ProvidentHider: an Algorithm to Preserve Historical k-Anonymity in LBS

Sergio Mascetti Claudio Bettini
DICo
Università di Milano

X. Sean Wang
Department of CS
University of Vermont

Dario Freni
DICo
Università di Milano

Sushil Jajodia
CSIS
George Mason University

Abstract

One of the privacy threats recognized in the use of LBS is represented by an adversary having information about the presence of individuals in certain locations, and using this information together with an (anonymous) LBS request to re-identify the issuer of the request associating her to the requested service. Several papers have proposed techniques to prevent this, assuming that the use of the service is considered sensitive. In this paper we investigate the more general case in which the adversary is also able to recognize traces of LBS requests by the same anonymous user, so that the identification of the issuer of one request can lead to the disclosure of the same user being in other possibly sensitive locations at different times or using sensitive services. Using the notion of "historical k-anonymity", this paper provides the first formalization of this class of privacy threats. Through extensive experiments based on realistic simulations, and runs of an optimal algorithm, we show some negative results for the defenses based on spatial generalization against these attacks under very conservative assumptions. Under more realistic location knowledge assumptions, we propose two defense algorithms, based on a strategy of changing and reusing of pseudo-identifiers, whose correctness is formally proved. Our experiments show that, among all the proposed algorithms, the ProvidentHider *algorithm is particularly effective in protecting privacy for reasonably long sequences of requests.*

1. Introduction

Location-based services (LBS) are foreseen to become very popular, but studies indicate that many potential users have serious concerns about the involved privacy threats. These threats are due to the potential acquisition of LBS requests by untrusted parties that may use the request data in conjunction with external information to violate the privacy of users. The location of the user and the current time, contained in each LBS request, may play a different role in different types of threats: a) privacy violations may occur when an individual identity is associated with specific location data which is considered sensitive, and b) privacy violations may occur when location data, despite not being sensitive, may be used, jointly with external information, to identify an individual, hence revealing the specific service being requested by that individual, which in this case is the sensitive information. In case a) the identity of the user is either considered explicitly given or possibly derivable, and defense techniques are based on obfuscating the location information (see e.g., [14]). In case b) the defenses are based on anonymization techniques, which do not obfuscate the sensitive information, but usually generalize quasi-identifier values, including location, so that the group of potential issuers has a given minimum cardinality [4], [8], [10], [11].

Most approaches have considered threats a) and b) only in the *snapshot* case, i.e., when the attack is based on the requests issued at the current time. In this paper we consider threats of type b), but in the more realistic situation in which the adversary may understand that a set of requests is issued by the same (albeit anonymous) user. We call this *historical attack*, since it involves the consideration of sequences of requests issued at different times in the past. An adversary may adopt a number of techniques to understand that two or more requests are sent by the same user. For example, if the requests are sent close in time, a form of spatio-temporal reasoning may be effective to associate the requests to the same user [1], [7]. Moreover, for accounting and/or service personalization, a user request may contain a pseudo-identifier (PID), analogous to cookies in traditional Internet services. In this case, if an adversary obtains two requests with the same PID, he can conclude that they have been issued by the same (anonymous) user.

In the historical attack scenario, a new class of threats can be identified. For example, suppose that requests r_1 and r_2 were recognized as being issued by the same anonymous user, and that user u happens to be identified as the issuer of request r_1; then, even if u may not be concerned about the release of parameters in r_1, the adversary is able to conclude that u was in the location contained in r_2 and that it was u to request the specific service in r_2. As first observed in [2], this threat is not avoided by applying to r_1 and r_2, separately, the defense techniques proposed for snapshot attacks.

A solution to historical attacks may be the use of recently proposed privacy information retrieval techniques [5], since by encrypting the location data, the adversary cannot exploit background location knowledge. However, these techniques still need to be extended to LBS based on queries other than k-nn and more importantly there are several concerns

about their efficiency in real environments, especially when moving resources are involved.

The use of spatio-temporal generalization functions to defend from historical attacks has been first proposed in [2], by considering specific patterns of locations as quasi-identifiers, and generalizing the current request's location in order to preserve *historical k-anonymity*. A similar strategy, considering single locations as quasi-identifiers, and progressively applying spatial generalizations has been proposed in [3] and [13]. The work in [3] also aims at preventing the "outlier problem" arising when the generalization function is assumed to be secret while it is actually known by the adversary [10]. Since attacks and defenses are not sufficiently formalized, it is unclear which privacy guarantees are actually provided by the proposed technique. In particular, since a user belonging to an anonymity set, can disappear from the same set in subsequent generalizations of requests by the same issuer, the solution does not seem to provide anonymity under the assumptions considered in our paper. The work in [13] proposes two generalization algorithms. The first one, called *plainKAA* is a simple greedy algorithm following a spatial generalization strategy analogous to the one presented in [2]. However, as opposed to the algorithms we propose in this paper, this defense may fail when the generalization function is known to the adversary. The second one is an optimization of the first, based on the idea that, in the generalization of the requests, the users that were not in the anonymity set of a previous request can contribute to anonymity protection. It is unclear to us if this optimization can really preserve anonymity.

More generally, the main problem with the illustrated related work is the lack of a formal model of the problem which makes it difficult to evaluate the correctness of proposed solutions. This paper provides such a model, and captures a more general class of attacks than the ones previously considered. Indeed, in most related work on anonymity in LBS (case b), it is implicitly or explicitly assumed as a worst case that the adversary may have complete location knowledge, i.e., in order to be safe in the occasional situation in which the adversary knows about the presence of a user in a location, it is assumed he always may acquire from external sources the identities of users in a given location. Clearly, in many scenarios, and for certain types of adversaries, this assumption is excessively conservative. For example, there are locations, as crowded places in a city, public event locations, or busy streets for which it is highly unlikely that all users can be identified together with their precise position.

