study spatial cloaking in peer-to-peer systems.

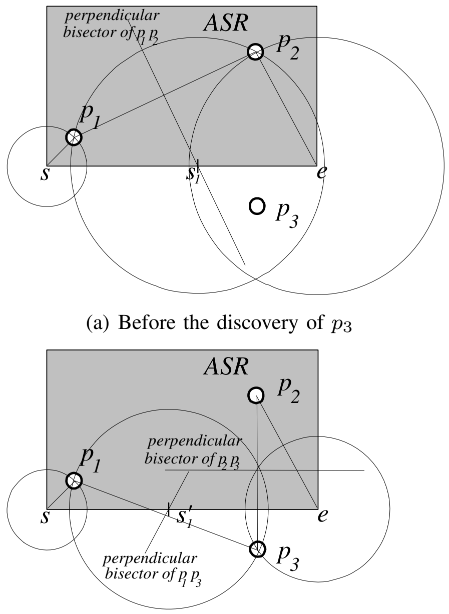
C. Related Spatial Query Processing Techniques

The LBS maintains the locations of points-of-interest and answers cloaked queries. The most common spatial queries, and the focus of the existing systems, are ranges and nearest neighbors. While the cloaking mechanism at the anonymizer is independent of the query type, query processing at the LBS depends on the query. Range queries are usually straightforward; assume that a user U wants to retrieve the data objects within distance d from his current location. Instead of the position of U, the LBS receives (from the anonymizer), an ASR that contains U (as well as several other users) and d. In order to compute the candidate results, the LBS extends the ASR by d on all dimensions and searches for all objects in the extended ASR. The set of candidates is returned to the anonymizer which filters out false hits and returns the actual result to U.

The processing of NN queries is more complicated. If the ASR is an axis-parallel rectangle (as in *Interval Cloak*, *Casper* and

[3] Beresford and Stajano [6] introduce the concept of mix zone, which is similar to the K-ASR, but do not provide concrete algorithms for spatial cloaking.

[4] We use the coordinates of the lower-left and upper-right points to denote a quadrant.


perpendicular bisector of p_1p_2
ASR
p_2
p_1
s
s_1
e
p_3
ASR
p_2
p_1
perpendicular bisector of p_2p_3
s
s'_1
e
perpendicular bisector of p_1p_3
p_3

(a) Before the discovery of p_3

(b) After the discovery of p_3

Fig. 4. Example of continuous NN search

Clique Cloak), then the candidate results can be retrieved using *range nearest neighbor* search [17], which finds the NN of any point inside a rectangular range. Assume the running example of Figure 1b, where the ASR is the shaded rectangle. The LBS must return the NN of every possible location in the ASR. Such candidate data points lie inside (e.g., p_1, p_2), or outside the ASR (e.g., p_3, p_4). For instance, p_3 would be the NN for a user at the lower-right corner of the ASR.

Figure 4 shows the application of range nearest neighbor search in the above example. The initial set of candidates contains all points (p_1, p_2) inside the input range (i.e., the ASR). Then, four *continuous NN* (CNN) queries [29], one for each side of the ASR, retrieve the remaining candidates. Consider, for instance, the CNN query for the bottom side se. The initial candidates split se into two intervals: ss_1 and s_1e, where s_1 is the point where the perpendicular bisector of p_1p_2 intersects se. Currently, the NN of every point in ss_1 is p_1, whereas the NN of every point in s_1e is p_2. The three *vicinity circles* in Figure 4a, are centered at s, s_1, e and their radii equal the distances between s and p_1, s_1 and p_1 (or p_2), and e and p_2, respectively. The only data points that can be closer to se (than p_1 and p_2) must fall inside some vicinity circle.

Continuing the example, p_3 falls inside the last two vicinity circles and updates the result as shown in Figure 4b. Specifically, s'_1 is the point where the perpendicular bisector of p_1p_3 intersects se: p_1 becomes the NN of every point in ss'_1, and p_3 the NN of every point in s'_1e. Note that the vicinity circles shrink as new data points are discovered. The process terminates when no more points are found within the vicinity circles. It can be shown [17] that four CNN queries for the four sides of the ASR find all candidate objects. A similar technique (also for rectangular ranges) is presented for *Casper* in [23]; in Section IV, we develop a method capable of processing circular ranges. Next, we proceed with cloaking techniques at the anonymizer.

III. The Anonymizer

Section III-A presents the basic assumptions and goals of our techniques. Sections III-B and III-C propose two novel cloaking techniques, *Nearest Neighbor Cloak* and *Hilbert Cloak*, respectively.

A. Assumptions and Goals of Spatial Anonymization

The anonymizer is a trusted server, which collects the current location of users and anonymizes their queries. Each query has a required degree of anonymity *K*, which ranges between 1 (no privacy requirements) and the user cardinality (maximum privacy). We assume that an attacker has complete knowledge of *(i)* all the ASRs ever received at the LBS, *(ii)* the cloaking algorithm used by the anonymizer, and *(iii)* the locations of all users. The first assumption states that either the LBS is not trusted (e.g., a commercial service that collects unauthorized information about its clients for unsolicited advertisements), or the communication channel between the anonymizer and the LBS is not secure. The second assumption is common in the security literature since the data privacy algorithms are usually public.

The third assumption is motivated by the fact that users may often (or always) issue queries from the same locations (home, office), which may be easily identified through public databases, telephone directories, etc. Furthermore, they may reveal their locations by issuing queries without privacy requirements. In scenarios with highly mobile users, the attacker may not be able to learn exact user locations. However, one can argue that in these cases spatial *K*-anonymity is not important, because *(i)* the user ids are removed by the anonymizer anyway, and *(ii)* a query at a random position does not necessarily reveal information about the identity of the corresponding user. However, in practice, a determined attacker may be able to acquire (through triangulation, public databases, physical observation, etc.) the locations of at least a few users in the vicinity of the targeted victim.

Similar to existing work on LBS query privacy [10], [15], [23] we focus on *snapshot* queries, where the attacker uses current data, but not historical information about movement and behavior patterns of particular clients (e.g., a user often asking a particular query at a certain location or time). This assumption is reasonable in practice, because if a client obtains the items of interest (e.g., the closest restaurant), it is unlikely to ask the same query from the same location again in the future. We also assume that the attacker does not have a priori knowledge of the user query frequencies (i.e., a query may originate from any user with equal probability). Furthermore, the value of *K* is not subject to attacks since it is transferred from the client to the anonymizer through a secure channel.

Given a query, the anonymizer removes the user id, applies cloaking to hide the user's location through an ASR, and forwards the ASR to the LBS. The cloaking algorithm is said to preserve spatial *K*-anonymity, if the probability of the attacker pinpointing the query source under the above assumptions does not exceed 1/*K*.

Note that simply generating an ASR that includes *K* users is not sufficient for spatial *K*-anonymity. Consider for instance, a naïve algorithm, called *Center Cloak* (*CC*) in the sequel, which given a query from U, finds his *K*-1 closest users, and sets the ASR as the minimum bounding rectangle (MBR) or circle (MBC) that encloses them. In fact, a similar technique is proposed in

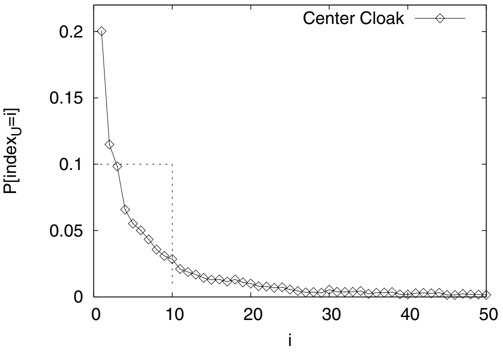


Fig. 5. Distance from MBR center for *Center Cloak* (*K*=10)

[10] for anonymization in peer-to-peer systems, i.e., the *K*-ASR contains the query issuing peer and its *K*-1 nearest nodes. *CC* is likely to disclose the location of *U* under the *center-of-ASR* attack. Specifically, let $index_U$ be the position of *U* in the sequence of users enclosed by the *K*-ASR, sorted in ascending order of their distance from the center of the *K*-ASR; for example, if $index_U =$ 1, then *U* is the closest user to the center. The *center-of-ASR* attack is successful if $P[index_U = 1] >$ 1/*K*, i.e., if the probability of *U* being the closest user to the center exceeds 1/*K*.

Figure 5 shows the distribution of the positions of *U* inside an MBR enclosing its 9 NNs (for details of the experimental setting, see Section V). In most cases, *U* is close to the center of the 10-ASR (i.e., $P[index_U = 1] > 1/10$). Hence, an attacker with knowledge of the cloaking algorithm (assumption *ii*) may easily pinpoint *U* as the query source. Note that, since the MBR may enclose more than 10 users it is possible to get $P[index_U = i] > 0$ for $i > 10$. The dashed line in the graph corresponds to the “flat” index distribution obtained by an ideal anonymization technique, which would always generate 10-ASRs with exactly 10 users.

In addition to the preservation of spatial *K*-anonymity, we define the following objectives of cloaking:

1) The generated ASR should be as small as possible.
2) The cloaking algorithm should not compromise the quality of service (QoS).
3) The ASR should not reveal the exact location of any user.

Goal 1 is induced by the fact that a large ASR incurs higher processing overhead (at the LBS) and network cost (for transferring a large number of candidate results from the LBS to the anonymizer). In real-world services, users may be charged depending on the overhead that the anonymization requirements impose on the system. Note that, as long as the anonymity requirements of the user are satisfied, the size of the ASR is irrelevant in terms of *K*-anonymity. Goal 2 states that systems that delay or reject service requests, such as *Clique Cloak* [11], are unacceptable. In general, since temporal cloaking compromises QoS, we focus our attention on spatial cloaking. Goal 3 ensures that the anonymizer does not help the attacker obtain the locations of users through the cloaking algorithm (although, as discussed before, he may obtain them through other means). The disclosure of exact locations by a service is undesirable to most users (independently of their queries), and in some cases forbidden by law. As an example, consider that the anonymizer picks *K*-1 random users and sends *K* independent queries (including the real one) to the LBS. This method achieves spatial *K*-anonymity, but reveals the exact locations of *K* users. Furthermore, it has several efficiency problems: *(i)* depending on the value of *K*, a potentially large number of locations are transmitted to the LBS and *(ii)* the LBS has to process *K* independent queries and send back all their results.

