On the Formation of Historically k-Anonymous Anonymity Sets in a Continuous LBS

Rinku Dewri[1], Indrakshi Ray[2], Indrajit Ray[2], and Darrell Whitley[2]

[1] University of Denver, Denver, CO 80208, USA
rdewri@cs.du.edu
[2] Colorado State University, Fort Collins CO 80523, USA
{iray,indrajit,whitley}@cs.colostate.edu

Abstract. Privacy preservation in location based services (LBS) has received extensive attention in recent years. One of the less explored problems in this domain is associated with services that rely on continuous updates from the mobile object. Cloaking algorithms designed to hide user locations in single requests perform poorly in this scenario. The historical k-anonymity property is therefore enforced to ensure that all cloaking regions include at least k objects in common. However, the mobility of the objects can easily render increasingly bigger cloaking regions and degrade the quality of service. To this effect, this paper presents an algorithm to efficiently enforce historical k-anonymity by partitioning of an object's cloaking region. We further enforce some degree of directional similarity in the k common peers in order to prevent an excessive expansion of the cloaking region.

Keywords: historical k-anonymity, continuous LBS, anonymity sets.

1 Introduction

Application domains are potentially endless with location-tracking technology. These applications deliver customized information based on the location of a mobile object. The services can be classified into two types – *snapshot* LBS where the current location of the mobile object is sufficient to deliver the service, and *continuous* LBS where the mobile object must periodically communicate its location as part of the service agreement. For example, a Pay-As-You-Drive insurance service must receive location updates from the mobile object to bill the consumer accurately. A serious concern surrounding their acceptance is the potential usage of the location data to infer sensitive personal information about the mobile users. With access to the location data, sender anonymity can be violated even without the capability to track a mobile object. We refer to this class of adversaries as *location-unaware adversaries.* Such adversaries use external information to perform attacks resulting in restricted space identification, observation identification and location tracking [1].

S. Jajodia and J. Zhou (Eds.): SecureComm 2010, LNICST 50, pp. 71–88, 2010.

1.1 Motivation

Location obfuscation is one of the widely researched approaches to safeguard location anonymity. This technique guarantees that the location data received at the LBS provider can be associated back to more than one object – to at least k objects under the *location k-anonymity* model [1]. For this, a *cloaking region* is communicated to the service provider instead of the actual location. A k-anonymous cloaking region contains at least $k-1$ other mobile objects besides the service user. However, this approach is not sufficient to preserve privacy in a continuous LBS. In the continuous case, an object maintains an ongoing session with the LBS, and successive cloaking regions may be correlated to associate the session back to the object. Such *session associations* reveal the trajectory of the involved object, and any sensitive information thereof. Assuring that every cloaking region contains k objects is not sufficient since the absence of an object in one of the regions eliminates the possibility that it is the session owner. Performing such elimination is much easier for a *location-aware adversary* who has the capability to monitor users. This class of adversaries has exact location information on one or more objects and uses it to eliminate possibilities and probabilistically associate the session to consistently existing objects. It may seem that these attacks can be avoided by using a different identifier for every cloaking region. However, location data can still be correlated using techniques such as multi-target tracking [2]. Besides, the provider needs to be able to distinguish updates from the same object in order to maintain service quality [3].

Session association attacks can be avoided if it can be assured that every cloaking region in a session contains k common objects. This is referred to as *historical k-anonymity* [4]. However, as a result of the movement of objects, a historically k-anonymous cloaking region is very likely to grow in size over time, thereby deteriorating service quality. Without the proper strategies to control the size of the cloaking region, historical k-anonymity is only a theoretical extension of k-anonymity for continuous LBS. The work presented in this paper is the first known attempt that identifies the issues in effectively enforcing historical k-anonymity.

1.2 Related Work

While significant research has gone into algorithms that enforce k-anonymity [1,5,6,7], very few of them address historical k-anonymity. Gruteser and Liu specifically investigate privacy issues in continuous LBS [8]. They introduce the location inference problem where an adversary can infer supposedly hidden locations from prior or future location updates. Hoh and Gruteser propose a perturbation algorithm to cross paths of objects (by exchanging their pseudonyms) when they are close to each other [9]. Kido et al. use false dummies to simulate the movement of mobile nodes in order to hide the trace of an actual object [10]. Xu and Cai propose using historical traces of objects to derive a spatio-temporal cloaking region that provides trajectory protection [11].

Bettini et al. first introduced historical k-anonymity and proposed a spatio-temporal generalization algorithm to enforce it [4]. The generalization algorithm enlarges the area and time interval of the request to increase the uncertainty about the real location, while including k common objects. The method fails to account for mobility of the objects, without which the generalized area can easily cross acceptable limits. Chow and Mokbel argue that spatial cloaking algorithms should satisfy the *k-sharing* and *memorization* properties to be robust against session associations [12]. Although the focus of their work is query privacy, the two properties together imply historical k-anonymity. Their algorithm maintains groups of objects based on the two properties, along with query types involved with the objects. However, a query may not be equally significant at all locations occupied by a group's members. Xu and Cai propose an information theoretic measure of anonymity for continuous LBS [13]. They define a *k-anonymity area* as the cloaking region whose entropy is at least k. However, the algorithm is prone to *inversion attacks* where an adversary uses knowledge of the anonymizing algorithm to breach privacy. The most recent of algorithms in this domain is *ProvidentHider* [14]. It uses a maximum perimeter constraint to ensure that cloaking regions are not too large, and the starting set of objects is as big as possible (to take care of leaving objects). This algorithm is later used in our comparative study.

1.3 Contributions

The drawbacks present in the above algorithms point out three issues that must be addressed before historical k-anonymity can be efficiently enforced. Our first contribution in this paper is the identification of these issues, namely *defunct peers*, *diverging trajectories* and *locality of requests*. Our second contribution is an anonymization algorithm, called Continuous ANONymizer (CANON), that implements explicit strategies to resolve each of the three identified issues. In particular, we argue that a cloaking region should be determined using direction information of the objects and show how this restricts the inferences that can be made about the issuer of the request. Large cloaking regions are also avoided by this process. Further, we propose using multiple cloaking regions while issuing a query in order to maintain better service quality.