A more realistic assumption is for the adversary to have partial location information like in the scenario described and considered in a recent paper [12], in which the adversary has background location knowledge about individuals only in certain places (in that case shops where users make purchases with the same identifying card). In order to model this scenario we assume that the overall space can be partitioned into areas where users are considered visible and areas where they are considered not visible. This partitioning can be based on very different considerations, including statistical data and users' input, and may be tuned to be more or less conservative. Note that the very conservative assumption used in most related work is captured in this model as the partitioning considering the whole space as visible.

With respect to considering partial location knowledge, the recent work on trajectory anonymization [12] is the most related; however, that work considers a given historical sequence of locations for each individual and provides a technique based on the suppression to publish the data, preserving the anonymity even in the presence of partial location knowledge. Our algorithm can be seen as doing a similar task but using data generalization instead of suppression, and in an *online* way, i.e., by anonymizing each request (like a point of the trajectory) as soon as it is produced. This is clearly a more difficult task.

The major contributions of this paper are the following:

a) It presents the first formalization of historical attacks and defenses under partial location knowledge;
b) By developing an optimal (yet impractical) algorithm, it provides experimental evidence of the difficulty of privacy protection through spatial generalization under complete location knowledge;
c) It presents two defense algorithms and formally proves their correctness with respect to an adversary having partial location knowledge. Experimental results show that the *ProvidentHider* algorithm, in particular, is effective under partial knowledge for reasonably long sequences of requests.

The rest of the paper is organized as follows. Section 2 formally characterizes privacy attacks. Section 3 illustrate defense techniques, and Section 4 describes our experimental setting and illustrates the results. Section 5 concludes the paper.

2. Formal modeling of historical k-anonymity

In [10], we proposed a formal framework to model LBS privacy attacks and defenses for the snapshot case. The main idea is that the safety of a defense technique can be formally evaluated only if the *context*, i.e., the assumptions about the adversary's external knowledge, is explicitly stated. In this section we first briefly present the formal framework and then characterize the contexts that we will consider in this paper.

As in the majority of related work, our reference scenario includes a location-aware trusted server (LTS) that is aware of the actual locations of all users. The LTS acts as a proxy that filters and generalizes each user request before forwarding it to the service provider (SP).

Each LBS request r is logically divided into three parts: $IDdata$, $STdata$, and $SSdata$, containing user identification data, location and time of the request, and other service parameters, respectively. In the sequel, the spatial and temporal components in $STdata$ are denoted with $Sdata$ and $Tdata$, respectively. The LTS transforms each *original* request r into a *generalized* request r' by replacing $r.IDdata$ with a PID (pseudo-ID) and generalizing the spatial component $r.Sdata$.

To formally define the *generalization* function, we use R to denote the set of all the possible original requests issued by the users to the LTS as well as all the possible generalized requests that the LTS would forward to the SP. We use $issuer(r)$ to denote the user who actually issued the request r. Note that $issuer(r)$ is only known by the LTS, and r does not necessarily contain the identity of $issuer(r)$. We use I to denote the set of all the users. A generalization function can be simply defined as $g : R \rightarrow R$, such that the only difference between r and $g(r)$ is in their $Sdata$ and furthermore $r.Sdata \in g(r).Sdata$, i.e., the spatial region of $g(r)$ contains that of r.

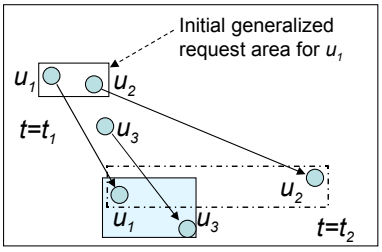


Figure 1. Three users moving around

As an illustrative example, consider the three users moving around in Figure 1. At time t_1, the user u_1 issues a request. The generalization function will first replace u_1's identity with a PID, and then enlarge the location information of the request to be the solid-line box shown in the figure. The reason for this enlargement of the area is to confuse any adversary as which user (u_1 or u_2) actually issued the request, thus achieving so-called 2-anonymity for u_1's request. In this case, u_1's privacy is protected at t_1 by using an "anonymity set" of size 2 for the request. Two questions arise: (1) What is the definition of anonymity set for a generalized request? (2) How much privacy protection does an anonymity set provide?

To answer these two questions, we have to assume the knowledge used by the adversary. Such knowledge is called *context*. Intuitively, when the adversary sees a generalized request r', the anonymity set for the request is the set of users that can possibly issue the request under the specific context C. That is,

$$AS_C(r') = \{i \in I | \text{under } C, \text{ user } i \text{ can possibly issue a request that is generalized to } r'\}$$

Going back to Figure 1, we see that only u_1 and u_2 are possible users to issue the request with the enlarged area (i.e., the upper solid-line box) as we assume that the request location is where the user actually is and is contained in the generalized area. Obviously, $AS_C(r')$ can be formalized only when C is rigorously defined. In this section, we formally define the context we consider in this paper and the associated anonymity set.

The purpose of anonymity set is to render a request *safe* from privacy breaches by fuzzifying as who actually has issued the request. Intuitively, safety is measured by how likely an adversary is able, in a context C, to recover the identity of the issuer from a generalized request. For a given generalized request r' in R, this likelihood is formally defined as a probabilistic distribution $Att_C(r', i)$ over all the individuals $i \in I$. The distribution $Att_C()$ is called an *attack* under context C. An attack Att_C associates with a probability a generalized request r' to the user $i = issuer(r')$. If this probability is beyond (or below, resp.) a given threshold, then we say the request r' is *unsafe* (or *safe*, resp.).