Let U be the user issuing a query. The proposed cloaking algorithms first generate an anonymizing set (AS) that contains U and at least *K*-1 users in his vicinity. The ASR is an area that encloses all users in AS. Although the ASR can have arbitrary shape, we use minimum bounding rectangles (MBR) or circles (MBC) because they incur small network overhead (when transmitted to the LBS) and facilitate query processing. Note that, in addition to AS, the ASR may enclose some additional users that fall in the corresponding MBR or MBC.

B. Nearest Neighbor Cloak

The first algorithm, *Nearest Neighbor Cloak* (*NNC*), is a randomized variant of *Center Cloak*, and is not vulnerable to *center-of-ASR* attacks. Given a query from U, *NNC* first determines the set S_0 containing U and his *K*-1 nearest users. Then, it selects a random user U_i from S_0 (the probability of selecting the initial user U is 1/*K*) and computes the set S_1, which includes U_i and his *K*-1 NNs. Finally, *NNC* obtains $S_2 = S_1 \cup U$, i.e., S_2 corresponds to the anonymizing set. This step is essential, since U is not necessarily among the NNs of U_i. The *K*-ASR is the MBR or MBC enclosing all users in S_2.

Figure 6 shows an example of *NNC*, where U_1 issues a query with *K*=3. The 2 NNs of U_1 are U_2, U_3, and $S_0 = \{U_1, U_2, U_3\}$. *NNC* randomly chooses U_3 and issues a 2-NN query, forming $S_1 = \{U_3, U_4, U_5\}$. The 3-ASR is the MBR enclosing $S_2 = \{U_1, U_3, U_4, U_5\}$. *NNC* can be used with variable values of *K*. It is not vulnerable to the *center-of-ASR* attack since the probability of U being near the center of the *K*-ASR is at most 1/*K* (due to the random choice). Furthermore, as we show in the experimental evaluation, the ASR is much smaller than that of *Interval Cloak* and *Casper*.

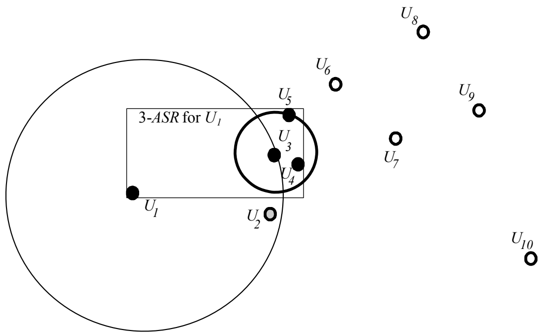


Fig. 6. Example of *NNC*

However, *NNC*, as well as *Interval Cloak* and *Casper*, may compromise location anonymity in the presence of outliers. Consider that in Figure 6, an adversary knows the locations of the users and the value of *K*. Then, he can be sure that the query originated from U_1 because if it were issued by any other user (U_3, U_4, U_5) in the 3-ASR, the ASR would not contain U_1. For *Interval Cloak* and *Casper* we use the example of Figure 2 assuming that *K*= 2. If a query originates from U_1, U_2, or U_3, the 2-ASR of *Interval Cloak* is quadrant $\langle(0, 2), (2, 4)\rangle$. Similarly, the 2-ASR of *Casper* is the concatenation of two sibling quadrants at level 2 (e.g., $\langle(0, 2), (1, 3)\rangle$ and $\langle(1, 2), (2, 3)\rangle$). On the other

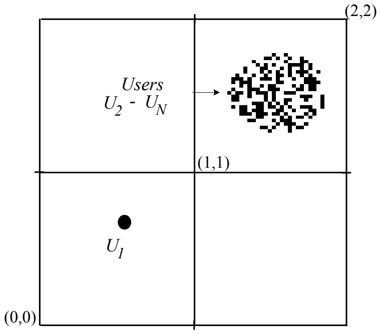


Fig. 7. Location anonymity compromise in the presence of outliers

hand, if a query originates from U_4, the 2-ASR is the entire data-space $\langle(0,0),(4,4)\rangle$) for both *Interval Cloak* and *Casper*. Thus, an attacker can identify U_4 for all 2-ASRs that cover the entire data-space.

For illustration purposes, in the above examples we assumed that the attacker knows K, although as discussed in Section III-A, K is not subject to attacks. Nevertheless, even for variable and unknown K, the presence of outliers may compromise spatial anonymity. We demonstrate the problem for *Interval Cloak* and *Casper* using Figure 7. There is a single user U_1 in quadrant $\langle(0,0),(1,1)\rangle$ and $N-1$ users in $\langle(1,1),(2,2)\rangle$, where N is the user cardinality. Quadrant $\langle(1,1),(2,2)\rangle$ may be subdivided further, but this is not important for our discussion. Each user has equal probability to issue a query, and the degree of anonymity required by different queries distributes uniformly in the range $[1,N]$. The term *event* signifies the issuance of a query with anonymity degree K at a random user U. Then, an ASR covering the entire data space is generated by *(i)* a query originating from U_1 and $2 \leq K \leq N$ (i.e., $N-1$ events), or *(ii)* a query originating from another user and $K= N$ (i.e., $N-1$ events). Thus, if the attacker detects such an ASR and has knowledge of the user distribution (assumption *iii* in Section III-A), then he concludes that it originated from U_1 with probability 1/2. Thus, the spatial anonymity of U_1 is breached for all values $K> 2$.

In general, following a similar analysis it can be shown that, if the two quadrants contain a different number of users, the location anonymity is compromised (for all values of K exceeding a threshold) in the quadrant containing the smaller number. Analogous examples can be constructed for *NNC*. Next, we propose an algorithm that avoids this problem.

C. Hilbert Cloak

Hilbert Cloak (*HC*) satisfies *reciprocity*, an important property that is *sufficient* for spatial K-anonymity.

Definition 1 (Reciprocity): Consider a user U issuing a query with anonymity degree K, associated anonymizing set AS, and anonymizing spatial region K-ASR. AS satisfies the reciprocity property if *(i)* it contains U and at least *K-1* additional users and *(ii)* every user in AS also generates the same anonymizing set AS for the given K. The second condition implies that each user in AS lies in the K-ASRs of all other users in AS.

In general, *Interval Cloak*, *Casper* and *NNC* do not satisfy reciprocity as they violate condition *(ii)*. For instance, in the example of Figure 7, although users $U_2 \ldots U_N$ lie in the K-ASR of U_1, U_1 is not in the K-ASR of $U_2 \ldots U_N$ for $2 \leq K < N$. Similarly for *NNC*, although in Figure 6 $U_3 \ldots U_5$ are in the 3-ASR of U_1, U_1 is not in the 3-ASR of $U_3 \ldots U_5$.

Theorem 1: A spatial cloaking algorithm guarantees spatial K-anonymity, if every anonymizing set satisfies the reciprocity property.

Proof: Since every anonymizing set satisfies reciprocity, a K-ASR may have originated from every user in the corresponding anonymizing set AS with equal probability $1/|AS|$, where $|AS|$ is the cardinality of AS. Because $|AS| \geq K$, the probability of identifying the query issuer does not exceed $1/K$. ■

An optimal cloaking algorithm would partition the user population into anonymizing sets that yield minimal ASRs and obey the reciprocity property. However calculating such an optimal partitioning is NP-Hard [21] and would require a fixed K by all queries. *HC* overcomes these problems by utilizing the Hilbert space-filling curve [22] to generate small (but not necessarily optimal) ASRs for variable values of K. The Hilbert space filling curve transforms the 2-D coordinates of each user into a 1-D value $H(U)$. Figure 8 illustrates the Hilbert curves for a 2-D space using a 4×4 and 8×8 space partitioning. With high probability [24], if two points are in close proximity in the 2-D space, they will also be close in the 1-D transformation. A major benefit of Hilbert (and similar) curves, is that they permit the indexing of multidimensional objects through one-dimensional structures (e.g., B-trees).

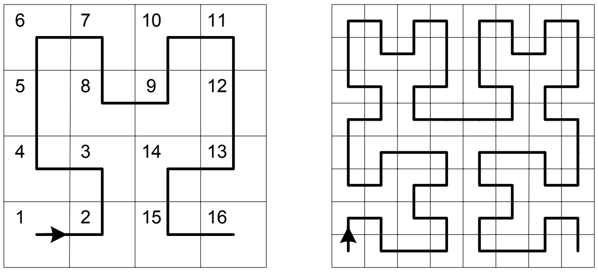


Fig. 8. Hilbert Curve (left: 4×4, right: 8×8)

Given a query from user U with anonymity requirement K, *HC* sorts the Hilbert values and splits them into K-buckets. Each K-bucket has exactly K users, except the last one which may contain up to 2·K-1 users. Let $H(U)$ be the Hilbert value of U and $rank_U$ be the position of $H(U)$ in the sorted sequence of all locations. *HC* identifies the K-bucket containing $rank_U$. The users in that K-bucket constitute the corresponding AS. Figure 9 illustrates an example, where the user ids indicate their Hilbert order. For K=3, the users are grouped into 3 buckets (the last one contains 4 users). When any of U_1, U_2 or U_3 issues a query, *HC* returns the first bucket (shown shaded) as the AS; the MBR (or MBC) of that bucket becomes the 3-ASR.

