Università degli Studi di Milano
Dipartimento di Informatica e Comunicazione

Rapporto interno N. 23-07

Preserving Anonymity in Location-based Services When Requests from the Same Issuer May Be Correlated

Sergio Mascetti, Claudio Bettini, X. Sean Wang, Dario Freni, Sushil Jajodia

Preserving Anonymity in Location-based Services When Requests from the Same Issuer May Be Correlated

Sergio Mascetti[*], Claudio Bettini[*], X. Sean Wang[†], Dario Freni[*], Sushil Jajodia[‡]

Abstract

This paper investigates privacy issues in Location-Based Services (LBS) under the assumption that the adversary may be able to understand that a sequence of (anonymous) requests have been issued by the same user. A formal framework is presented to model this kind of privacy attack under two different assumptions in terms of the adversary's knowledge about the users' actual locations. Defense techniques are proposed and evaluated in a realistic simulation environment. The paper provides results and insights on the trade-off between the assumptions of the adversary's knowledge about users' locations and the feasibility of preserving privacy while providing personalized LBS services.

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. It has been shown that, if an LBS request is trivially anonymized by only removing the user identification data, the issuer may be re-identified by using the remaining data in the request in association with some external information.

Previous work has considered re-identification attacks based on associating the issuer's location, contained in the LBS request, with external information on individual user's locations. The following example illustrates this situation.

Example 1 *Alice is affected by a serious disease, for which she frequently needs to purchase a specific medicine. Since the medicine is not always available at every pharmacy when she needs the medicine, Alice uses a "proximity marketing" service (see [1]) to locate a nearby pharmacy that carries the medicine. Alice feels threatened by the fact that someone could understand that she takes that particular medicine, since this information may reveal her disease. For this reason, she uses an anonymizing service that avoids sending explicit identifying information (like full name, SSN, and etc.) with the request, and also hides her device's network address.*

Assume that Alice issues one of these requests from her office. An adversary, who obtains the anonymized request, may understand from the included location information that the anonymous issuer is in the building of a business. The adversary may share this information with the management of the business who knows where each office is located, and has access to other presence data. Hence, the management would come to know that Alice seeks that particular medicine, a violation of Alice's privacy.

Previous work [6, 5, 12, 8, 3] has addressed the problem depicted in Example 1 by proposing different techniques based on a common idea, namely generalizing the location from which the request is issued so that the adversary would not be able to distinguish the issuer among a group of at least k users, even if he knows the location of each of these users at the time when the request is issued.

In this paper we consider a more complex yet realistic situation in which the adversary may be able to understand that a set of requests is issued by the same (albeit anonymous) user. We call this the *dynamic case*, as opposed to the *static case* in Example 1, since it involves the consideration of sequences of requests issued at different times.

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, the adversary can use a form of spatio-temporal reasoning to associate the requests to the same user [2, 7]. Similarly, requests with uncommon service parameters may also raise the probability of being issued by the same user. Finally, for accounting and/or service personalization, a user request may contain 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 rest of the 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.

[*]DICo, University of Milan, Italy
[†]Department of CS, University of Vermont, VT
[‡]CSIS, George Mason University, VA

Example 2 shows that the algorithms devised for the static case do not guarantee privacy in the dynamic case.

Example 2 *Suppose Alice issues a request in the morning for the medicine she needs. To protect her privacy, the location sent in the request is generalized to include the present locations of two other potential issuers, Bob and Carl, so that an adversary cannot distinguish the actual issuer among the three users. In the afternoon, Alice is in a different place and issues a different request, asking for nearby gas stations offering good deals. Generalization is applied again, and a location including two other issuers, David and Tom, is used. If the adversary does not know that the two requests are made by the same user, then he can only attribute each request to Alice with* $1/3$ *probability. However, if he figures out that the two request are issued by the same user, e.g., because the same PID has been included, he can easily intersect the two sets of three users, and uniquely identify Alice as the actual issuer of both requests, resulting in a privacy breach for Alice.*

The privacy threats involved in the dynamic case have been considered before. Beresford et al. [2] propose the use of mix-zones to measure, and possibly avoid, correlation of requests based on spatio-temporal information. Gruteser [7] discusses privacy threats related to correlating requests. Finally, Bettini et al. [4] show that the generalization techniques proposed in the static case are insufficient to provide protection in the dynamic case, as illustrated in Example 2, and propose the notion of historical k-anonymity.