In some contexts, different users in $AS_C(r')$ may have different likelihood to issue r' [10]. (By definition of generalization function, each user outside of $AS_C(r')$ has 0 likelihood to be the issuer.) However, the context we study in this paper does not provide discriminating information to different users in $AS_C(r')$ and hence the size of $AS_C(r')$ determines the degree of protection, with a greater size providing better protection. This brings a special class of attack, called "uniform attacks". Formally, we say that an attack is *uniform* if, for each request r' and each pair of users i, i' in $AS_C(r')$, $Att_C(r', i) = Att_C(r', i')$.

Under uniform attack, the size of the anonymity set determines whether a request is safe or not. Formally, we have,

Definition 1: Given a context C and a positive integer k, a generalized request r' is *k-anonymous* under context C if $|AS_C(r')| \geq k$.

Now given a threshold k and anonymity set $AS_C(r')$, the request r' is safe if and only if $AS_C(r')$ has at least cardinality k.

We now formalize the situation shown in Figure 1. First of all, as motivated in the introduction, we assume the overall area (the area monitored by the LTS or the "world") is partitioned into a visible region A_v and a hidden region A_h. Intuitively, when the user i is in A_v, the LTS will assume that the adversary knows the exact location where she is, while, when she is in A_h, the LTS assumes that the adversary only knows she's in A_h without any more information.[1] Formally speaking, we assume the following function, for each user

1. This model can be extended to further partitioning A_h into smaller areas so that the adversary may know in which small area the user is in but no other information. But this is beyond the scope of this paper.

i, is known to the adversary:

$$loc_i(t) = \begin{cases} \text{exact location of } i & \text{if } i \text{ is in } A_v \text{ at time } t \\ A_h & \text{otherwise} \end{cases}$$

When A_v is the whole area (and A_h is empty), then this degenerates to one that assumes the adversary has the knowledge of the whereabout of every user at all times. In previous papers, we use context C_{st} to refer to the assumption that $A_h = \emptyset$ and the adversary knows the $loc_i(t)$ functions [10], and this context has been explicitly or implicitly assumed in most LBS anonymity research. When $A_h \neq \emptyset$ and the adversary knows the $loc_i(t)$ functions, we will call it context C_{pst}, for partial spatiotemporal context.

The focus of this paper is on privacy defense against request correlation. That is, we assume that the adversary has the ability to deduce with some certainty that a set of requests are issued by the same user. We now formalize this situation. We say that a set of requests are *linked* to a request r', if, in a given context, it is possible for an adversary to understand that all the requests in the set are issued by the same user who issued r'. The ability of the adversary to link requests is modeled through an L function that associates, to each generalized request r', the set of the linked generalized requests, denoted $L(r')$.

In this paper we focus on sequences of requests identified by PIDs, since this case, in addition to being of practical interest by itself, can also be considered as a way to model many possible techniques for correlating requests, like those based on users' spatio-temporal locations [6].

Indeed, we assume the adversary is only able to link each generalized request with the requests issued with the same PID in the past. Formally, the following linking function is given:

$$L_{pid}(r') = \{r'' \in R | r''.IDdata = r'.IDdata\}.$$

When both $loc_i(t)$ and $L_{pid}(r')$ functions are known, the adversary becomes more powerful than when he knows $loc_i(t)$ alone. Consider Figure 1 again. (For simplicity, assume the area in the figure is in the visible region A_v.) When u_1 issues another generalized request r' at time t_2, the shaded box would give the request 2-anonymity if the adversary only knows $loc_i(t)$ function. However, when $L_{pid}(r')$ is also known, this new request is linked to the request u_1 issued at t_1, that is, the adversary can figure out these two requests are issued by the same user. Now by using $loc_i(t_1)$ and $loc_i(t_2)$, the only possible users who may have issued first request are u_1 and u_2, and the only possible users who may have issued the second request are u_1 and u_3. Since the two requests must be from the same user, the adversary can conclude that u_1 is the issuer. Hence, the shaded box does not provide 2-anonymity anymore.

Therefore, considering the history of previous requests, we will need to revise our notion of anonymity set. Indeed, the anonymity set for a generalized request r' should only include the users who could have possibly issued all the requests that r' is linked to. That is, the historical anonymity set of a request r' under context C is:

$AS_C(r') = \{i \in I | \text{under } C, \text{ user } i \text{ can possibly have issued requests that have been generalized to the requests in } L(r')\}$

This is a more general notion than the $AS_C(r')$ previously given, which can be viewed as when $L(r') = \{r'\}$ for all r'. For simplicity, we will abuse the notation and we will mean historical anonymity set whenever we use $AS_C(r')$ unless otherwise specified.

In the above example, u_3 cannot have issued the first request, while u_2 cannot have issued the second request. On the contrary, u_1 has possibly issued both requests and hence the historical anonymity set only contains u_1. One way to fix the problem shown in the above example is to generalize the second request to the lower dashed-line box. In this way, u_2 may still possibly issue the (altered) request and hence the historical anonymity set for the (altered) request has two users in it, namely u_1 and u_2, providing historical 2-anonymity. We will discuss such a strategy in the next section. Another way is through "unlinking", i.e., breaking the possibility for an adversary to track request sequences by the same user. Unlinking methods have been proposed in the literature to contrast specific linking techniques (see, e.g., [1]). As we already observed, the use of PIDs can be perceived as an abstraction to the specific linking techniques; analogously we consider the change of the PID from one request to another from the same user as an abstraction of specific unlinking methods. An important consideration is that unlinking decreases the quality of service because it prevents the SP from providing a personalized service due to PID changes. Therefore, it is desirable for a defense to limit the number of PIDs used for each issuer.

There is one more assumption that is often explicitly or implicitly assumed. That is, it is important to assume that the adversary may know the generalization function g itself [8], [10]. This follows a prevalent practice in security research. As we proved in [10], many defense algorithms proposed in the literature do not provide k-anonymity correctly if g is also assumed. Note that to assume that the adversary knows g is analogous to consider the "reciprocity" problem [8] and prevents the "outlier problem" [10].