The remainder of the paper is organized as follows. Section 2 highlights the issues related to historical k-anonymity. Our approach to resolve the issues, the CANON algorithm, is presented in section 3. Section 4 details the experimental setup and results from the comparative study. Finally, section 5 concludes the paper.

2 System Architecture

Figure 1 depicts our system consisting of three layers – (i) *mobile objects*, (ii) a *trusted anonymity server*, and (iii) a *continuous LBS provider*. The trusted anonymity server acts as a channel for any communication between mobile objects

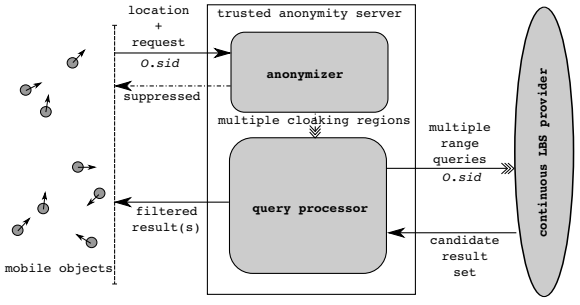

Fig. 1. Schematic of the system architecture

and continuous LBS providers. A mobile object $\mathcal{O}$ initiates a service session by registering itself with the anonymity server. The registration process includes the exchange of current location information ($\mathcal{O}.loc$) and service parameters signifying the request to forward to the LBS provider, as well as the anonymity level ($\mathcal{O}.k$) to enforce while doing so. The anonymity server issues a pseudo-identifier and uses it both as a *session identifier* ($\mathcal{O}.sid$) with the mobile object and as an *object identifier* when communicating with the LBS provider. A set of cloaking regions is then generated for the requesting object and multiple range queries are issued to the LBS provider for these regions. Communication between the anonymity server and the LBS provider is always referenced using the object identifier so that the LBS can maintain service continuity. The candidate results retrieved from the LBS provider are filtered at the anonymity server and then communicated to the mobile object. Subsequent location updates from the mobile object are handled in a similar fashion (with the pre-assigned session identifier) until the anonymity level cannot be satisfied or the service session is terminated. A request is *suppressed* (dropped) when the anonymity requirements can no longer be met within the same service session. A new identifier is then used if the mobile object re-issues the same request. We further assume that an object does not change its service parameters during a session. A separate session is started if a request with different service parameters is to be made. Therefore, an object can have multiple sessions running at the same time, each with a different session identifier.

2.1 Historical k-Anonymity

The primary purpose of a cloaking region is to make a given mobile object $\mathcal{O}$ indistinguishable from a set of other objects. This set of objects, including $\mathcal{O}$, forms the *anonymity set* of $\mathcal{O}$. Objects in the anonymity set shall be referred to as *peers* of $\mathcal{O}$ and denoted by $\mathcal{O}.peers$. A cloaking region for $\mathcal{O}$ is usually characterized by the minimum bounding rectangle (MBR) of the objects in $\mathcal{O}.peers$. Larger anonymity sets provide higher privacy, while at the same time can result in reduced service quality owing to a larger MBR. Therefore, the cloaking region is typically required to achieve an acceptable balance between anonymity and service quality.

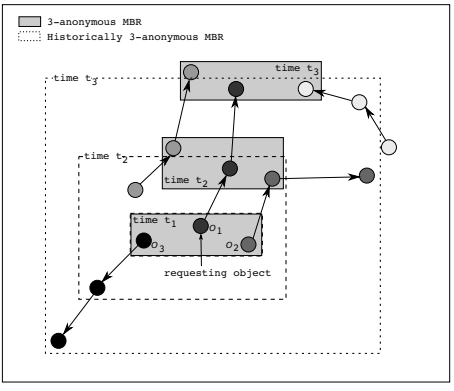

Fig. 2. Conventional k-anonymity and historical k-anonymity

As demonstrated in a number of prior works [5,6,7], achieving reasonable levels of anonymity and service quality is not difficult in the case of a snapshot LBS. However, in our assumed architecture for a continuous LBS system, maintaining the two properties is significantly difficult.

Consider the movement pattern of the objects depicted in figure 2. A 3-anonymous MBR is computed for O_1 during three consecutive location updates. If O_1's requests at the three time instances are mutually independent from each other (as in a snapshot LBS), then the privacy level of O_1 is maintained at 3-anonymity across the different MBRs. However, when the same identifier is associated with all the MBRs (as in a continuous LBS), it only requires an adversary the knowledge of O_1, O_2 and O_3's positions at time t_1, t_2 and t_3 to infer that the requests are being issued by object O_1. This is because O_1 is the only object common across the anonymity sets induced by the cloaking regions. We refer to this as a case of *full disclosure*. Assuming that each object is equally likely to be included in another object's cloaking region, the probability of full disclosure is unacceptably high.

Remark 1: Let $A_1, \ldots, A_n$ be a sequence of anonymity sets corresponding to $n > 1$ consecutive k-anonymous cloaking regions for a mobile object $\mathcal{O}$, generated from a collection of N mobile objects. Then, the probability that the intersection of the anonymity sets $\mathcal{S}_n = \bigcap_i A_i$ has at least p objects, $p > 1$, is $\left(\prod_{i=1}^{p-1} \frac{k-i}{N-i}\right)^n$.

