7-2009

Spatial Cloaking Revisited: Distinguishing Information Leakage from Anonymity

Kar Way TAN
Singapore Management University, karway.tan.2007@smu.edu.sg

Yimin LIN
Singapore Management University, yimin.lin.2007@smu.edu.sg

Kyriakos MOURATIDIS
Singapore Management University, kyriakos@smu.edu.sg

DOI: https://doi.org/10.1007/978-3-642-02982-0_10

Citation

TAN, Kar Way; LIN, Yimin; and MOURATIDIS, Kyriakos. Spatial Cloaking Revisited: Distinguishing Information Leakage from Anonymity. (2009). *Advances in Spatial and Temporal Databases: 11th International Symposium, SSTD 2009 Aalborg, Denmark, July 8-10, 2009 Proceedings*. 5644, 117-134. Research Collection School Of Information Systems.
Available at: https://ink.library.smu.edu.sg/sis_research/880

Spatial Cloaking Revisited: Distinguishing Information Leakage from Anonymity

Kar Way Tan, Yimin Lin and Kyriakos Mouratidis

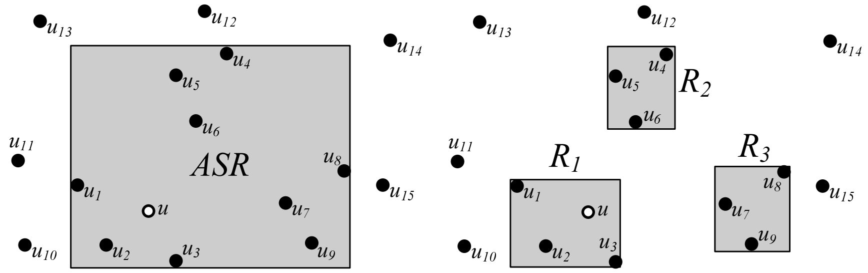
Singapore Management University
School of Information Systems
80 Stamford Road, Singapore 178902
{karway.tan.2007, yimin.lin.2007, kyriakos}@smu.edu.sg

Abstract. Location-based services (LBS) are receiving increasing popularity as they provide convenience to mobile users with on-demand information. The use of these services, however, poses privacy issues as the user locations and queries are exposed to untrusted LBSs. *Spatial cloaking* techniques provide privacy in the form of k-anonymity; i.e., they guarantee that the (location of the) querying user u is indistinguishable from at least k-1 others, where k is a parameter specified by u at query time. To achieve this, they form a group of k users, including u, and forward their minimum bounding rectangle (termed *anonymizing spatial region*, ASR) to the LBS. The rationale behind sending an ASR instead of the distinct k locations is that exact user positions (querying or not) should not be disclosed to the LBS. This results in large ASRs with considerable dead-space, and leads to unnecessary performance degradation. Additionally, there is no guarantee regarding the amount of location information that is actually revealed to the LBS. In this paper, we introduce the concept of *information leakage* in spatial cloaking. We provide measures of this leakage, and show how we can trade it for better performance in a tunable manner. The proposed methodology directly applies to centralized and decentralized cloaking models, and is readily deployable on existing systems.

1 Introduction

The increasing trend of location-aware mobile devices, such as GPS-enabled mobile phones and palm-tops, has lead to a growing market of location-based services (LBS). Users of these services query the LBS to retrieve information about data (points of interest, POI) in their vicinity. The main issue arising in this environment is that the users reveal their locations to the untrusted LBS. In turn, this information may lead to the identity of the users (a process generally termed re-identification) through publicly available information, physical observation, mobile device tracking, etc [1]. The nature of the POIs (e.g., HIV clinics) may disclose sensitive personal information to the LBS, or lead to receipt of unsolicited targeted advertisements (e.g., if the POIs are providers of particular services or products).

To solve the above problem, *spatial cloaking* methods replace the user location with an *anonymizing spatial region* (ASR) prior to sending his/her query to the LBS. The ASR is typically an axis-parallel rectangle[1] that encloses the querying user u and at least $k-1$ additional users; the set of these k users is called the anonymizing set (AS) of u. Parameter k is specified by u at query time and reflects the degree of anonymity required. Figure 1(a) shows a cloaking example where the query originator u, shown as a hollow point, requests for 10-anonymity (i.e., $k = 10$). The figure shows the computed ASR, assuming that the anonymizing set of u additionally contains $u_1, ..., u_9$.



(a) Single ASR (k=10)

(b) Multiple ASRs (k=10)

Fig. 1. Anonymity versus information leakage

Depending on the assumed architecture, there exist centralized and decentralized ASR computation methods. In the first case, the ASR is created by a trusted server (the *anonymizer*) who maintains all user locations [2, 3]. In the second case, the ASR is computed in a collaborative way by the users themselves (assuming that they are mutually trusted and communicate wirelessly) [4, 5]. In either case, the ASR is subsequently sent to the LBS. The latter computes the candidate query answers (i.e., POIs that satisfy the user query) for any possible query position inside the ASR. The set of returned POIs is called *candidate set* CS, and is filtered by the anonymizer or the querying user (in the centralized and the decentralized model, respectively) in order to retrieve the actual query result.

A principle underlying the general approach described above is that user locations should not be disclosed to the LBS. However, if the LBS already knew the locations of all users, there would be no need for an ASR. Anonymity would still be preserved by directly sending the exact user locations in the anonymizing set, e.g., forwarding to the LBS all 10 user locations in Figure 1(a) instead of the ASR. This approach not only would honor anonymity, but it would also lead to a smaller CS and thus a better performance; as we explain later, the main factor affecting the overall performance of the system is the size of the CS (i.e., the

[1] Circular ASRs have also been studied, without however resulting in performance benefits over rectangular ASRs [2].

number of POIs inside the CS). Furthermore, the cost in dollars may be lower, since in commercial LBSs, the amount paid by the user is proportional to the number of POIs provided by the LBS.