To summarize, we have the following.

Definition 2: Context C_H is one in which we assume the adversary knows the following functions: $loc_i(t)$, $L_{PID}(r')$, and $g(r)$.

When constraints are made to the three functions in C_H, we have more restricted contexts. Two of them are:

- Context C_{pst+g} is C_H but assuming $L_{PID}(r') = \{r'\}$ for all r', i.e., no linking is possible.
- Context C_{st+g} is C_{pst+g} but assuming $A_h = \emptyset$, i.e., all locations are visible.

Before formally presenting the historical anonymity set under C_H, we define a function $o()$, such that $o(r', i)$ denotes the original request r (i.e., not generalized) that could be issued by user i from $loc_i(r'.Tdata)$, at the time $r'.Tdata$ and with service specific data $r'.SSdata$, i.e., exactly the same as for r'. In other words, $o(r', i)$ is a request that is exactly the same as r' except that the location of the request is changed to the actual location of i at the request time. The following is immediate:

$$AS_{C_H}(r') = \{i \in I | \forall r'' \in L_{pid}(r') g(o(r'', i)) = r''\}$$

Now historical k-anonymity is defined as follows.

Definition 3: Given a generalized request r' and a positive integer k, if $|AS_{C_H}(r')| \geq k$, we say r' is *historically k-anonymous* under context C_H.

The simplicity of the above definition hides rather complex details. Indeed, the specification of the anonymity set for a given request depends on the generalization function g. In this paper we use generalization functions with a specific property named *segregation*: A segregated generalization function (1) does not generalize (i.e., leave intact) any request that is issued from a hidden location, and (2) computes the generalization of the requests issued from a visible location only considering the locations of the other visible users. The rational for this special family is the following: If a user's location is not known to the adversary (i.e., it is hidden), then we will assume that any adversary cannot tell her apart from other hidden users (at the time), and the generalization of her requests does not usually help to significantly increase the size of the anonymity set. Here, we are tacitly assuming that there are many more hidden users than visible users at any given time, as we believe this is a realistic assumption.

Operation (2) above may appear unintuitive. Why do we only consider other visible users? This is to avoid a possible problem that we call "inversion attack". Indeed, consider the following example. Let r' be a generalized request issued from user u_1 that is issued from a visible position. Assume the generalized region of r' includes the location of only one other user, say u_2, to provide 2-anonymity. Assume u_2 is in a hidden location, and also issues a request. Since this u_2's request is from a hidden location, no generalization is done (due to operation 1 above). When the adversary sees these two requests, with the knowledge of $g()$ function, it is clear that request r' is issued by a user whose location is visible and this user is not u_2. Since no other visible users are in the generalized region of r', r' has to be issued by u_1.

With the above assumptions of the generalization function, we may rewrite our historical anonymity set as follows. Let r' be a generalized request, and g a segregated generalization function. The historical anonymity set of r' in context C_H is:

$$AS_{C_H}(r') = \{i \in I | \forall r'' \in L_{pid}(r') \\ ((r''.Sdata \in A_h \wedge loc_i(r''.Tdata) \in A_h) \vee \\ (r''.Sdata \notin A_h \wedge g(o(r'', i)) = r''))\}.$$

Intuitively, under context C_H, since we assume segregated generalization functions, the anonymity set of a generalized request r' is given in two cases when considering all the past request r'' that are linked to r': (1) if the location of the issuing user was hidden at the time when her request r'' was issued, then everyone in the anonymity set must also be hidden at the same time. (2) if the location of the issuing user was visible at the time when her request r'' was issued, then everyone in the anonymity set must be visible at the same time and everyone could have issued a request that generalizes r'' with generalization function g.

3. Defense algorithms

In this section we describe two generalization algorithms we developed to protect historical k-anonymity. To evaluate the correctness of the proposed solutions, we formally define what we mean by a *defense algorithm*.

Definition 4: An algorithm is a *defense algorithm* in a context C_H if, for each original LBS request and anonymity threshold k provided as input, it returns as output either a request r' that is historically k-anonymous under that context or the value **null**.

An important parameter in the evaluation of the generalization algorithms is the size of the generalized areas. Since an LBS request may become useless if the generalized area is too large, we introduce a new parameter, called P_{max}, that indicates the maximum perimeter that the generalized area can have. The algorithms presented in this section are designed to always generate requests with perimeter not larger than P_{max}. When it is not possible to generalize a request into a historically k-anonymous one with perimeter not larger than P_{max} and the same PID of a previous request, unlinking is performed. If the generalized request has perimeter larger than P_{max}, even after unlinking, then our choice is to suppress the original request. In terms of user interface this may imply the notification to the issuer that the required level of privacy cannot be guaranteed.

3.1. A structured approach

As illustrated in Figure 2, the generalization algorithms we propose are logically structured into procedures such that each one provides protection against a different piece of adversary's knowledge. We say that the procedures that provide protection against the same piece of adversary knowledge belong to the same *level*. The overall idea is that adversary's knowledge can restrict the set of possible issuers of the request. Then, starting from the set I of all users, one

procedure for each level restricts the set of possible users. At the third level the generalized request can be eventually generated.

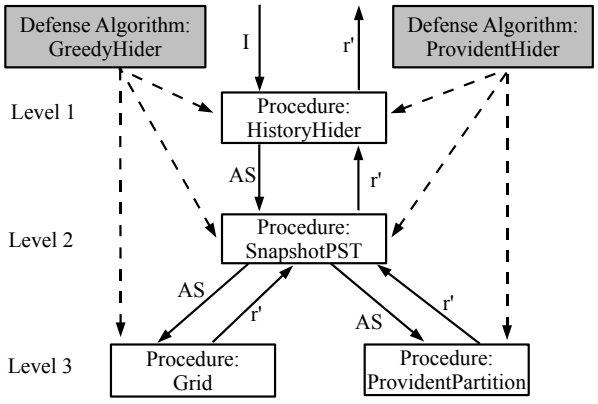


Figure 2. Three-levels logical structure of the generalization algorithm.