Remark 2: If $k \leq \frac{N+1}{2}$ then the probability of full disclosure is at least $\frac{3}{4}$. The full disclosure risk is given as $\mathcal{D}_{full} = Pr(|\mathcal{S}_n| = 1) = Pr(|\mathcal{S}_n| \geq 1) - Pr(|\mathcal{S}_n| \geq 2)$. Since intersection of the anonymity sets contain at least one object, we have $Pr(|\mathcal{S}_n| \geq 1) = 1$. Hence, $\mathcal{D}_{full} = 1 - (\frac{k-1}{N-1})^n$. With $k \leq \frac{N+1}{2}$, or $\frac{k-1}{N-1} \leq \frac{1}{2}$, we have $D_{full} \geq 1 - \frac{1}{2^n} \geq 1 - \frac{1}{2^2} = \frac{3}{4}$.

We also observe in figure 2 that it does not require knowledge on the objects' locations at all three time instances in order to breach O_1's privacy. In fact, location knowledge at time instances t_1 and t_2 is sufficient to lower O_1's privacy

to 2-anonymity. This is referred to as a *partial disclosure.* Such disclosures occur when the intersection of anonymity sets (corresponding to the same object) contain less than the desired number of peers (the anonymity level k).

A straightforward extension of the conventional k-anonymity model that can counter risks of full and partial disclosures in a continuous LBS is to ensure that all anonymity sets within a service session contain at least k common objects.

Remark 3: **Historical k-anonymity.** Let $A_1, \ldots, A_n$ be a sequence of anonymity sets corresponding to the cloaking regions with the same identifier and at time instants $t_1, \ldots, t_n$, $t_i > t_j$ for $i > j$, respectively. The anonymity set A_i is then said to satisfy historical k-anonymity if $|A_1 \cap \ldots \cap A_i| \geq k$.

In other words, the sequence of anonymity sets preserve historical k-anonymity if all subsequent sets after A_1 contain at least k same objects from A_1. Figure 2 depicts how the cloaking regions should change over time in order to ensure that object O_1 always has historical 3-anonymity.

2.2 Implications

The privacy guarantees of historical k-anonymity in a continuous LBS is similar to that of k-anonymity in a snapshot LBS. In addition, historical k-anonymity also impedes session association attacks by location-aware adversaries. However, maintaining acceptable levels of service can become increasingly difficult in case of historical k-anonymity. We have identified three issues for consideration that impact the practical usage of historical k-anonymity.

1) **Defunct peers:** A defunct peer in an anonymity set is an object that is no longer registered with the anonymity server. As a result, it can no longer be ascertained that a cloaking region includes the peer. If the first cloaking region generated during a particular session contains exactly k objects, then every other anonymity set in that session must contain the same k objects for it to be historically k-anonymous. A defunct peer in this case does not allow subsequent cloaking regions to satisfy historical k-anonymity and introduces possibilities of partial disclosure.

2) **Diverging peer trajectories:** The trajectories of peers influence the size of a cloaking region (satisfying historical k-anonymity) over time. Refer to figure 2. The MBR for object O_1 becomes increasingly larger owing to the trajectory of object O_3. Bigger cloaking regions have a negative impact on service quality. In general, the more divergent the trajectories are, the worse is the effect. Algorithms that use a maximum spatial resolution will not be able to facilitate service continuity as spatial constraints will not be met.

3) **Locality of requests:** The significance of a particular service request can often be correlated with the locality where it originated. For instance, let us assume that the region shown in figure 2 corresponds to a urban locality. Further, object O_1 issues a request to periodically update itself with information (availability, price, etc.) on the nearest parking garage. At time instance t_1, an adversary cannot infer which object (out of O_1, O_2 and O_3) is the actual issuer of the request. However, as O_3 moves away from the urban locality (suspiciously ignoring the high concentration of garages if it were the issuer), an adversary

Procedure 1. CANON(Object $\mathcal{O}$)

Input: Mobile object $\mathcal{O}$ (includes all associated data).
Output: A set of peer groups (one of them includes $\mathcal{O}$); **null** if request is suppressed (cannot satisfy anonymity).

```
1: if (O.sid = null) then
2:    O.peers = CreatePeerSet(O)
3:    O.sid = new session identifier
4: else
5:    remove defunct objects in O.peers
6: end if
7: if (|O.peers| < O.k) then
8:    O.sid = null
9:    return null
10: end if
11: peerGroups = PartitionPeerSet(O)
12: if (∃g ∈peerGroups such that |g| < 2) then
13:    O.sid = null
14:    return null
15: end if
16: return peerGroups
```

can infer that the issuer of the request is more likely to be O_1 or O_2. We say that these two objects are still in the *locality of the request*. If historical k-anonymity is continued to be enforced, O_3 (and most likely O_2 as well) will be positioned in different localities, thereby allowing an adversary infer with high confidence that O_1 is the issuer of the request.

Note that these three issues are primarily applicable in the context of a continuous LBS. Defunct peers is not an issue in snapshot LBS since the set of peers can be decided on a per request basis. Further, since the same peers need not be present in subsequent anonymity sets, their trajectories do not influence the size of the next privacy preserving cloaking region. Differences in locality also do not provide additional inferential power to an adversary. However, in a continuous LBS, these three issues are direct residues of providing privacy by historical k-anonymity.

3 The CANON Algorithm

CANON is an anonymization algorithm that enforces historical k-anonymity for use with a continuous LBS. The algorithm defines explicit procedures to handle each of the three potential issues identified in the previous section. An overview of the algorithm is shown in Procedure 1.