HC is reciprocal because all users in the same bucket share the same K-ASR; therefore, it guarantees spatial anonymity according to Theorem 1. Furthermore, it can deal with variable values of K by not physically storing the K-buckets. Instead, it maintains a balanced sorting tree, which indexes the Hilbert values. When a user U initiates a query with anonymity degree K, *HC* performs a search for $H(U)$ in the index and computes $rank_U$. From $rank_U$, we calculate the start and end positions defining the K-bucket that includes $H(U)$, as follows:

$$start = rank_U - (rank_U \mod K), \qquad end = start + K - 1$$

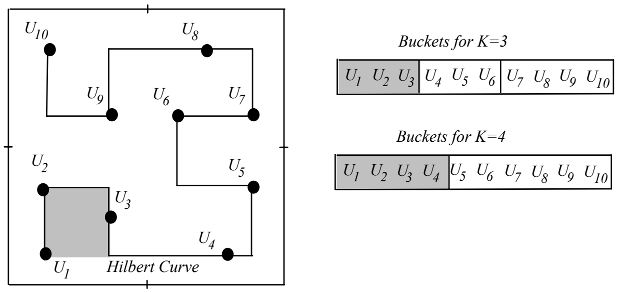

U_{10}
U_8
U_9
U_6
U_7
U_2
U_5
U_3
U_4
U_1
Hilbert Curve
Buckets for K=3
U_1 U_2 U_3 | U_4 U_5 U_6 | U_7 U_8 U_9 U_{10}
Buckets for K=4
U_1 U_2 U_3 U_4 | U_5 U_6 U_7 U_8 U_9 U_{10}

Fig. 9. Example of *Hilbert Cloak*

The complexity of the in-order tree traversal is $O(N)$, where N is the number of indexed users. To compute $rank_U$ efficiently, we use an aggregate tree [28], where each node w stores the number w_{count} of nodes in its left subtree (including itself). Using this data structure, $rank_U$ is calculated in $O(logN)$ as follows: we initialize $rank_U$ to zero and perform a normal lookup for $H(U)$. For every node w we visit, we add w_{count} to $rank_U$ only if we follow a right branch. The complexity of maintaining the aggregate information is $O(logN)$ because changes are propagated from the leaves to the root. Since the complexity of constructing the K-ASR is $O(logN + K)$, whereas search, insert and delete cost $O(logN)$, the data structure is scalable. Therefore, *HC* is applicable to a large number of mobile users who update their location frequently and have varying requirements for the degree of anonymity. Note that, while our description assumes a main memory index, the technique can be easily extended to secondary memory by using B$^+$-trees.

IV. Location-Based Service

The Location-Based Service (LBS) receives the query from the anonymizer, processes it and sends the results back to the anonymizer. In our implementation, the data in the LBS are indexed by an R*-Tree [5]; our methods, however, are independent of the index structure. We support two types of queries:

1) Range queries: The LBS receives the query range which is either an axis-parallel rectangle $\mathcal{R}$ or a circle $\mathcal{C}$. Processing is straight-forward; the R-tree is traversed from the root to the leaves and any object inside $\mathcal{R}$ (or $\mathcal{C}$) is returned.
2) kNN queries: This case is more complex, since the LBS must find the k nearest neighbors of the entire range. For rectangular ranges, we adopt the Range Nearest Neighbor ($\mathcal{R}k$NN) algorithm [17] (see Section II-C for details). The rest of this section describes our $\mathcal{C}k$NN algorithm, which computes the kNNs of circular ranges.

A. $\mathcal{C}k$NN - Circular Range kNN

Similar to rectangular ranges [17], the set of kNNs of a circular range $\mathcal{C}$ also consists of two subsets of objects: *(i)* all the objects inside $\mathcal{C}$ and *(ii)* the kNNs of the circumference of $\mathcal{C}$. The objects in *(i)* are retrieved by a range query; in the rest of the section, we present the novel $\mathcal{C}k$NN-Circ algorithm which computes the kNNs of the circumference of $\mathcal{C}$. Intuitively $\mathcal{C}k$NN-Circ is similar to CNN (see Section II-C). However, some of the properties of 1-D shapes which are used in CNN (e.g., continuity by the definition of [29]) do not hold for 2-D shapes, rendering the problem more complex.

Conceptually, $\mathcal{C}k$NN-Circ partitions the circumference of $\mathcal{C}$ into disjoint arcs, and associates to each arc the data objects nearest to it. Consider the example of Figure 10, where p_1, p_2 and p_3 are the data objects. Let s_0, s_1 be the intersection points of the *perpendicular bisector* of p_1p_2 (denoted by $\perp p_1p_2$) with $\mathcal{C}$, i.e., $|p_1s_0| = |s_0p_2|$ and $|p_1s_1| = |s_1p_2|$. Assuming that the center c of $\mathcal{C}$ is the origin of the coordinate system, the *polar* coordinates of s_0 are $(r, \hat{s_0})$, where r is the radius of $\mathcal{C}$ and $\hat{s_0}$ is the (anti-clockwise) angle between the x-axis and the vector $c\vec{s}_0$. Similarly, the polar coordinates of s_1 are $(r, \hat{s_1})$. The NN of every point in the arc $[\hat{s_0}, \hat{s_1}]$ is p_1; we denote this as: $[\hat{s_0}, \hat{s_1}] \rightarrow p_1$. Likewise $[\hat{s_1}, \hat{s_0}] \rightarrow p_2$, since any point in the arc $[\hat{s_1}, \hat{s_0}]$ is closer to p_2 than to any other object. Therefore, the set of NNs of $\mathcal{C}$ is $\{p_1, p_2\}$. Note that p_3 is not in this set, even though it is closer to $\mathcal{C}$ than p_2, because p_1 is closer than p_3 to any point on $\mathcal{C}$; we say that p_1 *covers* p_3.

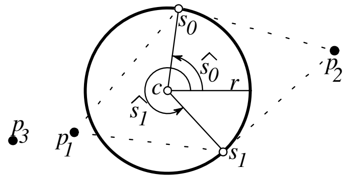


Fig. 10. The 1-NNs of $\mathcal{C}$ are p_1 and p_2

Let $\mathcal{D} = \{p_1, p_2, \ldots, p_n\}$ be the set of all data objects. $\mathcal{C}k$NN-Circ maintains a list SL of mappings $[a, b] \rightarrow p_i$, where a, b are angles defining an arc on $\mathcal{C}$, $0 \leq a < b \leq 2\pi$, and $p_i \in \mathcal{D}$ is the object which is closest to every point of arc $[a, b]$ than any other object $p_j \in \mathcal{D}$. The $\mathcal{C}k$NN-Circ pseudocode is shown in Figure 13.

In the example of Figure 11a, let $p_1 \in \mathcal{D}$ be the first object encountered by the algorithm. Since SL is initially empty, p_1 is closest to the entire $\mathcal{C}$. Without loss of generality, we pick two points $s_0, s_0' \in \mathcal{C}$, where $\hat{s_0} = 0$ and $\hat{s_0'} = 2\pi$ (i.e., they are the same point), and insert the mapping $[\hat{s_0}, \hat{s_0'}] \rightarrow p_1$ into SL (line 2 of the pseudocode). For each subsequent point $p \in \mathcal{D}$, the algorithm traverses SL (line 4) and examines all existing mappings $[a, b] \rightarrow q$. There are three possible cases:

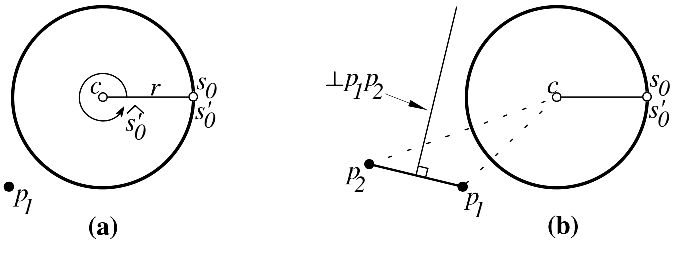


Fig. 11. $\mathcal{C}k$NN example: The perpendicular bisector does not intersect $\mathcal{C}$

Case 1: $\perp pq \cap \mathcal{C} = \varnothing$ or $\perp pq$ is tangent to $\mathcal{C}$ (lines 5-6). This case is exemplified[5] in Figure 11b. The only existing mapping is $[\hat{s_0}, \hat{s_0'}] \rightarrow p_1$, and p_2 is processed next. Any point on the right-hand side of $\perp p_1p_2$, is closer to p_1. Therefore, the entire $\mathcal{C}$ is closer to p_1 than to p_2. Since the mapping to p_1 already exists, there is no change in SL. Furthermore, even if there were more

[5] For simplicity, all objects are shown outside $\mathcal{C}$. However, the algorithm also works for objects inside $\mathcal{C}$.

mappings inside SL, it would not be necessary to compare with p_2, since p_1 covers p_2. On the other hand, if p_2 was at the right-hand side (and p_1 on the left), then p_2 would be closer to $\mathcal{C}$ than p_1. In this case, the algorithm would remove the $[\hat{s_0}, \hat{s'_0}] \rightarrow p_1$ mapping from SL and add a new one $[\hat{s_0}, \hat{s'_0}] \rightarrow p_2$ (line 6).