Following the above reasoning, one could argue that even if the LBS did not know the user locations, we could break the ASR into smaller sub-ASRs in order to avoid CS results that correspond to dead-space (i.e., regions of the ASR that contain no user). This technique is exemplified in Figure 1(b), where three sub-ASRs (R_1, R_2, and R_3) are used instead of one, to deal with the same querying user u and the same anonymity requirement $k = 10$. According to the principles and objectives of existing spatial cloaking approaches, this multiple sub-ASR method both (i) preserves the anonymity of the querying user, and (ii) does not disclose any user locations. Furthermore, it leads to a smaller CS and hence to a better performance. However, this approach somehow reveals more location information than a single ASR, because the LBS would acquire more precise knowledge about where users are located. Specifically, in Figure 1(a) the LBS would infer that there are some users inside the ASR.[2] On the other hand, the multiple sub-ASRs in Figure 1(b) would disclose additional and more precise information, because now the LBS infers that each of the three rectangles R_1, R_2, and R_3 contains some users.

In this paper, we define the concept of *information leakage* to capture the location information revealed by spatial cloaking, and provide measures to quantify it. We propose the *Information Leakage-aware Cloaking* (ILC) methodology that incorporates this notion into existing techniques and enables control over the trade-off between performance/cost and information leakage. In particular, we forward multiple sub-ASRs to the LBS, constructed in a way which guarantees that anonymity is preserved and that information leakage does not exceed the amount tolerable by the users/system. Note that this contrasts with existing methods, where there is no control over the amount of location information revealed to the LBS. Our method is readily applicable to existing spatial cloaking systems, and works transparently in both the centralized and the decentralized model.

The rest of the paper is structured as follows. Section 2 surveys related work. Section 3 states out assumptions and objectives, and presents definitions central to our work. Section 4 presents our methodology and shows how it can be incorporated into available spatial cloaking systems. Section 5 experimentally evaluates our approach, studying the trade-offs between performance and information leakage. Finally, Section 6 concludes the paper with directions for future work.

2 Background and Related Work

k-anonymity [6, 7] has been used for publishing sensitive data (e.g., medical records) in a way that each record is indistinguishable from at least k-1 others.

[2] Note that the LBS does not know the value of k, but only sees the query ASR.

In the context of location-based services, spatial k-anonymity is achieved by obfuscating the locations of querying users so that they cannot be identified with a probability higher than $1/k$. Location obfuscation is performed by a *cloaking algorithm*. Most systems adopt the *centralized architecture* [2,3]. In this setting, the cloaking algorithm is executed by a trusted third party (anonymizer), which is regularly being updated with the most current user locations. On the other hand, in a *decentralized architecture* [4,5], no anonymizer is required. Instead, the users collaboratively construct ASRs communicating via an overlay network (e.g., a peer-to-peer system). Our proposed methodology affects primarily the cloaking process and as such it can be applied to both the centralized and the decentralized architectures. Section 2.1 reviews existing cloaking techniques, and Section 2.2 discusses processing techniques for cloaked queries. Section 2.3 describes alternative location privacy models.

2.1 Cloaking Techniques

Interval Cloak [1] is one of the first cloaking techniques. The anonymizer indexes the users with a Quad-tree [8]. To form an ASR for user u, *Interval Cloak* descends the Quad-tree up to the topmost node that contains at least k users (including u). The extent of this node is returned as the ASR. In Figure 2, if user u_1 issues a query with $k = 2$, *Interval Cloak* will search till quadrant $[(0,0),(1,1)]$ which contains less than 2 users. Then, it will backtrack for one level, and return the parent quadrant $[(0,0),(2,2)]$ as the ASR. The returned quadrant may contain much more than k users, burdening query processing at the LBS.

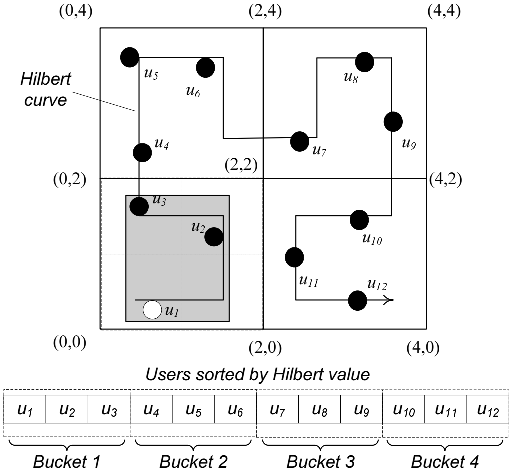


Fig. 2. Cloaking example

Casper Cloak [3] is similar to *Interval Cloak*, with two major differences. First, *Casper Cloak* identifies and accesses the leaf level of the Quad-tree directly through the use of a hash table. Second, instead of immediately backtracking to the parent quadrant, it first checks the two neighboring quadrants to see if their combination with the user quadrant contains k (or more) users. In Figure 2, if u_1 issues a query with $k = 2$, *Casper Cloak* first checks the neighboring quadrants $[(0, 1), (1, 2)]$ and $[(1, 0), (2, 1)]$. If combination with one of them results in k users, then this composite rectangle is returned as the ASR. In this example, rectangle $[(0, 0), (1, 2)]$ is returned.

[2] shows that there are situations where anonymity is breached with the above methods, and proves that absolute anonymity can be guaranteed if *reciprocity* is honored. Reciprocity is defined as follows:

Definition 1. *Let $AS_k(u)$ be the anonymizing set of u for anonymity degree k. A cloaking algorithm satisfies reciprocity iff (i) $AS_k(u)$ contains at least k users, and (ii) for every user u' in $AS_k(u)$ it holds that $AS_k(u') \equiv AS_k(u)$ (i.e. all users in $AS_k(u)$ have the same AS).*