CANON is initiated by the anonymity server whenever it receives a request from a mobile object $\mathcal{O}$. The algorithm starts by first checking if $\mathcal{O}$ has an open session with respect to the current request. If it finds one then the set of peers is updated by removing all defunct peers from the set. Otherwise, a peer set is

generated for $\mathcal{O}$ through the procedure *CreatePeerSet* and a session identifier is assigned. The newly generated (or updated) peer set must have at least $\mathcal{O}.k$ objects in order to continue to the next step; otherwise the request is suppressed and the session is terminated. Historical k-anonymity is ensured at the end of Line 10 since at least k objects inserted into $\mathcal{O}.peers$ by *CreatePeerSet* are still registered with the anonymity server. The next step is to divide the peer set into groups over which the range queries will be issued. A *peer group* is defined as a subset of $\mathcal{O}.peers$. *PartitionPeerSet* divides $\mathcal{O}.peers$ into disjoint peer groups. We shall often use the term "object's peer group" to signify the group that contains $\mathcal{O}$. Each peer group defines a smaller cloaking region than that defined by the entire peer set and reduces the impact of diverging trajectories on service quality. The peer groups returned by CANON are used to issue multiple range queries (one for each) with the same object identifier. Line 12 checks that each peer group contains at least two objects in order to avoid the disclosure of exact location information (of any object) to location-unaware adversaries.

All object agglomerations, namely into peer sets and then into peer groups, are performed so that the *reciprocity* property is satisfied. This property states that the inclusion of any two objects in a peer set (group) is independent of the location of the object for which the peer set (groups) is being formed. Reciprocity prevents inversion attacks where knowledge of the underlying anonymizing algorithm can be used to identify the actual object. The *Hilbert Cloak* algorithm [7] was first proposed in this context for the conventional k-anonymity model. Hilbert Cloak orders the objects according to their Hilbert indices (index on a space filling curve) and then groups them into buckets of size k. The peer set of an object is the bucket that contains the object. The peer set is the same for any object in the same bucket. Further, objects close to each other according to their Hilbert indices also tend to generate smaller (not necessarily optimal) cloaking regions. *CreatePeerSet* and *PartitionPeerSet* thus use Hilbert-sorted lists to incorporate these properties.

3.1 Handling Defunct Peers

As mentioned earlier, defunct peers can influence the lifetime of a service session by reducing the peer set size to below the limit that satisfies historical k-anonymity. The resolution is to include more than k objects in the first peer set. An indirect way to achieve this is to specify a maximum spatial boundary around the requesting object's location and then include all objects within that boundary into the peer set. This is the method used in *ProvidentHider*. However, this approach cannot account for the varying density of objects across time and space. Using spatial boundaries also cannot account for the relative differences in MBR sizes corresponding to varying anonymity requirements. For example, an area of 1 km^2 may be sufficient to have enough peers to satisfy a historical 2-anonymity requirement, but may not be so to satisfy a stronger requirement (say historical 50-anonymity).

A more direct method to resolve the issue is to specify the desired peer set size explicitly. This removes any dependency on how the objects are distributed and

the area required to cover a reasonable number of them. We can specify the size as a sufficiently big constant. However, this strategy favors objects with weaker anonymity requirements as their peer sets are allowed a comparatively higher number of peers to defunct. For instance, a constant peer set size of 20 would allow the anonymizer to tolerate up to 18 defunct peers to preserve historical 2-anonymity, but only 5 defuncts to preserve historical 15-anonymity. Therefore, the strategy adopted in CANON uses an *oversize factor* τ that relatively specifies the number of extra peers that must be included in the peer set. The minimum initial size of the peer set of an object $\mathcal{O}$ is equal to $(1+\tau)\times\mathcal{O}.k$ with this strategy. We say "minimum" because other parameters introduced later can allow more peers to be included. Use of an oversize factor prevents the problem associated with constant peer set sizes. Note that since CANON partitions the peer set into further groups before issuing a query, the area of the cloaking region defined by the enlarged peer set has little or no influence on service quality. However, we would still not want the area to expand extensively in order to curb the issue of request locality.

3.2 Deciding a Peer Set

The *CreatePeerSet* procedure determines the initial peer set for an object. At this point, we need to ensure that majority of the objects in the peer set are in the locality of the request. We believe there are two requirements to address in this regard.

1. Objects in the peer set should define an area where the request is *equally significant* to all the peers.
2. Objects in the peer set should move so that the defined area does not expand *too much*.

The first requirement will prohibit the inclusion of peers that are positioned in a locality where the issued request is unlikely to be made. The second requirement addresses locality of requests in the dynamic scenario where the trajectories of the peers could be such that they are positioned in very different localities over time. Preventing the MBR of the peer set from expanding prohibits peers from being too far away from each other. The first requirement can be fulfilled by choosing peers according to the Hilbert Cloak algorithm. Peers chosen according to Hilbert indices will induce a small MBR, thereby ensuring that they are more likely to be in the same locality. However, a peer set generated by this process cannot guarantee that the second requirement will be fulfilled for long. This is because the neighbors of an object (according to Hilbert index) may be moving in very different directions.

It is clear from the above observation that the direction of travel of the objects should be accounted for when selecting peers. The direction of travel is calculated as a vector from the last known location of the object to its current location, i.e. if $\mathcal{O}.loc_1 = (x_1, y_1)$ and $\mathcal{O}.loc_2 = (x_2, y_2)$ are the previously and currently known positions of $\mathcal{O}$ respectively, then the direction of travel is given as $\mathcal{O}.dir = \mathcal{O}.loc_2 - \mathcal{O}.loc_1 = (x_2 - x_1, y_2 - y_1)$. $\mathcal{O}.dir$ is set to $(0, 1)$ (north) for

Procedure 2. CreatePeerSet(Object $\mathcal{O}$)