Case 2: $\perp pq \cap \mathcal{C} = \{s_0, s_1\}$ and either $\hat{s_0} \in [a,b]$ or $\hat{s_1} \in [a,b]$ (lines 12-14). This case is illustrated in Figure 12a: both p_1 and p_2 have already been processed, and there are two mappings in SL: $[\hat{s_1}, \hat{s'_1}] \rightarrow p_1$ and $[\hat{s'_1}, \hat{s_1}] \rightarrow p_2$. Let p_3 be the next object to be processed. p_3 is compared against the existing mappings. For the first one (i.e., $[\hat{s_1}, \hat{s'_1}] \rightarrow p_1$), $\perp p_1 p_3$ intersects $\mathcal{C}$ at s_2 and s'_2. Note that $\hat{s'_2} \notin [\hat{s_1}, \hat{s'_1}]$, so it is not considered further. On the other hand, $\hat{s_2} \in [\hat{s_1}, \hat{s'_1}]$ and p_3 is closer to s_1 than p_1. Therefore (line 13), the arc is split into two parts $[\hat{s_1}, \hat{s_2}]$ and $[\hat{s_2}, \hat{s'_1}]$, which are assigned to p_3 and p_1, respectively. Similarly, for the second mapping (i.e., $[\hat{s'_1}, \hat{s_1}] \rightarrow p_2$), $\perp p_2 p_3$ intersects $\mathcal{C}$ at s_3, s'_3. Only $\hat{s_3} \in [\hat{s'_1}, \hat{s_1}]$, so the arc is split into $[\hat{s'_1}, \hat{s_3}]$ and $[\hat{s_3}, \hat{s_1}]$, which are assigned to p_2 and p_3, respectively. After updating, $SL = \{[\hat{s_2}, \hat{s'_1}] \rightarrow p_1, [\hat{s'_1}, \hat{s_3}] \rightarrow p_2, [\hat{s_3}, \hat{s_1}] \rightarrow p_3, [\hat{s_1}, \hat{s_2}] \rightarrow p_3\}$. The last two mappings can be combined (i.e., $[\hat{s_3}, \hat{s_2}] \rightarrow p_3$) since they are consecutive and are mapped to the same object.

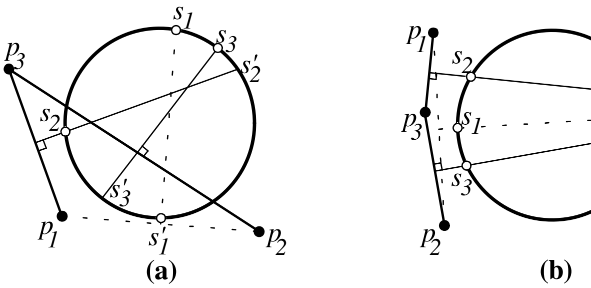


Fig. 12. The perpendicular bisector intersects $\mathcal{C}$

Case 3: $\perp pq \cap \mathcal{C} = \{s_0, s_1\}$ and both $\hat{s_0}, \hat{s_1} \in [a,b]$ (lines 9-11). This case is illustrated in Figure 12b: again, both p_1 and p_2 have already been processed, and $SL = \{[\hat{s'_1}, \hat{s_1}] \rightarrow p_1, [\hat{s_1}, \hat{s'_1}] \rightarrow p_2\}$. Next, p_3 is compared to the first mapping of SL. Note that $\perp p_1 p_3$ intersects $\mathcal{C}$ at s'_2, s_2 and both $\hat{s'_2}, \hat{s_2} \in [\hat{s'_1}, \hat{s_1}]$. Therefore (line 10), the arc is split into three parts and since p_3 is closer to s'_1 than p_1 the corresponding mappings are: $[\hat{s'_1}, \hat{s'_2}] \rightarrow p_3, [\hat{s'_2}, \hat{s_2}] \rightarrow p_1, [\hat{s_2}, \hat{s_1}] \rightarrow p_3$. Similarly, after considering $\perp p_2 p_3$, $[\hat{s_1}, \hat{s'_1}]$ is also split into three parts. Finally, after combining the consecutive mappings, $SL = \{[\hat{s'_2}, \hat{s_2}] \rightarrow p_1, [\hat{s_2}, \hat{s_3}] \rightarrow p_3, [\hat{s_3}, \hat{s'_3}] \rightarrow p_2, [\hat{s'_3}, \hat{s'_2}] \rightarrow p_3\}$.

For simplicity, the pseudocode of Figure 13 computes only the 1-NNs. To compute the kNNs, instead of a single object, the arcs in our implementation are mapped to an ordered list of k objects: $[a,b] \rightarrow (p_1, \ldots, p_k)$, where p_1 is the nearest neighbor of arc $[a,b]$, p_2 is the second NN of arc $[a,b]$, etc. The procedure is called for each position i $(1 \leq i \leq k)$ of the ordered list. In the ith call, if an object $p \in \mathcal{D}$ already exists in position j $(1 \leq j \leq i-1)$, then p is not considered for that mapping. Also, if an arc is split, the objects in positions $1 \ldots i-1$ (i.e. the $i-1$ nearest neighbors found already) are not altered. The worst case complexity of $\mathcal{C}k$NN is $O(|\mathcal{D}|^k)$, since any object may cause an arc split. In practice, however, the algorithm is faster, because the objects which are far away from $\mathcal{C}$ do not cause splits.

```
CkNN-Circ(D: the set of objects)
1.  for every object $p \in \mathcal{D}$ do
2.    if $SL = \varnothing$ then $SL := \{[0, 2\pi] \rightarrow p\}$
3.    else
4.      for every interval $\varphi \equiv [a,b] \rightarrow q, \varphi \in SL$ do
5.        if $\perp pq \cap \mathcal{C} = \varnothing$ or $\perp pq$ is tangent to $\mathcal{C}$ then
6.          if $|p\mathcal{C}| < |q\mathcal{C}|$ then $SL := (SL - \varphi) \cup \{[a,b] \rightarrow p\}$
            else break
7.        else
8.          let $s_0, s_1$ be two points such that $\perp pq \cap \mathcal{C} = \{s_0, s_1\}$
9.          if $\hat{s_0} \in [a,b]$ and $\hat{s_1} \in [a,b]$ then
            // Assume $\hat{s_0} < \hat{s_1}$ (the other case is symmetric)
10.           if $|p\mathcal{C}_a| < |q\mathcal{C}_a|$ then $SL := (SL - \varphi) \cup$
                  $\cup \{[a, \hat{s_0}] \rightarrow p, [\hat{s_0}, \hat{s_1}] \rightarrow q, [\hat{s_1}, b] \rightarrow p\}$
                  // $\mathcal{C}_a$, $\mathcal{C}_b$ are the endpoints of arc $[a,b]$
11.           else $SL := (SL - \varphi) \cup$
                  $\cup \{[a, \hat{s_0}] \rightarrow q, [\hat{s_0}, \hat{s_1}] \rightarrow p, [\hat{s_1}, b] \rightarrow q\}$
12.         else if $\hat{s_0} \in [a,b]$ or $\hat{s_1} \in [a,b]$ then
              // Let only $\hat{s_0} \in [a,b]$ ($\hat{s_1} \in [a,b]$ is symmetric)
13.           if $|p\mathcal{C}_a| < |q\mathcal{C}_a|$ then $SL := (SL - \varphi) \cup$
                  $\cup \{[a, \hat{s_0}] \rightarrow p, [\hat{s_0}, b] \rightarrow q\}$
14.           else $SL := (SL - \varphi) \cup \{[a, \hat{s_0}] \rightarrow q, [\hat{s_0}, b] \rightarrow p\}$
15.         else if $|p\mathcal{C}_a| < |q\mathcal{C}_a|$ then
              $SL := (SL - \varphi) \cup \{[a,b] \rightarrow p\}$
16. return $SL$

CkNN(D: the set of objects)
1.  call CkNN-Circ(D)
2.  return $\{p : p \in \mathcal{D} \wedge p$ is inside $\mathcal{C}\} \cup$
        $\cup \{p : p$ belongs to a mapping of $SL\}$
```

Fig. 13. Find the 1-NNs of a circular range $\mathcal{C}$

B. R-trees and $\mathcal{C}k$NN

In order to use the $\mathcal{C}k$NN algorithm with an R-tree, we employ a branch-and-bound heuristic. Starting from the root, the R-tree is traversed either in Depth-First or in Best-First [29] manner. When a leaf entry (i.e., object) p is encountered, the $\mathcal{C}k$NN algorithm is used to check whether p is closer to $\mathcal{C}$ than any of the objects in the current mappings (i.e., p is a *qualifying object*) and updates SL accordingly. For an intermediate entry E we avoid visiting its subtree if it is impossible to contain any qualifying object.

Figure 14 presents an example where p_1 and p_2 are the current 1-NNs of $\mathcal{C}$. Next, an entry E from an intermediate node of the R-tree is encountered. We observe the following:

Lemma 1: Let MBR_E be an axis-parallel MBR and let *st* be the side which is closest to circle $\mathcal{C}$. If *st* does not contain any of the kNNs of $\mathcal{C}$, then MBR_E cannot contain any kNN.