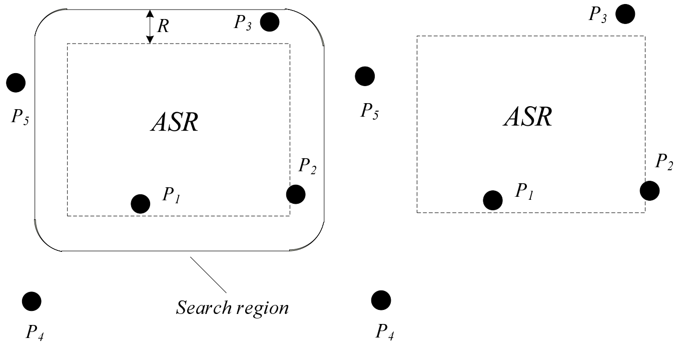
[2] proposes *Hilbert Cloak*, an algorithm that satisfies this property. The users are sorted according to the Hilbert space-filling curve [9]. The sorted sequence is equally divided into buckets of k consecutive users. AS is formed by the bucket that contains the querying user u. The reported ASR is computed as the minimum bounding rectangle of the AS. In Figure 2, if u_1 issues a query with $k = 3$, then 4 Hilbert buckets are created as shown at the bottom of the figure. User u_1 belongs to the first bucket, and its AS includes u_1, u_2, u_3. The derived ASR is the shaded bounding box. Due to its simplicity, *Hilbert Cloak* has been applied to decentralized systems too [4].

2.2 Query Processing at the LBS

The two most common spatial queries are the *Range Query* and the *Nearest-Neighbor (NN) Query*. Given only an ASR and the query type/parameters, the LBS needs to search for the POIs that satisfy the query for any possible user location within the ASR. Typically, the LBS stores the POIs in secondary storage, indexed by an R-tree [10, 11]. If an R-range query is given, the LBS computes CS as the union of all POIs that fall inside the ASR or are within distance R from its boundary. In the example of Figure 3(a), the LBS expands the ASR (shown with a dashed contour) by R, and performs an ordinary range query. The CS contains P_1, P_2, and P_3.

If a K-NN query is given, the CS contains the union of K nearest POIs [3] to any point within the ASR. To derive the CS for a NN query (i.e., $K = 1$) in Figure 3(b), the LBS needs to retrieve (i) all objects located inside the ASR (i.e., P_1) and (ii) the NN of any location along the boundary of the ASR (i.e., P_2, P_3, P_5). The latter component is processed using the *linear NN* method of [12] for each of the 4 edges of the ASR; the input to this method is one or more line segments, for which NNs are found in a single R-tree traversal.

[3] Note that parameter K used here is different from the k-anonymity requirement used in the anonymity context.



(a) Range Query (b) Nearest Neighbor Query

Fig. 3. Types of Query Processing at LBS

2.3 Alternative Location Obfuscation Approaches

There exist alternative location privacy approaches. [13] ignores non-querying users, and instead groups only querying users among themselves. [14] proposes a location privacy method specifically for approximate NN processing. In [15, 16] the user u forwards to the LBS a set of dummy locations in addition to his/her own. In [17], the user sends only a fake location to the LBS and incrementally retrieves its nearest neighbors. [18] applies *private information retrieval* to process NN queries. The above methods cannot ensure k-anonymity, are limited in the type of queries supported, and/or incur prohibitively high query processing cost.

3 Preliminaries

In this section we state the assumptions underlying our approach (in Section 3.1), our central observation and our design objectives (in Section 3.2). In Table 1 we list frequently used acronyms/symbols, and their interpretation.

3.1 Assumptions

The *Information Leakage-aware Cloaking* (ILC) methodology applies to (and is orthogonal to the choice between) the centralized and the decentralized cloaking models. However, to avoid confusion and for the sake of tangibility, we assume the centralized model in our examples unless otherwise stated. We focus on spatial (i.e., 2-dimensional) user and POI locations. Similar to existing spatial cloaking systems, we consider that the users (forming set U) are mobile, and constantly update the anonymizer with their most recent locations. The set of POIs is static, and it is indexed by a disk-resident R-tree. Note that ILC deals mostly with the cloaking part and, as such, indexing at the LBS side or the mobility

Term	Description
LBS	Location-Based Service
POI	Point Of Interest
ASR	Anonymizing Spatial Region
AS	Anonymizing Set
CS	Candidate Set
U	The set of users in the system
k	Anonymity parameter
m	Strictness on information leakage parameter
IL	Degree of information leakage

Table 1. Description of acronyms and symbols

of the POIs has little impact on it; alternative contexts can be dealt with in a straightforward manner. We focus on the most common spatial queries, that is, *snapshot*[4] range and nearest neighbor (NN) queries. However, our technique can be directly incorporated into the model of [19] to capture *continuous* queries too; this extension is discussed in Section 4.4. Regarding the communication channel we assume that:

1. The connection between the querying user u and the anonymizer (in the centralized model) or among users (in decentralized systems) is encrypted and secure.
2. The communication channel between the anonymizer (or the users, in a decentralized system) and the LBS needs not be secure.

Point 1 above implies that eavesdropping is not possible for the LBS, and that k is unknown to it. Point 2 practically implies that the LBS is not the only possible adversary, but our method should ensure anonymity and controlled information leakage versus any malicious entity that may intercept the cloaked queries (on their way to the LBS). For simplicity, we consider the LBS as the adversary, but ILC is safe against any of the aforementioned types of entities. Note that ensuring the authenticity of the POIs reported to the users is outside the scope of this paper; result verification methods (e.g., [20–22]) could be used in conjunction with ILC to detect any man-in-the-middle tampering with the results.

3.2 Main Observation and Design Objectives

Our motivating observation is that information leakage requirements have always existed in spatial anonymity approaches, but they have never been identified and treated with independently. Specifically, the methods described in Section 2.1 assume two kinds of adversaries:

[4] Term snapshot refers to queries that are evaluated once and then terminate. It is used to distinguish from continuous evaluation where the queries are standing and request constant updating of their results.

- **User-aware adversaries**: Adversaries of this type know the user locations. To achieve anonymity against such adversaries, it suffices to send to the LBS the exact positions of all users in the AS. If the AS is formed in a reciprocal way, anonymity is guaranteed. By definition, this would incur the smallest possible CS for the specific AS.
- **User-unaware adversaries**: Adversaries of this type do not know the user locations. Concealing the position of the querying user u from such adversaries is easy and can be done arbitrarily (e.g., by sending to the LBS a rectangle that encloses u). What is important here is that user-unaware adversaries stay unaware of exact user locations (be them querying or not).