Input: Mobile object $\mathcal{O}$ (includes all associated data), and system globals τ, θ and α_{full}.
Output: A set of peer objects (including $\mathcal{O}$).

```
1: L = set of available mobile objects sorted by their Hilbert index
2: k_of = (1 + τ) × O.k; P = φ
3: repeat
4:    L_c = φ
5:    for all (l ∈ L in order) do
6:       if (|L_c| ≥ k_of and AreaMBR(L_c ∪ {l})> α_full) then
7:          break
8:       end if
9:       L_c = L_c ∪ {l}
10:   end for
11:   P_prev = P; f = 1; O_pivot = first object in L_c
12:   repeat
13:      P = (fθ)-neighbors of O_pivot in L_c
14:      f = f + 1
15:   until (|P| ≥ min(k_of, |L_c|))
16:   L = L − P
17: until (O ∈ P)
18: if (|P| < k_of) then
19:   P = P ∪ P_prev
20: else if (|L| < k_of) then
21:   P = P ∪ L
22: end if
23: return P
```

newly registered objects. A *θ-neighborhood* for $\mathcal{O}$ is then defined as the set of all objects whose direction of travel is within an angular distance θ (say in degrees) from $\mathcal{O}.dir$. Therefore, a 0°-neighborhood means objects traveling in the same direction, while a 180°-neighborhood contains all objects. If all peers are chosen within a 0°-neighborhood then it is possible that the area defined by the initial peer set will more or less remain constant over time. However, the initial area itself could be very large due to the non-availability of such peers within a close distance. On the other hand, using a 180°-neighborhood essentially allows all objects to be considered and hence the area can be kept small by including close objects. Of course, the area may increase unwantedly over time. Peer set generation is therefore guided by two system parameters in CANON - the *neighborhood step size* θ and the *full-MBR resolution* α_{full}. The neighborhood step size specifies the resolution at which the θ-neighborhood is incremented to include dissimilar (in terms of travel direction) peers. The full-MBR resolution specifies some area within which the issued request is equally likely to have originated from any of the included objects, thereby making it difficult for an adversary to eliminate peers based on position and request significance. For small values of θ and some α_{full}, all objects in a peer set would ideally move

in a group, in and out of a locality. Procedure 2 outlines the pseudo-code of *CreatePeerSet*. We assume the existence of a function *AreaMBR* that returns the area of the minimum bounding rectangle of a set of objects.

CreatePeerSet first creates a sorted list $\mathcal{L}$ of all registered objects according to their Hilbert indices. It then continues to divide them into buckets (starting from the first one in the sorted list) until the one with $\mathcal{O}$ is found (Lines 3-17). Every time a bucket is formed, $\mathcal{L}$ is updated by removing all objects in the bucket from the list (Line 16). Lines 5-10 determine a set $\mathcal{L}_c$ of candidate objects that can potentially form a bucket. Starting from the first available object in $\mathcal{L}$, we continue to include objects in $\mathcal{L}_c$ as long as the minimum peer set size (denoted by k_{of} and decided by the oversize factor) is not met, or the area of the MBR of included objects is within the full-MBR resolution. Note that, as a result of this condition (Line 6), the minimum required size of the peer set receives more prominence than the resulting area. Hence, the full-MBR resolution is only a guiding parameter and not a constraint. Next, Lines 12-15 select k_{of} objects from the candidate set to form a bucket. The first object in $\mathcal{L}_c$ is chosen as a pivot and all objects in the θ-neighborhood of the pivot are included in the bucket. If the bucket is not full up to its capacity (k_{of}) and more objects are present in $\mathcal{L}_c$, then the neighborhood is increased by the step size θ. By the end of this process, the bucket would either contain k_{of} objects or there are less than k_{of} objects in $\mathcal{L}_c$. The latter is only possible when list $\mathcal{L}$ contains less than k_{of} objects, i.e. the last bucket is being created. Note that object $\mathcal{O}$ is not explicitly used anywhere to decide the buckets, thereby guaranteeing reciprocity. Once the bucket with $\mathcal{O}$ is found, two more checks are required (Lines 18-22). First, if $\mathcal{O}$'s bucket has less than k_{of} objects (possible if it is the last one), then it is merged with the previous bucket. Second, if the number of objects remaining in $\mathcal{L}$ is less than k_{of} (implying $\mathcal{O}$'s bucket is second to last), then the remaining objects are included into $\mathcal{O}$'s bucket to maintain reciprocity.

CreatePeerSet uses θ-neighborhoods and the full-MBR resolution to balance between dissimilar peers and the resulting MBR area. While the step size θ allows incremental selection of dissimilar peers, α_{full} guides the extent of increment admissible to generate a localized peer set. Note that the creation of a peer set is a one time procedure every service session. Hence, a good estimation of the direction of travel is required to avoid diverging trajectories. One possibility is to obtain destination points of objects and generate an average direction of travel. An average direction can also be calculated based on the displacement vector of the object from its starting position. One can also estimate a direction of travel based on a set of last known locations. CANON uses an instantaneous direction vector. We believe this method performs reasonably well in road networks, although the efficacy of other techniques remains to be determined.

3.3 Handling a Large MBR

The full-MBR resolution parameter is used to control breaches related to request localities. Typical values are in the range of 10 to 50 km^2. The parameter is therefore not intended to help generate cloaking regions with small MBRs.

Procedure 3. PartitionPeerSet(Object $\mathcal{O}$)

Input: Mobile object $\mathcal{O}$ (includes all associated data) and system global α_{sub}.
Output: A set of peer groups.

```
1: Sort objects in O.peers by their Hilbert index
2: peerGroups = φ
3: bucket = φ
4: for all (l ∈ O.peers in order) do
5:    if (AreaMBR(bucket∪{l}) ≤ α_sub) then
6:       bucket = bucket∪{l}
7:    else
8:       peerGroups = peerGroups ∪ {bucket}
9:       bucket = {l}
10:   end if
11: end for
12: peerGroups = peerGroups ∪ {bucket}
13: return peerGroups
```

A continuous LBS would require a much finer resolution to deliver any reasonable service. Further, depending on variations in velocity and the underlying road network, some extent of expansion/contraction of the MBR is very likely. The MBR of a peer set is therefore not a good candidate to issue the range queries. Instead, the peer set is partitioned into multiple disjoint groups by *PartitionPeerSet*. Partitioning of the peer set eliminates empty spaces between peers (introduced in the first place if trajectories diverge) and produces smaller MBRs for the range queries [15]. This partitioning can be done either in a way such that each peer group has a minimum number of objects or each peer group has a maximum spatial resolution. The former approach cannot guarantee that the resulting MBR will have an acceptable area. The latter method is adopted in CANON where the maximum spatial resolution of a peer group is specified as the *sub-MBR resolution* α_{sub}. α_{sub} is relatively much smaller than α_{full}. Procedure 3 outlines the partitioning method.