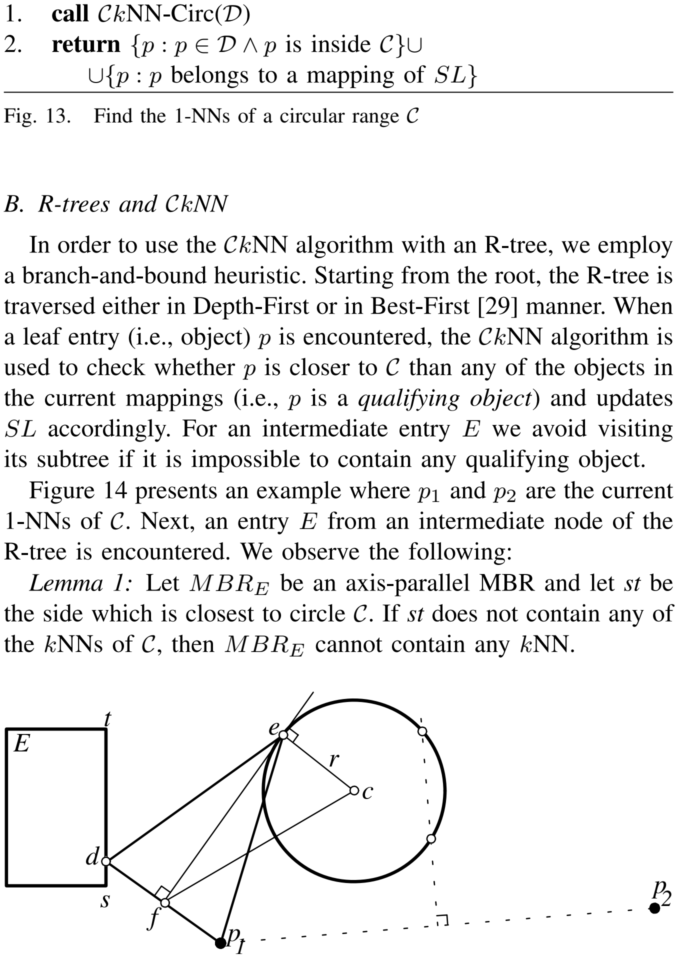


Fig. 14. Check if E may contain qualifying objects

The proof is straight-forward, since any point in the MBR will be further away from $\mathcal{C}$ than the closest point on st. In our example, the right side st of E is closer to $\mathcal{C}$. Assume there is a point d on st, such that the perpendicular bisector $\perp dp_1$ is tangent to $\mathcal{C}$, and let $e \equiv \perp dp_1 \cap \mathcal{C}$. Then we get the following system of equations[6]:

$$\begin{cases} |ce| = r \\ |p_1e| = |de| \\ |p_1e|^2 - |p_1f|^2 = |cf|^2 - r^2 \end{cases} \tag{1}$$

The first equation is derived from the fact that $e \in \mathcal{C}$, while the second one is because the distance from any point on $\perp dp_1$ to d and p_1 is equal. The third equation results from the application of the Pythagorean theorem on the orthogonal triangles p_1fe and fec which have a common side ef. After substituting the points with their Cartesian coordinates, we get the following system (note that $x_f = \frac{x_d+x_{p1}}{2}, y_f = \frac{y_d+y_{p1}}{2}$, since f is the middle of dp_1):

$$\begin{cases} (x_e - x_c)^2 + (y_e - y_c)^2 = r^2 \\ (x_d - x_e)^2 + (y_d - y_e)^2 = (x_{p1} - x_e)^2 + (y_{p1} - y_e)^2 \\ (x_{p1} - x_e)^2 + (y_{p1} - y_e)^2 - \frac{(x_d - x_{p1})^2 + (y_d - y_{p1})^2}{4} = \\ = \left(\frac{x_d+x_{p1}}{2} - x_c\right)^2 + \left(\frac{y_d+y_{p1}}{2} - y_c\right)^2 - r^2 \end{cases}$$

There are three equations and three unknowns: x_e, y_e, y_d. If there is a real solution to this system, under the condition $(x_d, y_d) \in st$, then there *may* be a qualifying object inside the subtree of E. Else all objects in E are further away from $\mathcal{C}$ than the current objects in SL, so the subtree under E can be pruned.

Solving this system, however, is slow (in the order of 100's of msec in an average computer); given that an entry E must be checked against many objects, the running time is prohibitively long. Therefore, in our implementation, we use the $\mathcal{R}k$NN algorithm to traverse the R-tree and employ the $\mathcal{C}k$NN algorithm only for the objects at the leaf-level. Our strategy is based on the following observation:

Lemma 2: Let $\mathcal{C}$ be a circle, MER the maximum enclosed axis-parallel rectangle of $\mathcal{C}$ and S the set of kNNs of MER's perimeter. Let p_i be an object, such that p_i is inside MER and $p_i \notin S$. Then p_i cannot be a kNN for any point of $\mathcal{C}$.

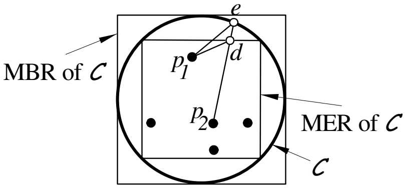


Fig. 15. The MBR and the MER of $\mathcal{C}$

Proof: Assume the lemma does not hold. Figure 15 shows an example where p_2 is inside MER and $p_2 \notin S$. Assume that p_2 is the NN of point $e \in \mathcal{C}$. Let d be the point where the line segment p_2e intersects the perimeter of MER, and p_1 be the object which is the NN of d. It follows from our hypothesis that: $|p_2e| < |p_1e|$. Using the triangular inequality, we get: $|p_2d| + |de| < |p_1d| + |de| \Rightarrow |p_2d| < |p_1d|$ which is a contradiction, since p_1 is the NN of d. Therefore, the lemma holds. ■

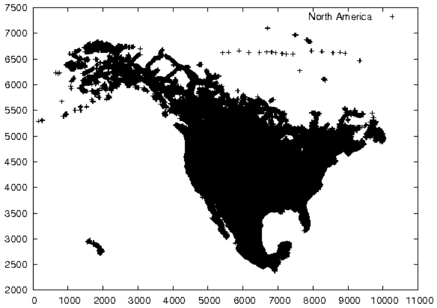


Fig. 16. North-America (NA) dataset

We construct the Minimum Bounding Rectangle[7] MBR and the Maximum Enclosed Rectangle MER of $\mathcal{C}$ (the side-length of MER is $\sqrt{2}r$). Conceptually, our implementation works in three steps:

1) Use the $\mathcal{R}k$NN algorithm to find the set S_1 of kNNs of MBR (including all the objects inside MBR). Recall that S_1 is a superset of the kNNs of any point inside MBR; therefore, it contains all the kNNs of $\mathcal{C}$.
2) Use CNN (see Section II-C) to find the set S_2 of kNNs of *only* the perimeter of MER. Use Lemma 2 and S_2 to prune objects from S_1.
3) Call the $\mathcal{C}k$NN algorithm with the objects remaining in S_1.

In practice, these steps can be combined. In a single traversal of the R-tree, steps (1) and (2) can be used at the intermediate levels to prune the tree and step (3) is applied on the leaf-level objects.

V. Experimental Evaluation

This section evaluates the proposed anonymization and query processing algorithms. We implemented prototypes for both the anonymizer and the LBS using C++. All experiments were executed on an Intel Xeon 2.8GHz machine with 2.5GB of RAM and Linux OS. Our workload for user positions and landmarks/points of interest consists of the NA dataset [30], which contains 569K locations on the North-American continent (Figure 16). Performance is measured in terms of CPU time, I/O time and communication cost. At the anonymizer we employed main-memory structures, therefore we measured only the CPU time. At the LBS, we used an R*-Tree and measured the total time (i.e., I/O and CPU time); in all experiments we maintained a cache with size equal to 10% of the corresponding R*-Tree. The communication cost was measured in terms of number of candidates sent from the LBS back to the anonymizer.

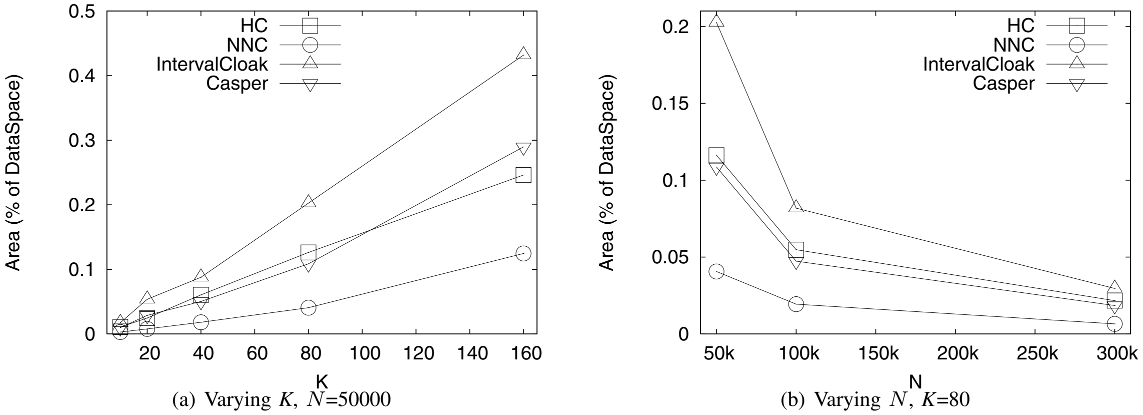
In the following, Section V-A focuses on cloaking algorithms at the anonymizer, whereas Section V-B evaluates query processing at the LBS.

A. Anonymizer Evaluation

We compare the proposed *Nearest Neighbor Cloak* (*NNC*) and *Hilbert Cloak* (*HC*) against *Casper* and *Interval Cloak* (*IC*). The first experiment measures the area of rectangular K-ASRs. Recall that we wish to minimize the ASR area, since it affects the processing time at the LBS and the communication cost between

[6] If a different side of E is closer to $\mathcal{C}$, the equations are modified accordingly.

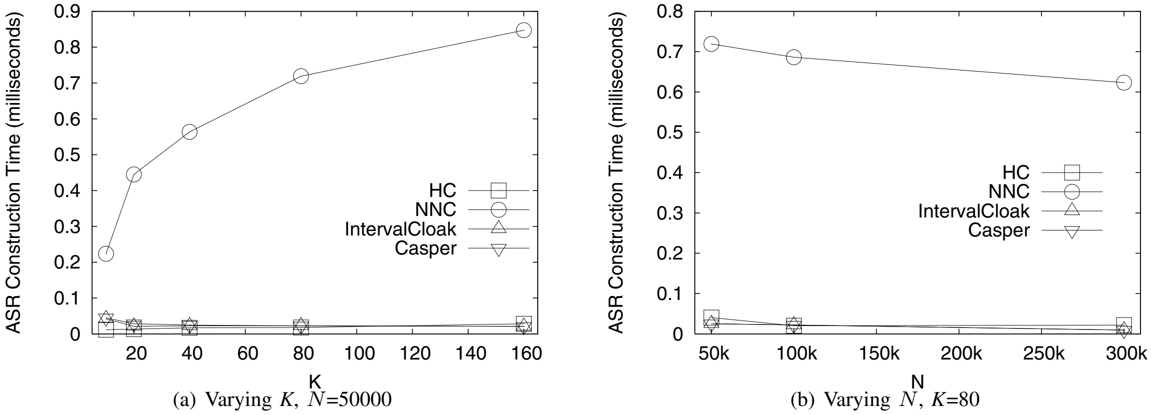
[7] For a set of users $U_{1\ldots n}$, the MBR of $\mathcal{C}$ is not the same as their corresponding anonymizing rectangle $\mathcal{R}$.