At the first level, the problem of request linking is addressed. We propose the procedure *HistoryHider* that provides to the next levels a set of user that are indistinguishable from the issuer i if history of past requests issued by i is considered as well as the context.

At the second level, the problem of partial knowledge is considered. The procedure *SnapshotPST* distinguishes between users that are currently in a visible location from those that are in a hidden location and, if the issuer is in a visible location, provides to the next level the subset of the users received from the previous level that are visible.

The third level addresses the problem of computing the generalization. In this case we propose two solutions: *Grid* and *ProvidentPartition*. The former is derived from an existing defense algorithm in context C_{st+g} [10] while the latter is specifically designed to improve the performance in the historical case in terms of the total number of PIDs used to generalize a set of request, and the average number of requests that a user can issue before a change of the PID is required.

In this paper, we call *GreedyHider* the defense algorithm in context C_H that adopts *HistoryHider*, *SnapshotPST* and *Grid*, while we call *ProvidentHider* the defense algorithm composed by *HistoryHider*, *SnapshotPST* and *ProvidentPartition*.

3.2. Hiding history of past request

Procedure 1 shows the pseudo code for the *HistoryHider* procedure. The input is an original request r that is to be generalized, the set R' of generalized requests already issued by $issuer(r)$, the anonymity parameter k and the value P_{max}. The output is a generalized request r' that is historical k-anonymous and that has perimeter of the generalized region not larger than P_{max} if the procedure can find one, **null** otherwise.

HistoryHider first tries to use a PID that has already been used in previous requests (Lines 2 to 10). To this aim, for each pid used in any of the requests in R', the procedure computes the last, in temporal order, request r_l issued with $IDdata = pid$. Then, variable AS is assigned to store the anonymity set, in context C_H, of r_l. In order to efficiently compute AS, we store an anonymity set associated to each PID. Each time a request r' is issued, we update the value of the anonymity set associated to $r'.IDdata$ with the anonymity set of r'. With this solution, the computation of AS (Line 4) is performed by simply retrieving the anonymity set associated to pid.

Variable AS is then used as a parameter for the second-level procedure called *SnapshotPST* that searches for a possible anonymity set to generalize r among the users in AS. If *SnapshotPST* can compute a safe generalization with perimeter not larger than P_{max}, then it returns a request, otherwise it returns **null**. Hence, if the result r' of *SnapshotPST* is not **null**, then r' is a k-anonymous generalization of r with perimeter not larger than P_{max} and it is returned.

If a result cannot be found for any pid used in the requests in R', then *HistoryHider* creates a new PID. Function *SnapshotPST* is called again using the entire set of user I as the last parameter. Intuitively this means that, since *HistoryHider* is going to unlink, then the knowledge of L_{pid} will not help the adversary to restrict the set of possible issuers and hence the entire set of users can be used. If the result of *SnapshotPST* is different from **null**, then a new PID is assigned to $r'.IDdata$ and the request is returned. Otherwise, the input request is suppressed and the algorithm returns **null**.

The correctness of the procedure is proven by Theorem 1.

Theorem 1: HistoryHider is a defense algorithm in context C_H.

3.3. Distinguishing users in visible and hidden locations

Procedures at level two receive a set I' of users from level one and have to distinguish, in this set, the users that are in a visible location from those that are in a hidden one. This is the main idea of the *SnapshotPST* procedure (Procedure 2). The procedure takes in input an original request r to be generalized, the anonymity parameter k, a value P_{max} and the set I' of users among which the anonymity set of the generalization of r should be computed. The output is analogous to *HistoryHider*.

The "segregated" approach of the algorithms we design is implemented by this procedure. Indeed, *SnapshotPST*

Procedure 1 *HistoryHider*

Input: an original request r, a set R' of generalized requests issued by $issuer(r)$, a positive integer k, a value P_{max}.
Output: **null** or a generalized request r' such that r' is historical k-anonymous and $Perimeter(r'.Sdata) \leq P_{max}$.
Method:

1: $PIDS = \bigcup_{r' \in R'} r'.IDdata$ *// PIDs of prev. reqs*
2: **for all** ($pid \in PIDS$) **do**
3: r_l = the last, in temporal order, request of R' with $IDdata = pid$
4: $AS = AS_{C_H}(r_l)$
5: $r' = SnapshotPST(r, k, P_{max}, AS)$ *// level 2*
6: **if** ($r' \neq$ **null**) **then**
7: $r'.IDdata = pid$
8: **return** r'
9: **end if**
10: **end for**
11: $r' = SnapshotPST(r, k, P_{max}, I)$ *// Unlink*
12: **if** ($r' \neq null$) **then**
13: $r'.IDdata =$ create a new PID
14: **end if**
15: **return** r'

Procedure 2 *SnapshotPST*

Input: an original request r, a positive integer k, a value P_{max}, a set of users I'.
Output: **null** or a generalized request r' that is safe against $Att_{C_{pst+g}}$ with anonymity threshold k and such that $Perimeter(r'.Sdata) \leq P_{max}$.
Method:

1: **if** ($r.Sdata \in A_h$) **then** *// $issuer(r)$ is hidden*
2: $AS = \{i \in I' \text{ s.t. } loc_i(r.Tdata) \in A_h\}$
3: **if** ($|AS| < k$) **then return null**
4: $r' = r$
5: $r'.IDdata =$ **null**
6: **else** *// $issuer(r)$ is visible*
7: $AS = \{i \in I' \text{ s.t. } loc_i(r.Tdata) \in A_v\}$
8: $r' = ProvidentPartition(r, k, P_{max}, AS)$
9: *// in alternative, $r' = Grid(r, k, P_{max}, AS)$*
10: **end if**
11: **return** r'

generalizes the location of the requests issued from visible locations only; If a request is issued from a hidden location, *SnapshotPST* only checks if the number of users in I' that are in a hidden location is smaller than k. If this is the case, the original request is suppressed. Otherwise, the algorithm returns the request without generalizing user location (Lines 2 to 5). On the contrary, if the request is issued from a visible location, variable I'' is assigned to the set of users in I' that are in a visible location. I'' is then used as a parameter of the procedure that operates at the third level (Line 8). The result of the third level procedure is then returned.