The partitioning is performed in a manner similar to Hilbert Cloak, with the difference that each bucket now induces an area of at most α_{sub} instead of a fixed number of objects. Starting from the first object in the Hilbert-sorted peer set, an object is added to a bucket as long as the sub-MBR resolution is met (Line 6); otherwise the current bucket is a new peer group (Line 8) and the next bucket is created (Line 9). Reciprocity is preserved as before. Note that the pseudo-code in Procedure 3 does not handle the case when a peer group contains only one object. Procedure 1 checks that such groups do not exist (safeguard against location-unaware adversaries); otherwise the request is suppressed. However, the partitioning algorithm itself can relax the sub-MBR resolution when a peer group with a single object is found. One possible modification is to merge any peer group having a single object with the group generated prior to it. Another parameter-less technique is to create partitions that result in the minimum average peer group MBR with the constraint that each group must have at least two objects. We have kept these possibilities open for future exploration.

4 Empirical Study

The experimental evaluation compares the performance of CANON with the *ProvidentHider* algorithm. For every new request, *ProvidentHider* first groups all available objects from a Hilbert-sorted list such that each bucket holds $\mathcal{O}.k$ objects; more if adding them does not violate a maximum perimeter (P_{max}) constraint. The peer set of an object is the bucket that contains the object. A range query is issued over the area covered by the objects in the peer set only if the maximum perimeter constraint is satisfied; otherwise the request is suppressed. Refer to [14] for full details on the algorithm. We measure a number of statistics to evaluate the performance.

- *service continuity:* average number of requests served in a session
- *service failures:* percentage of suppressed requests
- *safeguard against location-unaware adversaries:* average size of the peer group to which the issuing object belongs

4.1 Experimental Setup

We have generated trace data using a simulator [6] that operates multiple mobile objects based on real-world road network information available from the National Mapping Division of the US Geological Survey. We have used an area of approximately $168 km^2$ in the Chamblee region of Georgia, USA for this study. Three road types are identified based on the available data – expressway, arterial and collector. Real traffic volume data is used to determine the number of objects in the different road types [1].

The used traffic volume information (table 1) results in 8,558 objects with 34% on expressways, 8% on arterial roads and 58% on collector roads. The trace data consists of multiple records spanning one hour of simulated time. A record is made up of a time stamp, object number, x and y co-ordinates of object's location, and a status indicator. The status indicator signifies if the object is registered to the anonymity server. An object's status starts off randomly as being active or inactive. The object remains in the status for a time period drawn from a normal distribution with mean 10 minutes and standard deviation 5 minutes. The status is randomly reset at the end of the period and a new time period is assigned. The granularity of the data is maintained such that the Euclidean distance between successive locations of the same object is approximately 100 meters. Each object has an associated k value drawn from the range $[2, 50]$ by using a Zipf distribution favoring higher values and with the exponent 0.6. The trace data is sorted by the time stamp of records.

During evaluation, the first minute of records is used only for initialization. Subsequently, the status of each record is used to determine if the object issues a request. Only an active object is considered for anonymization. If the object was previously inactive or its prior request was suppressed, then it is assumed that a new request has been issued. Otherwise, the object is continuing a service session. The anonymizer is then called to determine the cloaking region(s), if possible.

Table 1. Mean speed, standard deviation and traffic volume on the three road types

road type	traffic volume	mean speed	standard deviation
expressway	2916.6 cars/hr	90 km/hr	20 km/hr
arterial	916.6 cars/hr	60 km/hr	15 km/hr
collector	250 cars/hr	50 km/hr	10 km/hr

The process continues until the object enters an inactive (defunct) state. Over 2,000,000 anonymization requests are generated during a pass of the entire trace data.

Default values of other algorithm parameters are set as follows: $\tau = 0.0$, $\alpha_{full} = 25\,km^2$, $\alpha_{sub} = 1\,km^2$, $\theta = 180°$ and $P_{max} = 5000\,m$. A $5000\,m$ perimeter constraint for *ProvidentHider* is approximately an area of $1.6\,km^2$. Compared to that, α_{sub} has a smaller default value. The precision is around $1000\,m$ (assuming a square area) which serves reasonably well for a Pay-As-You-Drive insurance service. The full-MBR resolution of $25\,km^2$ evaluates to a locality about $\frac{1}{32}^{th}$ the size of New York City. The entire map is assumed to be on a grid of $2^{14} \times 2^{14}$ cells (a cell at every meter) while calculating the Hilbert indices [16]. Objects in the same cell have the same Hilbert index.

4.2 Comparative Performance

Figure 3a shows the average number of requests served in a session for different anonymity requirements. *ProvidentHider* demonstrates poor performance for higher k values, almost to the extent of one request per session. Comparatively, CANON maintains much better service continuity. As mentioned earlier, using a fixed area for varying anonymity requirements makes it difficult for *ProvidentHider* to keep the peer set within the required size. The task is more difficult for bigger peer sets as the algorithm does not consider the issue of diverging trajectories. In fact, more than 50% of the requests are suppressed for $k > 25$ (figure 3b). CANON's performance also seems to fluctuate depending on the oversize factor. In general, a maximum peer set size slightly larger than the minimum required (for example $\tau = 0.25$) gives the best performance, while any further increase degrades it. While a few extra peers is useful to handle defunct peers, having a much larger peer set implies having objects over a larger area and often far away from each other (over time). Therefore, it is possible that some peer groups are formed with a single object owing to the sub-MBR constraint. Requests are then suppressed in the absence of a strategy to handle such peer groups. This is also corroborated by the similar trend in request suppression.