(a) Varying K, N=50000

(b) Varying N, K=80

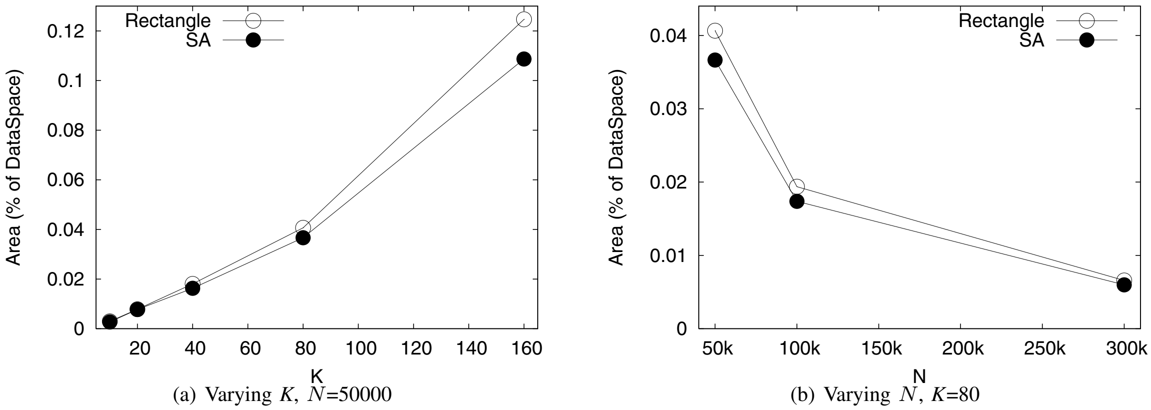
Fig. 17. Area of rectangular K-ASR



(a) Varying K, N=50000

(b) Varying N, K=80

Fig. 18. K-ASR generation time



(a) Varying K, N=50000

(b) Varying N, K=80

Fig. 19. Rectangular vs Smallest Area K-ASR, *Nearest Neighbor Cloak*

the LBS and the anonymizer. First, we fix the number of users $N = 50000$ and vary the degree of anonymity K. The K-ASR area is expressed as a percentage of the entire data space. We generated 1000 queries originating at random users. Figure 17a shows the average area per query. Clearly *IC* is the worst algorithm, whereas *NNC* is the best. *HC* and *Casper* exhibit similar behavior to each other. All algorithms scale linearly with K in terms of ASR area. Figure 17b, shows the K-ASR area for K = 80 and varying N. Since the extent of the data space remains constant, an increase in user population translates to higher user density, hence reduced K-ASR size for all methods. The relative performance among the algorithms remains the same. Observe that *HC* and *Casper* outperform *IC*, and generate ASRs with roughly twice the area of *NNC*.

Figure 18 shows the average ASR generation time (in milliseconds) for varying K and N. *HC*, *IC* and *Casper* behave similarly. *NNC*, on the other hand, has a significantly larger generation time, due to the more costly nearest-neighbor search. Nevertheless, we will show in the following that *NNC* is best in terms of overhead at the LBS.

So far, we focused on rectangular K-ASRs. However, depending on the user distribution, circular K-ASRs may have smaller size. Here we adopt a simple optimization: first we identify the set of users which belong to a K-ASR. Then we calculate the minimum bounding rectangle $\mathcal{R}$ and the minimum enclosing circle $\mathcal{C}$ of the K-ASR, and select the shape with the *smallest area*.

We call this method SA. *NNC* is more suitable to be combined with SA, since the nearest neighbor search tends to identify circular clusters of users. Figures 19a and 19b compare the rectangle-only approach against the SA optimization for varying K and N, respectively. SA manages to reduce the K-ASR area by up to 15%.

Finally, we measure the anonymity strength of the above-mentioned algorithms against the *center-of-ASR* attack[8]. We consider a workload of 1000 queries, originating at a set of random users, with K = 50. Figure 20 shows the probability $P[index_U = i]$ (the experiment is similar to that of Section III-A). Recall that $index_U = 1$ means that user U is the closest to the center of the K-ASR. Furthermore, the dashed line corresponds to the distribution of $index_U$ for the ideal anonymization technique. All studied algorithms preserve privacy in the case of the *center-of-ASR* attack. *NNC* is close to the ideal distribution and there are few cases where the K-ASR encloses more than K users, which explains the relatively small ASR size observed in the previous experiments. *HC* and *Casper* exhibit similar behavior to each other, but include a larger number of redundant users inside the K-ASR, compared to *NNC*; this is why $P[index_U = i] > 0$ for $i > K$. However, they are both better than *IC*.

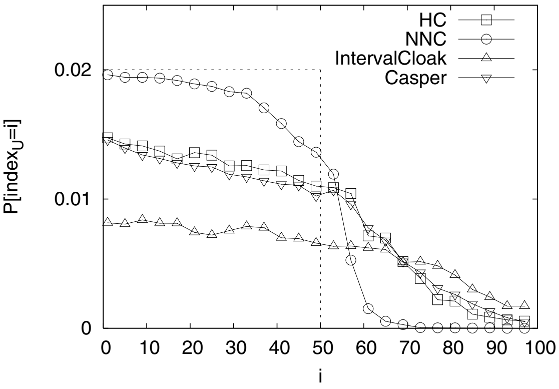


Fig. 20. *center-of-ASR* attack, K= 50

B. Location-Based Service Evaluation

For this experiment, we generate 1000 queries originating at random users. The corresponding K-ASRs are sent to the LBS and the queries are executed against the entire NA dataset, which is indexed by an R*-tree. For all K-ASR generation techniques, we compare the average processing time (i.e, CPU plus I/O time) per query, and the size of the candidate set. The latter is a superset of the actual result, and it reflects the communication cost between the LBS and the anonymizer. First, we focus on kNN queries. Figure 21 shows the performance for varying number of nearest neighbors k. *NNC* generates a significantly lower number of candidates compared to the other techniques. This is expected, since the sizes of the corresponding K-ASRs are also smaller. *HC* and *Casper* generate up to 50% more candidates than *NNC*. However, they both outperform *IC* by a large margin. In terms of processing time, *NNC* is the fastest, with *HC* and *Casper* considerably better than *IC*.

In Figure 22 we fix the number of neighbors $k = 2$ and vary the degree of anonymity K. Again, *NNC* performs best, followed by *HC* and *Casper*. The difference is more significant for larger K values, as the average size of the K-ASR increases. Figure 23 shows the number of candidates and processing time for varying N. Note that more users lead to higher density, thus smaller K-ASRs. Consequently, the number of candidates and the average processing time decrease with N.

We also evaluated the performance of the four techniques for range queries. The results are presented in Figure 24 for varying K and $N = 50000$. Again, we observe a significant advantage of *NNC* over the other techniques, while *HC* and *Casper* outperform *IC* in terms of both processing cost and candidate set size. The trends for varying N are similar.

The previous results were obtained for rectangular K-ASRs. We also investigated the effect of the SA (i.e., smallest area) optimization on query processing. For a given K-ASR, if SA generates a circular range $\mathcal{C}$, we employ $\mathcal{C}k$NN to execute the corresponding kNN query. For our workload, SA generated circular ranges for around 45% of the K-ASRs when K was small, and up to 90% for large values of K. Figure 25 compares SA against the rectangles-only approach for $k = 2$ neighbors and varying K. SA reduces the number of candidates by up to 18%, compared to the rectangular K-ASR. The tradeoff is the increased processing time. The same relative performance is observed in Figure 26, where we vary N.

C. Discussion

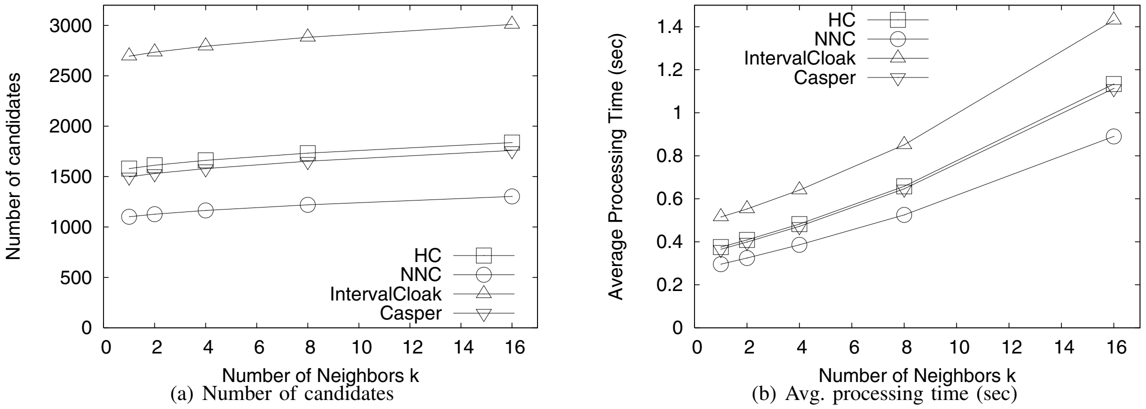
The experimental evaluation verifies the superiority of *Hilbert Cloak* and *Nearest Neighbor Cloak*, compared to the existing approaches. Our *HC* algorithm provides privacy guarantees under all user and query distributions, and its overhead in terms of ASR generation time, query processing time and communication cost is similar to *Casper*, the most recent and most efficient technique. On the other hand, *NNC* clearly outperforms *Casper* in terms of overhead at the LBS, while offering similar anonymity strength.