What has been implicitly assumed by previous systems is that there exist both kinds of the above adversaries at the same time (plus possibly adversaries with partial user location knowledge), and that they must be dealt with collectively; the AS was "masked" with a minimum bounding rectangle, so that exact user locations are not revealed to user-unaware adversaries, while anonymity is ensured even against user-aware ones. The sacrifice made in this approach is that the CS contains more POIs (than sending the AS locations directly).

Our observations here is that two different concepts underlie this design principle (i.e., anonymity and information leakage), and that the anonymity-centric approach taken so far ensures anonymity, but fails to control or even to quantify the degree of location information disclosed to user-unaware adversaries. Thus, our first contribution is to provide a meaningful measure of information leakage, and then suggest a methodology (i.e., ILC) to control it; the twofold objective of ILC is to ensure user anonymity *and* guarantee no more than the permissible degree of information leakage. In terms of anonymity, we adopt its strict, reciprocity-based definition (described in Section 2.1); recall that reciprocity is a sufficient (though not necessary) condition to achieve strongly k-anonymous services [2]. We elaborate on the information leakage requirements in Section 4.

Subject to the degree of information leakage tolerable, our second objective is to reduce the CS size; this is the primary factor that (i) determines the dollar cost paid by the user to the LBS, and (ii) determines the end-to-end query response time. Regarding (i), commercial LBSs often charge by the amount of information provided, i.e., the number of POIs returned. In terms of end-to-end response time, in the centralized model, the experiments of [2] and [3] indicate that the major performance factor is the I/O time spent at the LBS and secondarily the communication cost. Both these costs are proportional to the size of the CS.[5] In the decentralized model, the CS (on its way from the LBS to the querying user u) must pass through the overlay user network, and it must subsequently be filtered by u to retrieve the actual query result. The communication and processing costs incurred prolong the end-to-end time, but also consume the (typically scarce) power resources of the user devices. Thus,

[5] The page accesses performed at the higher levels of the POI R-tree are minimal compared to the POI (leaf) level. That is, the I/O cost is roughly proportional to the CS size.

our aim is to exploit any leeway in terms of information leakage to reduce the CS size.

4 Information Leakage-aware Cloaking

In this section we define a measure of information leakage and describe the ILC framework.

4.1 Measuring Information Leakage

Our approach is to control information leakage via a parameter m. This parameter is specified by the system (e.g., the anonymizer) as a requirement from the cloaking mechanism. We establish that:

Definition 2. *The strictness on information leakage of a cloaking algorithm is m iff any ASR (or sub-ASR) forwarded to the LBS contains at least m users.*

Intuitively, information leakage is inversely proportional to m, hence we quantify IL, the degree of information leakage as $\frac{1}{m}$. It holds that $\frac{1}{|U|} \leq IL \leq 1$, where $|U|$ is the total number of users in the system. Case $IL = 1$ (maximum information leakage) corresponds to $m = 1$, where exact user locations may be revealed to the LBS. Case $IL = \frac{1}{|U|}$ (minimum information leakage) corresponds to $m = |U|$, where a single ASR enclosing all users is sent to the LBS. In the situation illustrated in Figure 1(b), $m = 3$ and $IL = \frac{1}{3}$, because the smallest number of users contained in any sub-ASR is 3 (regardless of the fact that R_1 contains 4).

Regarding the rationale behind the IL measure, one may wonder why we do not express it in an absolute way as the maximum accuracy that the LBS would get about individual user locations. In other words, this alternative would require that each constructed ASR (or sub-ASR) would not be narrower than some threshold δ_x on the x axis and another threshold δ_y on the y axis. The reason for disqualifying this method is that it fails to capture the user distribution; the distribution can be easily estimated using publicly available information. For example, Figure 4 plots a real dataset of 25,000 locations in North America that could model our users. Illustrated rectangles ASR_1 and ASR_2 have the same extents (say δ_x and δ_y), but the first lies in a very sparse area, while the second includes numerous users (covering, for instance, the highly populated New York city). Clearly, ASR_1 reveals much more information to the LBS than ASR_2 regarding where users lie. Thus, we select a relative IL measure using parameter m.

Another approach is to define IL according to the area of the ASR (or the minimum area of any sub-ASR constructed). Figure 5 demonstrates the weaknesses of this approach with a counter-example. Here, after computing the AS of u (say, using *Hilbert Cloak*), we construct an ASR in the form of a poly-line, i.e., a set of connected line segments. The poly-line passes from all users in AS, and its turning points are selected in a randomized way so that they do not coincide with any user location. This ASR has zero area; under an area-based

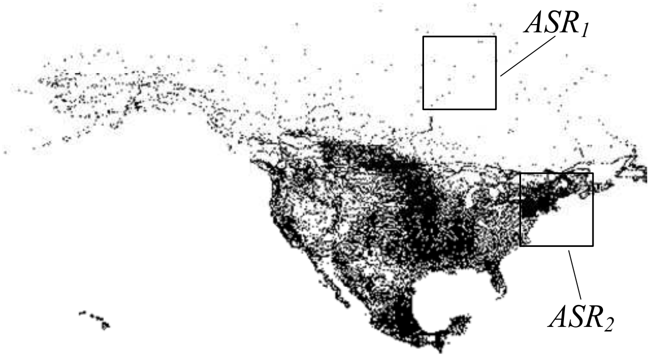


Fig. 4. Shortcomings of an absolute definition based on spatial precision

definition, this plain method would incur no information leakage[6]. However, it is obvious that this is not the case, as the location information revealed now is much more precise. Actually, the situation can be even worse if, for instance, the users move on a road network; the LBS could compute the user positions by retrieving the intersections between the ASR poly-line and the edges of the road network. We thus avoid an area-based definition, and adopt the one described in the beginning of the section.