4.3 Impact of Parameters

Each parameter in CANON is intended to address a specific issue with the use of historical k-anonymity. We performed some parametric studies to demonstrate the consequences of varying these parameters. The neighborhood step size is

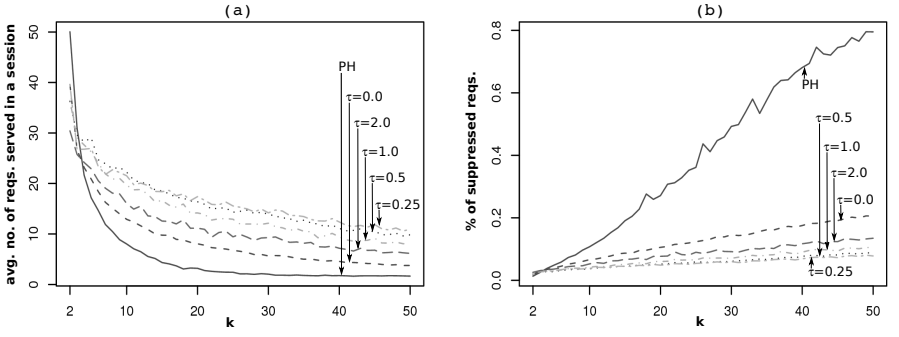

Fig. 3. Comparative performance of CANON with ProvidentHider (PH) for different anonymity requirements (k) and oversize factors (τ). (a) Average number of requests served in a session. (b) Percentage of requests suppressed.

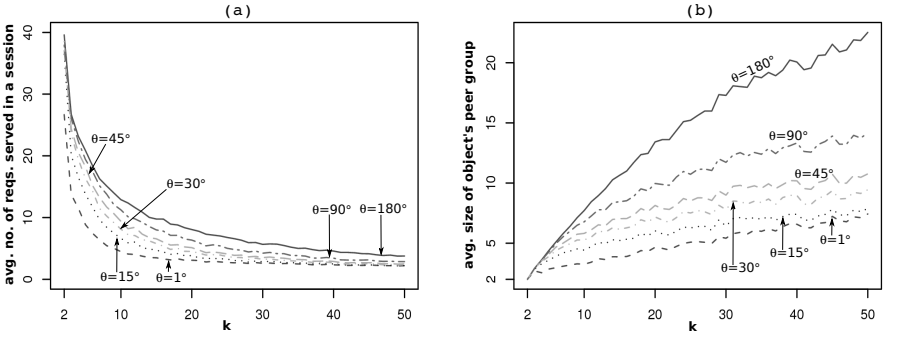

Fig. 4. Impact of different neighborhood step size θ on CANON. (a) Average number of requests served in a session. (b) Average size of requesting object's peer group.

varied between 1° and 180°, and performance is observed for three different settings of the sub-MBR ($\alpha_{sub} = 0.25, 1.0$ and $4\ km^2$) and full-MBR ($\alpha_{full} = 10, 25$ and $50\ km^2$) resolutions. Note that increasing/decreasing the full-MBR resolution will have no impact on peer sets if the required number of objects is always present within a small area. We therefore use a neighborhood step size of 15° while observing the impact of α_{full}. All parameters other than the ones mentioned take their default values.

Neighborhood Step Size θ. Performance in terms of service continuity does not differ a lot for varying step size (figure 4a). Some differences are observed for lower ranges of k $(2-15)$ where larger step sizes show a better performance. Differences are more prominent in terms of peer group size where a bigger neighborhood improves the safeguard against location-unaware adversaries (figure 4b). This behavior is expected since bigger neighborhood sizes allow the inclusion of more dissimilar peers, thereby inducing bigger peer groups due to the possibly

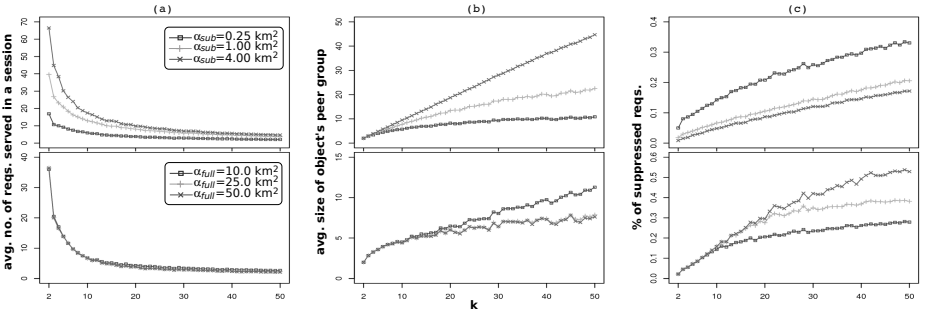

Fig. 5. Impact of spatial resolution parameters on CANON – top: sub-MBR area α_{sub} and bottom: full-MBR area α_{full} with $\theta = 15°$. (a) Average number of requests served in a session. (b) Average size of requesting object's peer group. (c) Percentage of suppressed requests.

close proximity of objects. The statistic of interest is the size of the MBR area defined by the objects in the peer set. We found that this area remains almost constant for the smaller step sizes, specifically for the more frequently requested anonymity levels (higher k), implying that the objects in a peer set move together as a group. Further, the area increases by more than two folds (across different anonymity requirements) when direction of travel is ignored ($\theta = 180°$).

Sub-MBR Resolution α_{sub}. Smaller sub-MBR resolutions mean higher precision in the range queries. However, they also mean higher chances of smaller peer groups, often ones with a single object. With reference to figure 5 (top row), a smaller α_{sub} results in a higher rate of failures, inducing shorter service sessions. Services requiring high location precision will therefore fail to provide longer service continuity. An object's peer group size is also comparatively smaller. Improvements in service continuity is more prominent for weaker anonymity requirements as α_{sub} is increased. However, improvements in peer group size is more noticeable in higher k values. In effect, finding a suitably balanced α_{sub} can help achieve good overall performance. α_{sub} is decided by the service requirements in most cases. Nonetheless, depending on how stringent the requirement is, both privacy (from location-unaware adversaries) and service quality may have scope for improvement.