The LBS is likely to maintain huge volumes of data and disk-based data structures, while the anonymizer typically uses memory-based data structures. For this reason, the query overhead at the LBS is considerably larger than at the anonymizer (observe that time is measured in milliseconds in Figure 18 instead of seconds in Figure 21b). Under these circumstances, the reduced LBS processing cost offers *NNC* an important performance advantage, despite its increased K-ASR generation time.

The choice between *Hilbert Cloak* and *Nearest Neighbor Cloak* involves a clear trade-off between privacy guarantees on one hand, and processing overhead on the other. If provable anonymity guarantees are required, *Hilbert Cloak* is the only option. Nevertheless, *Nearest Neighbor Cloak* also achieves strong anonymity for most of the cases, and may be acceptable for applications where outliers do not constitute an anonymity threat (e.g., very frequent user movement) and efficiency is crucial.

Finally, there is a tradeoff between rectangular-only K-ASRs and the SA optimization. The cost of $\mathcal{C}k$NN at the LBS is higher than $\mathcal{R}k$NN. However, $\mathcal{C}k$NN reduces the number of candidates. Therefore, $\mathcal{C}k$NN is preferable if the communication cost is more important than the processing cost at the LBS. In practice, this happens if a single anonymizer sends queries to several LBSs. In this case the bandwidth of the single anonymizer is shared among all connections. Thus, it is important to minimize the communication cost, whereas the processing cost is distributed among the LBSs.

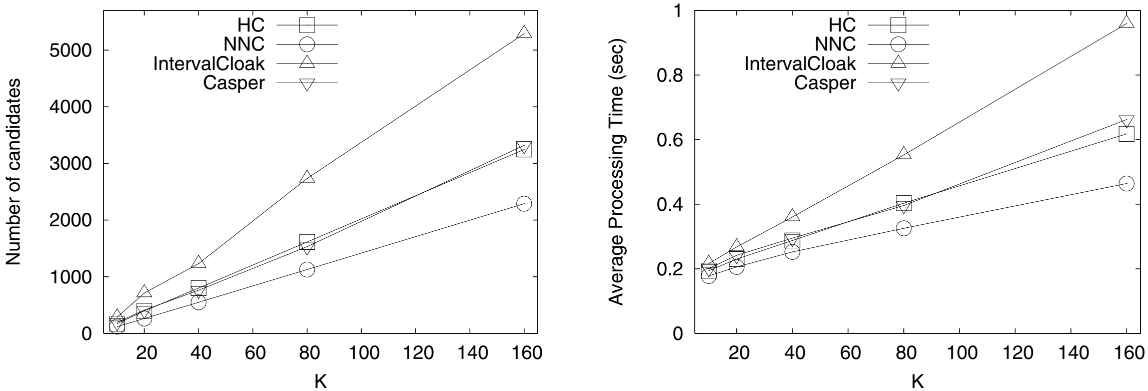
[8] Although we formally proved that *Hilbert Cloak* guarantees location anonymity, we include this experiment for illustration purposes.



(a) Number of candidates

(b) Avg. processing time (sec)

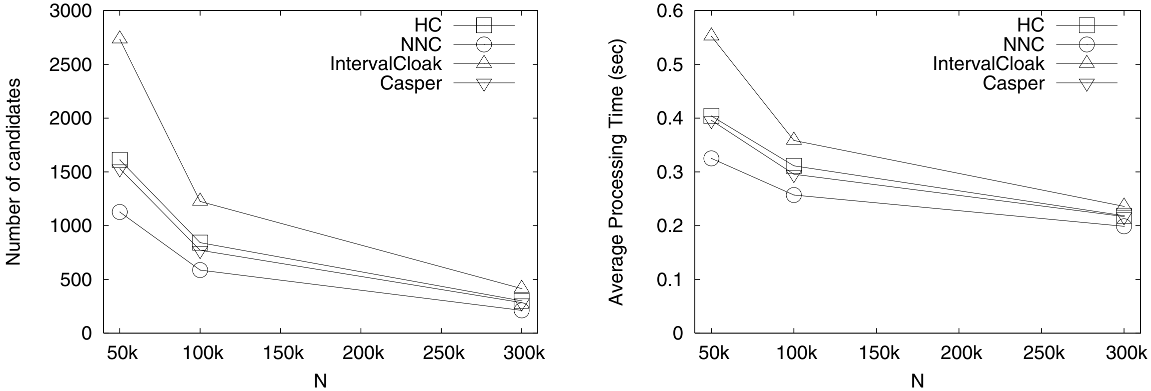
Fig. 21. *k*NN queries, varying number of neighbors, $N = 50000$, K = 80



(a) Number of candidates

(b) Avg. processing time (sec)

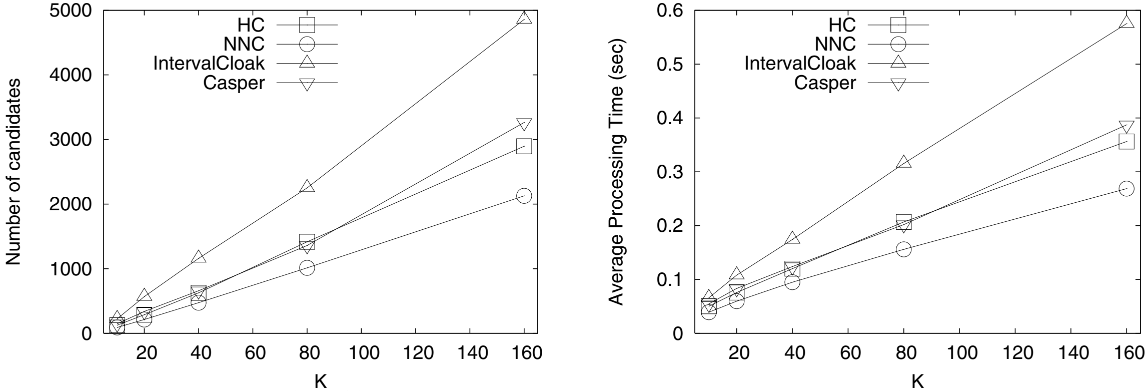
Fig. 22. *k*NN queries, varying K, $k = 2$ neighbors, $N = 50000$



(a) Number of candidates

(b) Avg. processing time (sec)

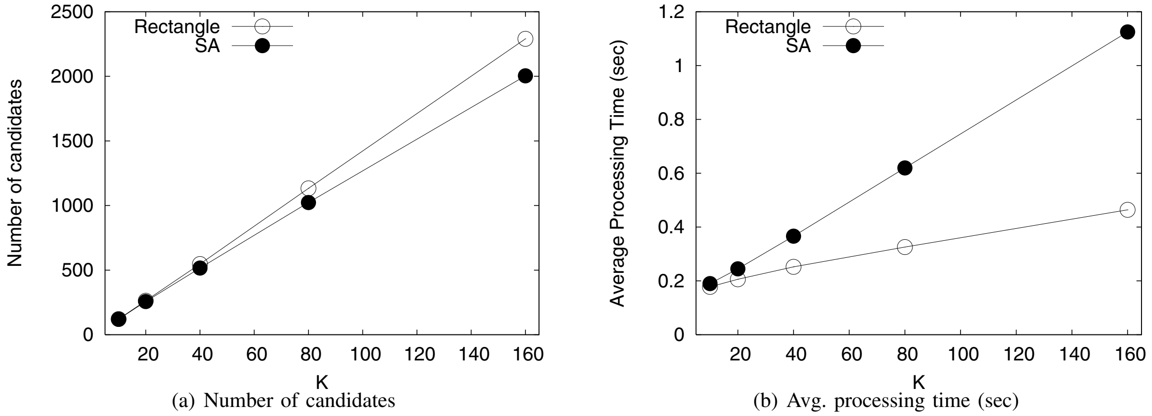
Fig. 23. *k*NN queries, varying N, $k = 2$ neighbors, K = 80



(a) Number of candidates

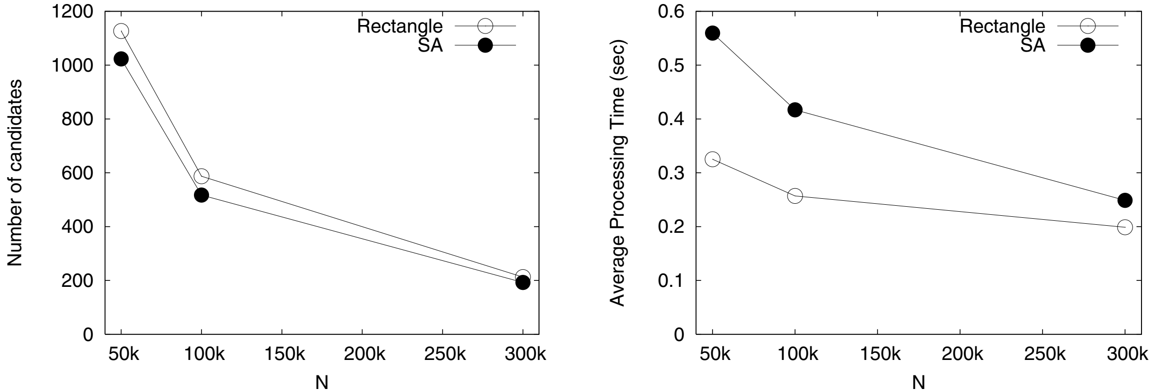
(b) Avg. processing time (sec)

Fig. 24. Range queries, $N = 50000$, varying K



(a) Number of candidates (b) Avg. processing time (sec)

Fig. 25. *NNC*, rectangular vs SA K-ASR, $k = 2$ neighbors, $N = 50000$, varying K



(a) Number of candidates (b) Avg. processing time (sec)

Fig. 26. *NNC*, rectangular vs SA K-ASR, $k = 2$ neighbors, varying N, K = 80

VI. Conclusions and Future Work

In this paper we studied the preservation of query anonymity in Location Based Services. The main idea is to conceal the user coordinates, by replacing them with a spatial region (either a circle or a rectangle). This region covers the query initiator and at least $K-1$ other users. We proposed methods that construct appropriate anonymizing regions and investigated their tradeoffs. We also designed algorithms that run at untrustworthy LBSs, and compute exact answers to anonymized range and nearest neighbor queries. Our work is the first to provide a formal guarantee for the anonymization strength. Moreover, the experimental evaluation showed that our methods outperform the existing state-of-the-art.