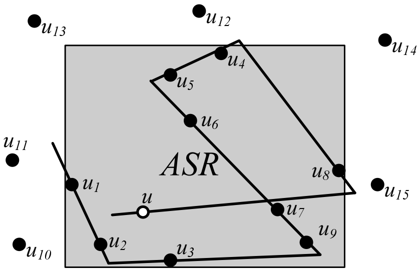


Fig. 5. Shortcomings of an area-based definition

Before presenting the ILC cloaking methodology, we need to clarify a few issues regarding the distinction between anonymity (defined by parameter k) and IL (defined by m):

- k and m implement different requirements and control different cloaking functions. However, a higher k implies higher privacy, just like a higher m implies lower information leakage. In this sense, increasing these parameters leads to "safer" cloaking in a general point of view.

[6] Furthermore, this approach would significantly reduce the CS size. Imagine an R-range query. The LBS would only return POIs within distance R from the ASR poly-line, leading to a much smaller CS than a traditional rectangular ASR.

- k is specified by each querying user individually, while m is a system-wide parameter (e.g., defined by the anonymizer); m is a global parameter, because it does no longer reflect individual user preferences, but the release of information from the system (as a whole) to the LBS. Note, however, that ILC can be easily adapted to contexts where it makes sense for m to be user-specific.
- Controlling information leakage does not violate anonymity (or reciprocity). In Figure 1(b), for example, if the AS is derived by a reciprocal anonymization algorithm, the LBS still cannot identify the querying user with a probability greater than $\frac{1}{k} = \frac{1}{10}$.
- Typically, m is considerably smaller than k, because the information that user u is at some location *and* asks a particular query (relating to anonymity) is more sensitive than simply knowing that there is a user at location u (relating to information leakage). In the following, we focus on situations where $m < k$, but we also consider the rare scenario where $m \geq k$.

4.2 The Multiple ASR Approach

The main idea in ILC is to cloak the AS using multiple (sub-)ASRs, none of which contains less than m users. In addition to providing the desired degree of information hiding (i.e., keeping IL lower than its maximum permissible value), we attempt to reduce the CS size by limiting the dead-space within the ASRs. The general idea in this approach is similar to Figure 1(b). Note that ILC does not violate (or interfere with) anonymity, because it does not affect the AS itself.

Although ILC can be applied in conjunction with other cloaking methods, here we choose to incorporate it into *Hilbert Cloak* since it is the current state-of-the-art. The AS for a user u (as output by *Hilbert Cloak*) has the form of a Hilbert-sorted list. ILC splits the AS (Hilbert-sorted) list into m-buckets. Each bucket has exactly m users, except for the last one which contains from m up to $2m - 1$ users. ILC returns a sub-ASR for each m-bucket, by computing its minimum bounding rectangle. The pseudo-code for sub-ASR generation is illustrated in Figure 6. Figure 7 shows an example where $k = 10$ and $m = 3$. The user order on the Hilbert curve is shown at the bottom of the figure. The AS output by *Hilbert Cloak* contains users $u_1, u_2, ..., u_{10}$; *Hilbert Cloak* would return their bounding box as the ASR. Instead, ILC breaks the AS into m-buckets (where $m = 3$), and creates one sub-ASR for each of them. This leads to sub-ASRs SA_1, SA_2, SA_3 shown striped. Observe that the last m-bucket/sub-ASR contains 4 users (i.e., more than $m = 3$).

Note that, given the AS, the construction of sub-ASRs is not concerned with which of the users was the querying one, and thus no reciprocity requirement underlies the IL-related handling. Therefore, ILC could work with any other sub-ASR creation method, subject to the IL constraint. Our objective is not to propose the best such algorithm, as similar bucketization problems are well-studied in the spatial database and computational geometry literatures. We use the aforementioned Hilbert-based technique merely to provide an example where ILC is readily deployable on an existing system that uses the *Hilbert*

Algorithm: Creating IL-aware sub-ASRs

```
1. Given a Hilbert-sorted AS
2. Split the AS into m-buckets
3. For each m-bucket do
4.    Create a sub-ASR as the minimum bounding rectangle of users inside
5. Return the list of all sub-ASRs computed in Step 4
```

Fig. 6. Algorithm for deriving sub-ASRs in ILC

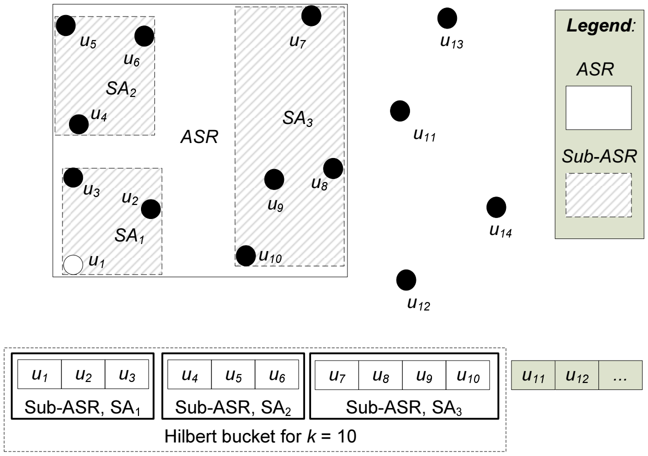


Fig. 7. ILC example

Cloak method (the system may be centralized or decentralized, with sub-ASR computation performed by the anonymizer or the users, respectively). It is worth mentioning that we did experiment with alternative partitioning techniques, such as STR [23] and the R$^+$-tree splitting algorithm [24], which however performed similar to the Hilbert-based method.

In the above discussion we assumed that $m < k$. However, in the rare case where $m \geq k$, a single ASR is returned. Moreover, in situations where $m > k$, if the ASR contains fewer than m users[7], we enlarge it in a greedy way so that it encloses m users with the minimum area increase. Greedy enlargement is permissible, because as explained above, reciprocity is not a requirement when information leakage is considered.