Full-MBR Resolution α_{full}. The full-MBR resolution is observed to have little or no influence on the average number of requests served in a session (figure 5 bottom row). However, larger areas tend to have higher percentage of failures. A possible explanation is as follows. A larger area with a small step size means similar objects are preferred over the proximity of objects. As a result, a peer set includes objects distributed far apart. This leads to the suppression of requests when the sub-MBR constraint is imposed on the formation of peer groups. Objects far apart cannot be grouped without violating the constraint. This also results in a comparatively smaller peer group size. On the other hand, a smaller

area allows inclusion of close proximity objects at the expense of similarity. The sub-MBR constraint is therefore easier to meet and suppression rate is lower.

4.4 Summary

The following points summarize the results from the experimental study.

- CANON has a superior performance compared to *ProvidentHider* in maintaining longer service sessions across a wide range of anonymity requirements. More requests are also successfully anonymized by CANON.
- Including a small number of extra objects in a peer set is advantageous in handling defunct peers. However, extremely large peer sets can be detrimental.
- Use of direction information during the formation of a peer set does help avoid peers drifting away from each other over time. Choice of a too small neighborhood affects service quality, but is not necessary to balance performance across different measures.
- Performance is better with larger sub-MBR resolutions. However, performance in high precision services may be improved with a good strategy to relax the constraint.
- Service continuity is marginally different for different full-MBR resolutions. However, failure to serve new requests is much lower with smaller resolutions.

5 Conclusions

Owing to the limitations of k-anonymity in a continuous LBS, an extended notion called historical k-anonymity has been recently proposed for privacy preservation in such services. However, all known methods of enforcing historical k-anonymity significantly affects the quality of service. In this paper, we identified the factors that contribute towards deteriorated service quality and suggested resolutions. We proposed the CANON algorithm that delivers reasonably good service quality across different anonymity requirements. The algorithm uses tunable parameters to adjust the size of a peer set, trajectories of peers and cloaking regions over which range queries are issued. Immediate future work includes optimizing the performance of CANON in terms of better usage of directional information. We believe this optimization is crucial in order to have similar performance across all levels of anonymity requirements. Merging location anonymity and query privacy in a continuous LBS is a natural extension of this work.

Acknowledgment

This work was partially supported by the U.S. Air Force Office of Scientific Research under contract FA9550-07-1-0042. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing official policies, either expressed or implied, of the U.S. Air Force or other federal government agencies.

References

1. Gruteser, M., Grunwald, D.: Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking. In: Proceedings of the First International Conference on Mobile Systems, Applications, and Services, pp. 31–42 (2003)
2. Reid, D.: An Algorithm for Tracking Multiple Targets. IEEE Transactions on Automatic Control 24(6), 843–854 (1979)
3. Beresford, A.R., Stajano, F.: Location Privacy in Pervasive Computing. IEEE Security and Privacy 2, 46–55 (2003)
4. Bettini, C., Wang, X.S., Jajodia, S.: Protecting Privacy Against Location-Based Personal Identification. In: Proceedings of the 2nd VLDB Workshop on Secure Data Management, pp. 185–199 (2005)
5. Bamba, B., Liu, L., Pesti, P., Wang, T.: Supporting Anonymous Location Queries in Mobile Environments with Privacy Grid. In: Proceedings of the 17th International World Wide Web Conference, pp. 237–246 (2008)
6. Gedik, B., Liu, L.: Protecting Location Privacy with Personalized k-Anonymity: Architecture and Algorithms. IEEE Transactions on Mobile Computing 7(1), 1–18 (2008)
7. 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), 1719–1733 (2007)
8. Gruteser, M., Liu, X.: Protecting Privacy in Continuous Location-Tracking Applications. IEEE Security and Privacy 2(2), 28–34 (2004)
9. Hoh, B., Gruteser, M.: Protecting Location Privacy Through Path Confusion. In: Proceedings of the First International Conference on Security and Privacy for Emerging Areas in Communication Networks, pp. 194–205 (2005)
10. Kido, H., Yanagisawa, Y., Satoh, T.: An Anonymous Communication Technique Using Dummies for Location-Based Services. In: Proceedings of the IEEE International Conference on Pervasive Services 2005, pp. 88–97 (2005)
11. Xu, T., Cai, Y.: Exploring Historical Location Data for Anonymity Preservation in Location-Based Services. In: IEEE INFOCOM 2008, pp. 1220–1228 (2008)
12. Chow, C.Y., Mokbel, M.: Enabling Private Continuous Queries for Revealed User Locations. In: Papadias, D., Zhang, D., Kollios, G. (eds.) SSTD 2007. LNCS, vol. 4605, pp. 258–275. Springer, Heidelberg (2007)
13. Xu, T., Cai, Y.: Location Anonymity in Continuous Location-Based Services. In: Proceedings of the 15th International Symposium on Advances in Geographic Information Systems, p. 39 (2007)
14. Mascetti, S., Bettini, C., Wang, X.S., Freni, D., Jajodia, S.: ProvidentHider: An Algorithm to Preserve Historical k-Anonymity in LBS. In: Proceedings of the 10th International Conference on Mobile Data Management: Systems, Services and Middleware, pp. 172–181 (2009)
15. Tan, K.W., Lin, Y., Mouratidis, K.: Spatial Cloaking Revisited: Distinguishing Information Leakage from Anonymity. In: Mamoulis, N., Seidl, T., Pedersen, T.B., Torp, K., Assent, I. (eds.) Advances in Spatial and Temporal Databases. LNCS, vol. 5644, pp. 117–134. Springer, Heidelberg (2009)
16. Liu, X., Schrack, G.: Encoding and Decoding the Hilbert Order. Software-Practice and Experience 26(12), 1335–1346 (1996)