Note that, if we assume to be in the conservative case in which every location is considered visible (i.e., $A_h = \emptyset$), the *SnapshotPST* procedure simply calls the third-level procedure passing, as argument, the same set I' received as input.

The correctness of the procedure is proven by Theorem 2.

Theorem 2: Let r be an original request, k a positive integer, P_{max} a value, I' a set of user and $r' = SnapshotPST(r, k, P_{max}, I')$. If $r' \neq$ **null**, then

$$|AS_{C_{pst+g}}(r') \cap I'| \geq k$$

3.4. Computing the spatial generalization

The aim of the procedures at the third level is to compute a spatial region that has perimeter not larger than P_{max} and that includes the locations of at least k users in the set I' received as input. As discussed in Section 1, several algorithms have been proposed in the litterature to solve this problem with the aim of minimizing the size of the generalized region. We modified the *Grid* algorithm we proposed in [10] to serve as a third-level procedure.

The *Grid* algorithm was originally designed to minimize the perimeter of the generalized request, and it has been shown that it achieves good performance with respect to this metric [10]. To obtain this result, the *Grid* algorithm includes in the anonymity set of the generalized request as few users as possible, given that this number is not smaller than k. As a consequence, when the algorithm is used as a procedure in the computation of historical k-anonymity, the first request issued with a given PID from a visible location tends to have a small perimeter and to have few users in its anonymity set. As a consequence, when successive requests are issued, it is unlikely to find, in this anonymity set, a sufficient number of users with whom the new anonymity set can be created. Indeed, the *GreedyHider* algorithm has the problem that, if the PID is not changed, while the area of the first request issued from a visible location is usually small, the area of the successive requests tends to grow quickly, since the users in the anonymity set, which are in close spatial proximity at the time of the first request, may move in different directions or may move to an hidden location. As we shall see in the experimental results, even for large values of P_{max}, with the *GreedyHider* algorithm, only few requests can be issued before the perimeter of the generalized area becomes larger than P_{max}.

In order to address this problem, we developed a different third-level procedure called *ProvidentPartition*. The idea of the partition is that, when a request r is to be generalized, an anonymity set is computed which is larger than the one that would be required to include k users. Indeed, the aim of the algorithm is to keep the anonymity set as large as

possible so that, when successive requests arrive, the users that moved away from the issuer can be removed from the anonymity set. In practice, the algorithm computes a generalized area that includes as many users as possible while having a perimeter smaller than P_{max}.

The computation of *ProvidentPartition*, shown in Procedure 3, is inspired by the *Hilbert Cloak* algorithm [8]. Users are totally ordered according to their locations (at time $r.Tdata$) along the Hilbert space filling curve. Then, they are partitioned into blocks that are constructed considering the users one by one (Lines 5 to 12). The idea is to include in each block as many users as possible until the area of the minimum bounding rectangle that covers the locations of the users in the block is smaller than P_{max}. However, each block should also contain at least k users and hence a user is always added to a block that contains less than k users (Line 5).

The computation described above can lead to the case in which the last block has less than k users. To ensure that each block has at least cardinality k, the algorithm takes the missing users from previous block (Lines 13 to 26). In more details, the blocks are processed one by one from the last one to the first one. At each iteration, the current block B takes, from its predecessor, $k - |B|$ users, i.e., the number of users that are required by B to have cardinality equal to k (Lines 22 to 25). Iteration terminates when the first block is reached or when the current block has more than k users, since all its previous blocks are guaranteed to contain at least k users. If the first block is reached, the first block is deleted and its users are added to the second one.

The correctness of the procedure is proven by Theorem 3.

Theorem 3: Let r be an original request, k a positive integer, P_{max} a value, I' a set of user and $r' = ProvidentPartition(r, k, P_{max}, I')$. If $r' \neq$ **null**, then

$$|AS_{C_{st+g}}(r') \cap I'| \geq k$$

Procedure 3 *ProvidentPartition*

Input: an original request r, a positive integer k, a value P_{max}, a set of users I'.

Output: **null** or a generalized request r' such that r' is k-anonymous in context C_{st+g} and $Perimeter(r'.Sdata) \leq P_{max}$.

Method:

1: **if** $|I'| < k$ **then return null**
2: $H =$ users of I' ordered according to Hilbert index
3: $part = \emptyset$ *// List of blocks*
4: $B = \emptyset$ *// Current bucket*
5: **for all** $(u \in H)$ **do**
6: **if** $(|B| < k$ OR $Perimeter(MBR(B \cup \{u\})) \leq P_{max})$ **then**
7: $B = B \cup \{u\}$
8: **else**
9: $part = part \cup \{B\}$
10: $B = \{u\}$
11: **end if**
12: **end for**
13: $part = part \cup \{B\}$
14: **for all** $(B \in part$ from last to first) **do**
15: **if** $(|B| \geq k)$ **then break**
16: **if** (B is the first block in $part$) **then** joint the first two blocks into a single one
17: **else** move $k - |B|$ users from the block preceding B to B
18: **end for**
19: B is the block of $part$ that contains $issuer(r)$
20: **if** $(Perimeter(MBR(B)) > P_{max}$ **then return null**
21: $r' = r$; $r'.IDdata =$ **null**; $r'.Sdata = MBR(B)$
22: **return** r'

4. Case studies and simulation results

In this section, we present the results of an extensive experimental evaluation of the algorithms presented in Sections 3. The main goal of our algorithms is to guarantee user privacy while providing a high service quality. This means that we want to allow the issuer to use the same PID for as many requests as possible so that the SP can provide a personalized service. Therefore, we want to minimize the number of different PIDs while maintaining the perimeter of each generalized area below the threshold P_{max}.