4.3 Query Processing at the LBS

At the LBS side, processing multiple ASRs is based on the same primitive functions that handle a single ASR. In both the range and nearest neighbor cases,

[7] Here containment refers to the spatial domain, and not to the Hilbert range of the AS.

processing is possible in a single traversal of the POI R-tree, so that multiple reads of the same disk pages are avoided. Specifically, for an R-range query, the LBS descends the R-tree visiting any node with minimum distance smaller than R from any of the sub-ASRs. In the case of K-NN processing, the CS contains the K-NN of any possible query point inside any sub-ASR. Handling is similar to single ASR processing, the difference being that instead of 4 line segments (i.e., single ASR edges), there are 4 segments for each sub-ASR input to the algorithm of [12] (note that this work describes how batch processing is possible in a single R-tree traversal, and that this feature is already being exploited in the standard *Hilbert Cloak* technique).

4.4 Different Query Types

So far we considered simple snapshot queries, but ILC can be easily applied to continuous processing too, extending the method of [19]. In particular, sub-ASR computation can be performed in the first evaluation of the query, with users in each m-block forming a group. In subsequent timestamps, the sub-ASRs can be found as the minimum enclosing rectangles of each group, subject to the updated positions of its users. An alternative to that, is to perform sub-ASR computation in every timestamp. This latter approach is expected to lead to smaller ASRs, and thus to smaller CS.

5 Experimental Evaluation

In this section, we empirically evaluate the performance of the ILC approach. Section 5.1 describes our experimental setting, while Section 5.2 presents the results and their interpretation.

5.1 Experimental Setting

We executed the experiments using prototypes written in C++ on an Intel Pentium IV 3GHz machine. We use a real dataset as the POIs available at the LBS; the dataset (denoted as NA) contains 569,120 endpoints of main road segments in North America, and is acquired from www.maproom.psu.edu/dcw. It is normalized into a 10,000 by 10,000 data-space. The set of users U is formed by randomly picking a percentage (1%, 5%, 10%, 15%, or 20%) of the POIs to serve as the users; this choice represents a realistic scenario where the users follow the distribution of the queried facilities (POIs). However, we explore different user and POI distributions towards the end of the section, where the POIs correspond to 1,314,620 locations in Los Angeles (dataset denoted as LA, and acquired from www.rtreeportal.org).

We use a centralized architecture due to its proliferation, and assume that the anonymizer stores the users in main memory, while the LBS indexes the POIs with a disk-resident R*-tree. We use the *Hilbert Cloak* as a basis, and denote its traditional (single ASR) version as HC, and our adaptation as ILC. We quantify the performance benefits of ILC in terms of the size of the candidate set retrieved ($|CS|$); as we demonstrate, the communication cost and the I/O cost (and, as a result, the total end-to-end response time too) are proportional to $|CS|$. In each experiment, we vary one parameter, while setting the remaining to their default values. The parameter ranges and default values are shown in Table 2.

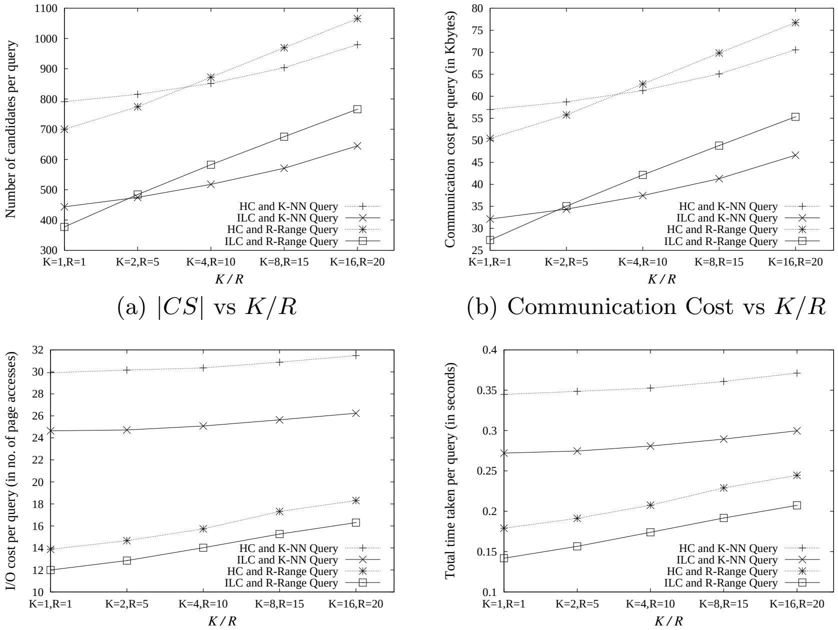
Parameter	**Default**	**Range**
Anonymity parameter k	50	5,10,50,100,150
Strictness on Information leakage parameter m	5	2, 3, 5, 7, 10, 15
Number of NNs K	4	1, 2, 4, 8, 16
Range R	10	1, 5, 10,15,20
User-to-POI percentage ρ	10%	1, 5, 10, 15, 20 (%)

Table 2. Experiment parameters and their respective default values

5.2 Experimental Results

The first set of experiments (in Figure 8) explores the effect of query selectivity, by varying K (in the case of K-NN queries) and R (in the case of R-Range queries). Additionally, it verifies our claim that the I/O time, the communication cost and the end-to-end time are proportional to $|CS|$. Figure 8(a) plots the CS size. HC and ILC exhibit the same pattern of increase as K and R increase. However, ILC has approximately 34%-44% (28%-46%) improvement in $|CS|$ over HC on the average for NN (range) queries. Note that the two lower curves correspond to ILC and the upper ones to HC. This is expected as HC returns numerous candidate POIs corresponding to ASR regions that contain no users. The gap between ILC and HC shrinks for large K or R (to around 34% and 28%, respectively) as for high selectivity, the union of the search regions of the sub-ASRs converges to the search region of a single ASR.