Our initial findings reveal interesting directions for future research. A challenging problem is to ensure anonymity for users issuing continuous spatial queries. Intuitively, preserving anonymity is more difficult in this case: asking the same query from successive locations may disclose the identity of the querying user, who will be included in all ASRs. Our framework can be extended for processing continuous queries as follows: a snapshot technique (e.g., *NNC*, *HC*) is first employed to determine the set AS of users included in the ASR for the initial snapshot of the query; this anonymizing set is "frozen" for the rest of the query lifetime. The MBR of AS is then used as ASR at subsequent snapshots. However, as users move in different directions, such an approach may yield large ASRs. Another possibility would be to employ an entirely different framework based on Private Information Retrieval (PIR) [9]. Existing PIR methods, however, are impractical due to huge network cost. Continuous queries involve several complex issues, and constitute a promising topic for further work.

Another interesting aspect is preventing "background knowledge" attacks, when the attacker has additional information about the preferences of certain users. For instance, if Bob, a rugby fan, asks for the location of the closest rugby club, and the associated ASR contains only female users in addition to Bob, the attacker may infer Bob as query source with higher probability. A solution to this problem would be to group users into partitions according to their areas of interest (e.g., users who query frequently about restaurants, or night clubs, etc). Then, when a query is issued, the corresponding ASR is generated with users from the same interest group as the query source, such that each user in the ASR has an equally likely probability of having asked the query.

Finally, it would be interesting to investigate methods that do not require an anonymizer. Assuming that the users trust each other, the query initiators could collaborate with peers in their vicinity to compute their anonymizing region.

Acknowledgment

This work was partially supported by grant HKUST 6184/06E from Hong Kong RGC.

References

[1] N. R. Adam and J. C. Wortmann, "Security-Control Methods for Statistical Databases: A Comparative Study," *ACM Computing Surveys*, vol. 21, no. 4, pp. 515–556, 1989.

[2] C. C. Aggarwal, "On k-Anonymity and the Curse of Dimensionality." in *Proc. of VLDB*, 2005, pp. 901–909.

[3] R. Agrawal and R. Srikant, "Privacy-Preserving Data Mining," in *Proc. of SIGMOD*, 2000, pp. 439–450.

[4] R. Bayardo and R. Agrawal, "Data Privacy through Optimal k-Anonymization." in *Proc. of ICDE*, 2005, pp. 217–228.

[5] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger, "The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles." in *Proc. of SIGMOD*, 1990, pp. 322–331.

[6] A. R. Beresford and F. Stajano, "Location privacy in pervasive computing," *IEEE Pervasive Computing*, vol. 2, no. 1, pp. 46–55, 2003.

[7] C. Bettini, X. S. Wang, and S. Jajodia, "Protecting Privacy Against Location-Based Personal Identification," in *Proc. of VLDB Workshop on Secure Data Management (SDM)*, 2005, pp. 185–199.

[8] R. Cheng, Y. Zhang, E. Bertino, and S. Prabhakar, "Preserving User Location Privacy in Mobile Data Management Infrastructures." in *Proc. of Privacy Enhancing Technologies*, 2006, pp. 393–412.

[9] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan, "Private information retrieval," *Journal of the ACM*, vol. 45, no. 6, pp. 965–981, 1998.

[10] C.-Y. Chow, M. F. Mokbel, and X. Liu, "A Peer-to-Peer Spatial Cloaking Algorithm for Anonymous Location-based Services," in *Proc. of ACM-GIS*, 2006, pp. 171–178.

[11] B. Gedik and L. Liu, "Location Privacy in Mobile Systems: A Personalized Anonymization Model." in *Proc. of ICDCS*, 2005, pp. 620–629.

[12] G. Ghinita, P. Kalnis, and S. Skiadopoulos, "PRIVE: Anonymous Location-based Queries in Distributed Mobile Systems." in *Proc. of World Wide Web Conf. (WWW)*, 2007, pp. 371–380.

[13] G. Ghinita, P. Kalnis, and S. Skiadopoulos, "MOBIHIDE: A Mobile Peer-to-Peer System for Anonymous Location-Based Queries." in *Proc. of the Int. Symposium in Spatial and Temporal Databases (SSTD)*, 2007, pp. 221–238.

[14] G. Ghinita, P. Karras, P. Kalnis, and N. Mamoulis, "Fast Data Anonymization with Low Information Loss." in *Proc. of VLDB*, 2007.

[15] M. Gruteser and D. Grunwald, "Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking." in *Proc. of USENIX MobiSys*, 2003, pp. 31–42.

[16] B. Hoh and M. Gruteser, "Protecting Location Privacy Through Path Confusion," in *Proc. of SecureComm*, 2005, pp. 194–205.

[17] H. Hu and D. L. Lee, "Range Nearest-Neighbor Query," *IEEE TKDE*, vol. 18, no. 1, pp. 78–91, 2006.

[18] P. Kamat, Y. Zhang, W. Trappe, and C. Ozturk, "Enhancing Source-Location Privacy in Sensor Network Routing," in *Proc. of ICDCS*, 2005, pp. 599–608.

[19] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan, "Incognito: Efficient Full-Domain K-Anonymity." in *Proc. of SIGMOD*, 2005, pp. 49–60.

[20] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam, "l-Diversity: Privacy Beyond k-Anonymity." in *Proc. of ICDE*, 2006.

[21] A. Meyerson and R. Williams, "On the Complexity of Optimal K-anonymity," in *Proc. of ACM PODS*, 2004, pp. 223–228.

[22] M. F. Mokbel, W. G. Aref, and I. Kamel, "Analysis of Multi-Dimensional Space-Filling Curves," *GeoInformatica*, vol. 7, no. 3, pp. 179–209, 2003.

[23] M. F. Mokbel, C. Y. Chow, and W. G. Aref, "The New Casper: Query Processing for Location Services without Compromising Privacy," in *Proc. of VLDB*, 2006, pp. 763–774.

[24] B. Moon, H. Jagadish, and C. Faloutsos, "Analysis of the Clustering Properties of the Hilbert Space-Filling Curve," *IEEE TKDE*, vol. 13, no. 1, pp. 124–141, 2001.

[25] P. Samarati, "Protecting Respondents' Identities in Microdata Release." *IEEE TKDE*, vol. 13, no. 6, pp. 1010–1027, 2001.

[26] H. Samet, *The Design and Analysis of Spatial Data Structures*. Addison-Wesley, 1990.

[27] L. Sweeney, "k-Anonymity: A Model for Protecting Privacy," *Int. J. of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 5, pp. 557–570, 2002.

[28] Y. Tao and D. Papadias, "Historical spatio-temporal aggregation," *ACM Trans. Inf. Syst.*, vol. 23, no. 1, pp. 61–102, 2005.

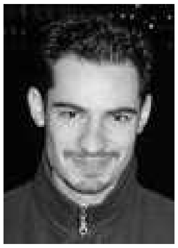
[29] Y. Tao, D. Papadias, and Q. Shen, "Continuous Nearest Neighbor Search." in *Proc. of VLDB*, 2002, pp. 287–298.

[30] Y. Theodoridis, "The R-tree-portal, http://www.rtreeportal.org," 2003. [Online]. Available: http://www.rtreeportal.org

[31] X. Xiao and Y. Tao, "Personalized Privacy Preservation." in *Proc. of SIGMOD*, 2006, pp. 229–240.

Panos Kalnis is an assistant professor at the Computer Science Dept., National University of Singapore. He received his Diploma in Computer Engineering from the Computer Engineering and Informatics Dept., University of Patras, Greece, and his PhD from the Computer Science Dept., Hong Kong UST. His research interests include Anonymity, Peer-to-Peer systems, Mobile Computing, OLAP, Data Warehouses and Spatial Databases.

Gabriel Ghinita is a PhD candidate at the Computer Science Department, National University of Singapore. He obtained his bachelor degree from the Computer Science department at the "Politehnica" University of Bucharest. His research interests include Data Privacy, Spatial Anonymity, Spatial Databases and Peer-to-Peer systems.



Kyriakos Mouratidis received the BSc degree from the Aristotle University of Thessaloniki, Greece, and the PhD degree from the Hong Kong University of Science and Technology, both in computer science. He is an assistant professor in the School of Information Systems, Singapore Management University. His research interests include spatiotemporal databases, data stream processing, and mobile computing.

Dimitris Papadias is a professor in the Department of Computer Science and Engineering, Hong Kong University of Science and Technology (HKUST). Before joining HKUST in 1997, he worked and studied at the German National Research Center for Information Technology (GMD), the National Center for Geographic Information and Analysis (NCGIA), Maine, the University of California, San Diego, the Technical University of Vienna, the National Technical University of Athens, Queens University, Canada, and the University of Patras, Greece. He has published extensively and has been involved in the program committees of all major database conferences including the ACM SIGMOD International Conference on Management of Data, the International Conference on Very Large Data Bases (VLDB), and the International Conference on Data Engineering. He is an associate editor of the VLDB Journal and the IEEE Transactions on Knowledge and Data Engineering, and he is on the editorial advisory board of Information Systems.