Tests were performed using simulation data. Users' locations were generated by the Siafu context simulator [9] that was set to simulate movements of $10,000$ individuals. The total area of the map is about 15 km^2. The resulting average density of users per km^2 is about 660. The simulation is designed to reproduce the movements of individuals for 10 consecutive business days; During each day, a user moves to her office in the morning, come back to her house after work hours, and then possibly goes to entertainment places during the evening or in the night. The time in which each movement starts is randomly chosen in a time range. For example, users goes to work between 7am and 9am.

We record users' positions every 10 minutes and generate a request by randomly choosing a time instant at which a user position is recorded. During the generation of the requests, the probability of choosing each time instant is not uniform; The intuition is that it is more likely that a user issues a request during the day than late in the night.

In our tests we consider two *scenarios* about the possible areas where the adversary knows users' locations.

- S1 Under the context C_H but $A_h = \emptyset$, users are visible in every location.
- S2 Under the context C_H with $A_h \neq \emptyset$, a user is in a visible location when she is located in her workplace.

Case S2 is motivated by the fact that, when a user is in

her workplace it is more likely that the association between her identity and her location is known. For example, an employer is likely to know this association. At the same time, a user does not necessarily need to be moving to be interested in LBS. For example, the user may like to be notified by proximity services, even while being at work, about sales or events in nearby shops, restaurants or cultural centers.

Considering that each user works on average 8 hours per day, the adversary is aware of her location 30% of the time. In addition to the assumption about the areas where users are visible, three main parameters affects each test: i) the value of k (default = 20); ii) the number of requests issued by each user in 10 days (default = 30); iii) the P_{max} value (default = $1000m$). In the experiments, the values of P_{max} vary from $200m$ to $1000m$. Note that a perimeter of $200m$ corresponds to a square with area of $2500m^2$, while a perimeter of $1000m$ corresponds to a square with area of $1/16km^2$. We consider a request whose perimeter of the region associated is under $200m$ adequate for services that require very high precision. Nonetheless, for many other services a value of P_{max} of 1000 may not significantly affect the quality of service.

The results presented in this section are obtained as the average computed for 100 users.

The first set of experiments we present is intended to measure how many consecutive requests can be issued by the same user without changing the PID or suppressing a request in scenario S1. In order to evaluate how effective any practical algorithms can be, we designed the *OptimalLength* algorithm that, given a set of original request, computes the generalization, among all the possible ones, that maximizes the number of successive historical k-anonymous requests, that can be issued without changing the PID, without suppressing a request and with a generalized region not larger than P_{max}. *OptimalLength* has two main differences with respect to *GreedyHider* and *ProvidentHider*. First, the input of the algorithm is the complete sequence of requests for a given user, while *GreedyHider* and *ProvidentHider* receive a single request each time. In a sense, *OptimalLength* "knows the future", and therefore it can compute the generalization knowing where the issuer, as well as the other users, will be located in the future. The second difference is that the *OptimalLength* algorithm computes a defense function under the assumption that g is not known to the adversary, as we only require a large intersection of the anonymity sets for the requests in the trace.

In Figure 3(a) we show the results of our tests. It can be noticed that, also for small values of k, on average, using both *ProvidentHider* and *GreedyHider*, only few requests can be issued before unlinking or suppressing is necessary. However, our results shows that this is unavoidable for any practical algorithm. Indeed, also with the *OptimalLength* algorithm, that computes an upper bound for this metric, only about 3.5 consecutive requests can be issued, on average, with $k = 20$.

The second set of experiments measures how many PIDs and suppressed requests are necessary to generalize a set of requests. In this set of tests we do not include *OptimalLength* because it is not designed to reuse the PIDs. In Figure 3(b) we show the impact of P_{max} on this metric. For small values of this parameter ($200m$), in scenario S1, both *ProvidentHider* and *GreedyHider* suppress most of the requests. On the contrary, in scenario S2, *ProvidentHider* can generalize the requests suppressing only few of them (0.2, on average) and using a limited number of PIDs (2.2, on average) while *GreedyHider* still suppresses most of the requests. For larger values of P_{max}, the performances of the two algorithm improves, but *ProvidentHider* is always significantly better than *GreedyHider*. It should be noticed that, even for a $P_{max} = 1000m$, in the scenario S1, using the *ProvidentHider* about 7.5 PIDs are required out of 30 requests. This means that, with our experimental settings, the quality of the service would be strongly affected. On the contrary, in scenario S2, *ProvidentHider* uses about 2 PIDs which means that the algorithm is able to use, in most cases, one PID for requests issued from hidden locations and one for requests issued from visible locations.