Figure 8(b) plots the total communication cost between the anonymizer and the LBS in terms of Kbytes transferred towards either direction. We assume that each coordinate is 4 bytes, that an ASR (or sub-ASR) is represented by 4 coordinates, and that each POI has an additional 64 byte non-spatial information attached to it. Figure 8(c) shows the I/O cost (of query processing at the LBS) in terms of the number of R-tree node accesses; note that NN queries require more I/Os than ranges because the linear NN algorithm of [12] accesses a considerable amount of POIs that do not belong to the CS. Figure 8(d) sums up all costs, and plots the end-to-end query response time assuming a 10 Mbps connection between the anonymizer and the LBS. From the aforementioned three figures it becomes clear that the individual communication and I/O costs, as well as the overall result delay are proportional to $|CS|$. As such, and in order to avoid cluttering the paper with correlated figures, we use $|CS|$ as the key measurement for ILC's performance in the subsequent experiments.



(a) $|CS|$ vs K/R

(b) Communication Cost vs K/R

(c) I/O Cost vs K/R

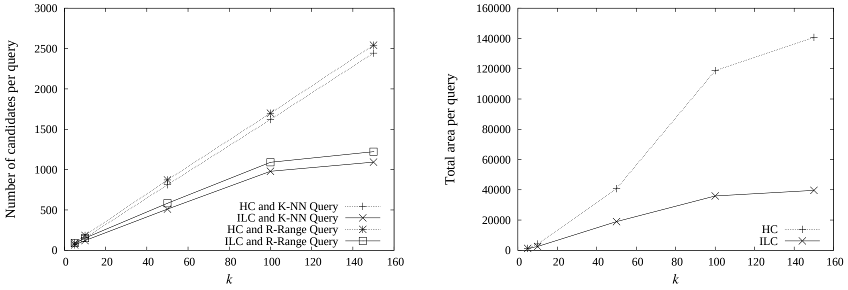
(d) End-to-end Cost vs K/R

Fig. 8. Effect of query selectivity (K for NN, and R for range queries)

Figure 9(a) explores the effect of k on $|CS|$, while setting m to 5. In HC, $|CS|$ grows almost linearly to k, because most candidate POIs returned fall inside the (growing) ASR. However, in ILC, we can see that performance degradation with k is slighter. This is because the sub-ASRs prune relatively more dead-space from a large ASR. Thus, ILC scales better than HC with k. To support our previous claim regarding the dead-space pruned, in Figure 9(b) we show the effect of k on the ASR area and the cumulative sub-ASR area per query, for HC and ILC, respectively. In the case of ILC, the gradient of the curve decreases significantly for $k > 100$.

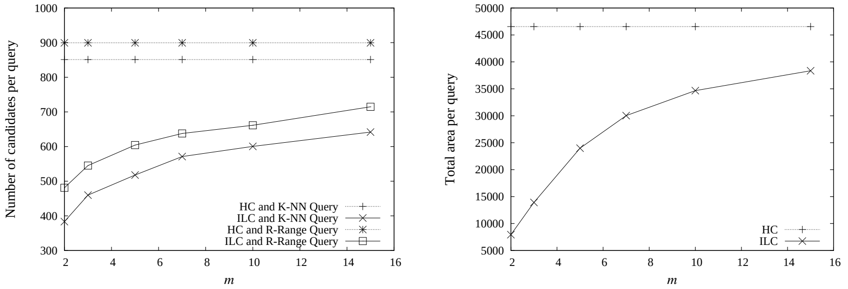
Figures 10(a) and 10(b) illustrate the effect of m on $|CS|$ and on the total ASR/sub-ASR area, respectively. Parameter m does not affect HC since it is only introduced in ILC. As such, all the HC related curves are horizontal lines. Figure 10(a) shows the trade-off between information leakage and performance. As m increases (i.e., IL decreases), the performance improvement of ILC over HC decreases because there are fewer sub-ASRs and, thus, weaker dead-space pruning. Intuitively, the more information hidden from the LBS, the higher the cost. Figure 10(b) supports our previous claim regarding the dead-space pruned; it shows clearly that the total sub-ASR area per query increases with m.

Figure 11(a) shows the effect of user density (denoted by ρ), e.g., $\rho = 10\%$ implies that there are 56,912 users, which is one tenth of the POI cardinality.



(a) $|CS|$ vs k (b) Total ASR area per query vs k

Fig. 9. Effect of varying degree of anonymity, k

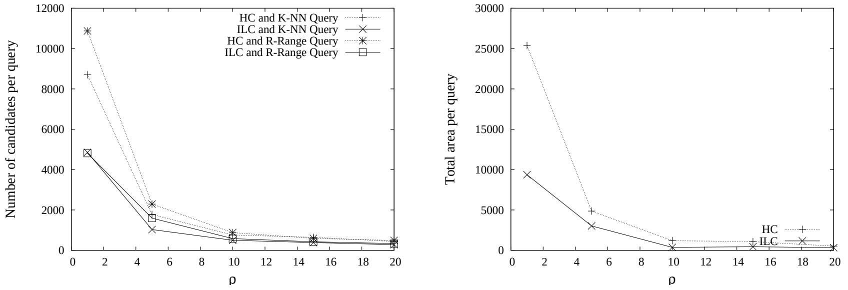


(a) $|CS|$ vs m (b) Total ASR area per query vs m

Fig. 10. Effect of varying strictness on information leakage, m

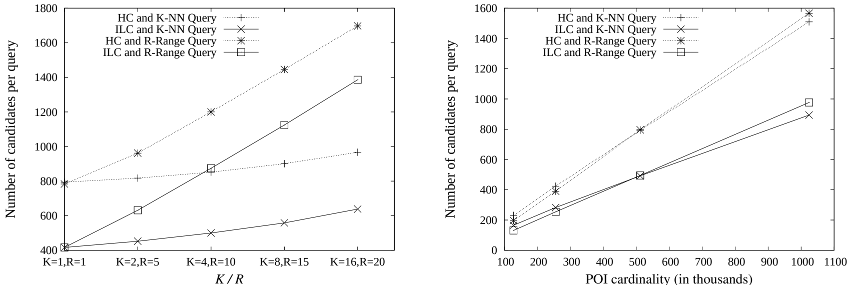
ILC has 55% to 63% the CS size of HC. The gains of ILC drop slightly for large ρ, because a dense user set implies an already small amount of dead-space in the (single) ASR. Therefore, the benefits of using multiple sub-ASRs become smaller, but ILC still performs significantly better than HC. This trend is also obvious in Figure 11(b), which plots the total ASR/sub-ASR area in the same experiment.

In Figure 12(a), we use LA as the POI dataset, and keep the same users as in previous experiments (i.e., following the NA distribution). The observed trends are similar to the results in Figure 8(a), even though the relative cost of range queries is higher than NN. In Figure 12(b) we generated synthetic POI sets, using a Zipfian distribution (with parameter 0.8) and varying their cardinality from 128K up to 1M POIs. The user set is the same as before. ILC scales better than HC; note that the two lower curves correspond to ILC. The results in Figure 12 show the generality of ILC, and its superiority over the traditional (single ASR) approach, regardless of data skewness and user/POI distribution.



(a) $|CS|$ vs user density ρ (b) Total ASR area per query vs ρ

Fig. 11. Effect of user density, ρ



(a) Using LA as POI set: $|CS|$ vs K/R (b) Zipf POIs: $|CS|$ vs POI cardinality

Fig. 12. Effect of different POI datasets

6 Conclusion

In this paper, we define the concept of information leakage in anonymous location-based queries. We describe meaningful leakage measures and propose a methodology to control it. Compared to previous systems, our technique can guarantee that information leakage does not exceed its maximum permissible degree and, moreover, it leads to significant performance benefits. Our method can be easily incorporated into existing cloaking systems and it is applicable to both the centralized and the decentralized cloaking models. A promising direction for future work is to design incremental versions of our methodology suited for continuous query evaluation, in order to improve upon the straightforward extensions of Section 4.4.

References

1. Gruteser, M., Grunwald, D.: Anonymous usage of location-based services through spatial and temporal cloaking. In: MobiSys. (2003)

2. Kalnis, P., Ghinita, G., Mouratidis, K., Papadias, D.: Preventing location-based identity inference in anonymous spatial queries. IEEE Transactions on Knowledge and Data Engineering **19**(12) (2007) 1719–1733
3. Mokbel, M.F., Chow, C.Y., Aref, W.G.: The new casper: Query processing for location services without compromising privacy. In: VLDB. (2006) 763–774
4. Ghinita, G., Kalnis, P., Skiadopoulos, S.: Prive: anonymous location-based queries in distributed mobile systems. In: WWW '07: Proceedings of the 16th international conference on World Wide Web, New York, NY, USA, ACM (2007) 371–380
5. Chow, C.Y., Mokbel, M.F., Liu, X.: A peer-to-peer spatial cloaking algorithm for anonymous location-based service. In: GIS. (2006) 171–178
6. Samarati, P.: Protecting respondents' identities in microdata release. IEEE Trans. Knowl. Data Eng. **13**(6) (2001) 1010–1027
7. Sweeney, L.: k-anonymity: a model for protecting privacy. Int. J. Uncertain. Fuzziness Knowl.-Based Syst. **10**(5) (2002) 557–570
8. de Berg, M., van Kreveld, M., Overmars, M., Schwarzkopf, O.: Computational Geometry: Algorithms and Applications (Second Edition). Springer-Verlag (2000)
9. Butz, A.R.: Alternative Algorithm for Hilbert's Space-Filling Curve. IEEE Trans. Comput. **20**(4) (1971) 424–426
10. Guttman, A.: R-trees: A dynamic index structure for spatial searching. In: SIGMOD Conference. (1984) 47–57
11. Beckmann, N., Kriegel, H.P., Schneider, R., Seeger, B.: The r*-tree: An efficient and robust access method for points and rectangles. In: SIGMOD Conference. (1990) 322–331
12. Tao, Y., Papadias, D.: Spatial queries in dynamic environments. ACM Trans. Database Syst. **28**(2) (2003)
13. Gedik, B., Liu, L.: Location privacy in mobile systems: A personalized anonymization model. ICDCS (2005) 620–629
14. Khoshgozaran, A., Shahabi, C.: Blind Evaluation of Nearest Neighbor Queries Using Space Transformation to Preserve Location Privacy. In: SSTD. (2007)
15. Duckham, M., Kulik, L.: A Formal Model of Obfuscation and Negotiation for Location Privacy. In: PERVASIVE. (2005)
16. Kido, H., Yanagisawa, Y., Satoh, T.: An Anonymous Communication Technique using Dummies for Location-based Services. In: ICPS. (2005)
17. Yiu, M.L., Jensen, C.S., Huang, X., Lu, H.: SpaceTwist: Managing the Trade-Offs Among Location Privacy, Query Performance, and Query Accuracy in Mobile Services. In: ICDE. (2008)
18. Ghinita, G., Kalnis, P., Khoshgozaran, A., Shahabi, C., Tan, K.L.: Private Queries in Location Based Services: Anonymizers are not Necessary. In: SIGMOD Conference. (2008)
19. Chow, C.Y., Mokbel, M.F.: Enabling private continuous queries for revealed user locations. In: SSTD. (2007) 258–275
20. Devanbu, P.T., Gertz, M., Martel, C.U., Stubblebine, S.G.: Authentic third-party data publication. In: DBSec. (2000) 101–112
21. Pang, H., Jain, A., Ramamritham, K., Tan, K.L.: Verifying completeness of relational query results in data publishing. In: SIGMOD Conference. (2005) 407–418
22. Li, F., Hadjieleftheriou, M., Kollios, G., Reyzin, L.: Dynamic authenticated index structures for outsourced databases. In: SIGMOD Conference. (2006) 121–132
23. Kanth, K.V.R., Ravada, S., Sharma, J., Banerjee, J.: Indexing medium-dimensionality data in oracle. In: SIGMOD Conference. (1999) 521–522
24. Sellis, T.K., Roussopoulos, N., Faloutsos, C.: The r+-tree: A dynamic index for multi-dimensional objects. In: VLDB. (1987) 507–518