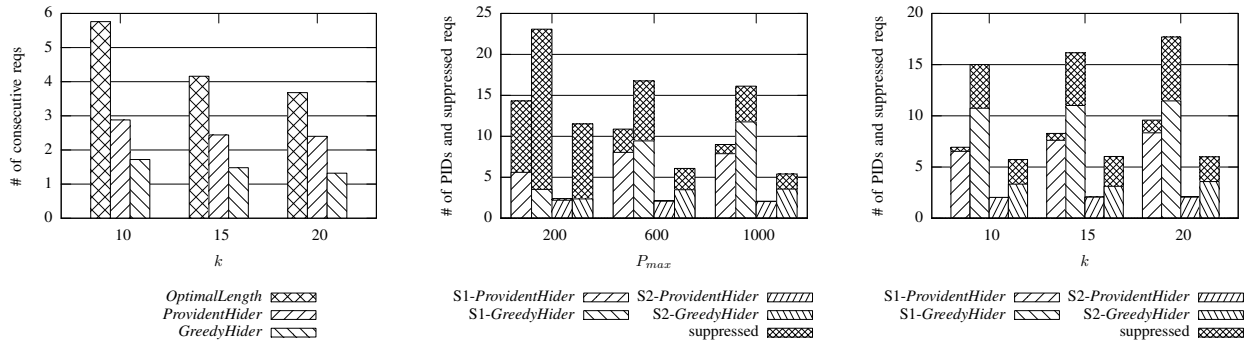
In Figure 3(c) we show the impact of k on the number of PIDs and suppressed requests. Again, we can notice that *ProvidentHider* has better performance with respect to *GreedyHider*. In addition, while the performance of the two algorithm is affected by the value of k in scenario S1, in scenario S2 no significant difference can be observed for different values of k.

5. Conclusions and future works

In this paper, we studied privacy defense algorithms under a realistic assumption that user requests may be correlated. When a prevalent conservative assumption is used in which the adversary knows the locations of all users at all times, this correlation renders privacy very difficult to preserve using spatial generalization techniques without severely degrading the usability. This paper proposed a relaxation of this assumption, formally modeled it and showed via case studies and simulations that a useful defense can be applied in some situations. Future work includes considering also correlation among requests issued by different users, as well as extensions to our framework considering approximate knowledge by the adversary.

Acknowledgments

This work was partially supported by National Science Foundation under grants CT-0716567, CT-0627493, CT-0716575, IIS-0430165 and IIS-0430402, by Air Force Office of Scientific Research under grant FA9550-07-1-0527 and by



(a) Average number of consecutive requests with same PID. scenario S1. 30 requests in 10 days.

(b) Average number of PIDs and suppressed requests. 30 requests in 10 days, $k = 20$.

(c) Average number of PIDs and suppressed requests. 30 requests in 10 days, $P_{max} = 1000m$.

Figure 3. Experimental results.

Italian MIUR under grants PRIN-2007F9437X and InterLink II04C0EC1D.

References

[1] Alastair R. Beresford and Frank Stajano. Mix zones: User privacy in location-aware services. In *Proc. of the 2nd Annual Conference on Pervasive Computing and Communications*. IEEE Computer Society, 2004.

[2] Claudio Bettini, X. Sean Wang, and Sushil Jajodia. Protecting privacy against location-based personal identification. In *Proc. of the 2nd VLDB workshop on Secure Data Management*, volume 3674 of *LNCS*, pages 185–199. Springer, 2005.

[3] Chi-Yin Chow and Mohamed Mokbel. Enabling private continuous queries for revealed user locations. In *Proc. of the 10th International Symposium on Spatial and Temporal Databases*. Springer, 2007.

[4] Bugra Gedik and Ling Liu. Protecting location privacy with personalized k-anonymity: Architecture and algorithms. *IEEE Transactions on Mobile Computing*, 7(1):1–18, 2008.

[5] Gabriel Ghinita, Panos Kalnis, Ali Khoshgozaran, Cyrus Shahabi, and Kian-Lee Tan. Private queries in location based services: Anonymizers are not necessary. In *Proc. of SIGMOD*. ACM Press, 2008.

[6] Marco Gruteser and Baik Hoh. On the anonymity of periodic location samples. In *Security in Pervasive Computing*, volume 3450 of *LNCS*, pages 179–192, 2005.

[7] Baik Hoh, Marco Gruteser, Ryan Herring, Jeff Ban, Daniel Work, Juan Carlos Herrera, Alexandre M. Bayen, Murali Annavaram, and Quinn Jacobson. Virtual trip lines for distributed privacy-preserving traffic monitoring. In *Proc. of the 6th International Conference on Mobile Systems*. ACM, 2008.

[8] Panos Kalnis, Gabriel Ghinita, Kyriakos Mouratidis, and Dimitris Papadias. Preventing location-based identity inference in anonymous spatial queries. *IEEE Transactions on Knowledge and Data Engineering*, 19(12):1719–1733, 2007.

[9] Miquel Martin and Petteri Nurmi. A generic large scale simulator for ubiquitous computing. In *Proc. of the 3rd Conference on Mobile and Ubiquitous Systems: Networks and Services*. IEEE Computer Society, 2006.

[10] Sergio Mascetti, Claudio Bettini, Dario Freni, and X. Sean Wang. Spatial generalization algorithms for LBS privacy preservation. *Journal of Location Based Services*, 2(1), 2008.

[11] Mohamed F. Mokbel, Chi-Yin Chow, and Walid G. Aref. The new Casper: query processing for location services without compromising privacy. In *Proc. of the 32nd International Conference on Very Large Data Bases*, pages 763–774. VLDB Endowment, 2006.

[12] Manolis Terrovitis and Nikos Mamoulis. Privacy preservation in the publication of trajectories. In *Proc. of the 9th International Conference on Mobile Data Management*. IEEE Computer Society, 2008.

[13] Toby Xu and Ying Cai. Location anonymity in continuous location-based services. In *Proc. of ACM International Symposium on Advances in Geographic Information Systems*. ACM Press, 2007.

[14] Man Lung Yiu, Christian S. Jensen, Xuegang Huang, and Hua Lu. Spacetwist: Managing the trade-offs among location privacy, query performance, and query accuracy in mobile services. In *Proc. of the 24th International Conference on Data Engineering*. IEEE Computer Society, 2008.