This paper advances the research on the dynamic case by making the following major contributions:

a) It presents the first general formal framework to characterize attacks and defense techniques in the dynamic case, in which the adversary may recognize sequences of requests by the same issuer;

b) It proposes defense algorithms for the dynamic case, under different assumptions about the adversary's knowledge users' locations. The conservative assumption, often implicitly used in the literature, about complete location knowledge is seen as a special case of a more realistic assumption, considering the knowledge as limited to a given set of locations. This by itself is a practically relevant extension even to existing techniques for the static case;

c) It provides experimental evidence of the difficulty of privacy protection under the conservative assumption: a theoretically optimal (yet impractical) algorithm reveals an upper bound to what can be achieved with any practical algorithm. Positive results are achieved for the dynamic case in a realistic simulation under the assumption of partial location knowledge.

The rest of the paper is organized as follows. Section 2 formally characterizes privacy attacks. Section 3 and Section 4 illustrate defense techniques, and Section 5 describes our experimental setting and illustrates the results. Section 6 concludes the paper.

2 A formal framework to model anonymity

In [3], we proposed a formal framework to model LBS privacy attacks and defenses for the static 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.

2.1 Formalization of attacks and defenses

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 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 $issuer()$ 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. We also denote with $loc_i(t)$ the location of user i at time t.

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.

The purpose of a generalization function is to render a request *safe* from privacy breaches by fuzzifying the location information. 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.

Definition 1 *Given an attack Att_C under context C, if $Att_C(r', i) > h$, where h is a threshold value and $i = issuer(r')$, then we say r' is* unsafe *with respect to Att_C and h. Otherwise, we say it is* safe.

In other words, 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.).

Given an attack Att_C and a generalized request r', we indicate with AS_C the function that associates with each generalized request r' the *Anonymity Set of* r', i.e., the set of users that have a nonzero probability of being identified as the issuers of r'. Formally, $AS_C(r')$ is the set of all users $i \in I$ such that $Att_C(r', i) > 0$.

By definition, given a request r', an attack may give a different user in the anonymity set of r' a different probability of being the issuer. A special case consists of those attacks that give all the users in the anonymity set the same probability value. We call them *uniform* attacks.

Definition 2 *An attack Att_C is* uniform *if, for all generalized request r', for each pair of users $i, i' \in AS_C(r')$, $Att_C(r', i) = Att_C(r', i')$.*

In this paper we consider contexts in which the adversary has no a-priori knowledge about the probability of a given user to issue a certain request. For this reason, and because we consider deterministic generalization functions only, all the attacks we model in this paper are uniform. For a description of the non-deterministic generalization functions and the corresponding non-uniform attacks, see [10].

In the remainder of this paper, when describing a specific uniform attack, we will simply identify the corresponding anonymity set, since it completely characterizes the attack. The relationship between a uniform attack and the anonymity set is formalized by the following proposition.

Proposition 1 *Given a uniform attack Att_C under context C, for each generalized request $r' \in R$ and for each $i \in I$,*

$$Att_C(r', i) = \begin{cases} 0 & \text{if } i \notin AS_C(r') \\ \frac{1}{|AS_C(r')|} & \text{otherwise} \end{cases}$$

In the following, we call *generalization algorithm* an algorithm that takes as input, possibly in addition to other data, (i) an original request and (ii) the re-identification threshold h, and returns as output a generalized request or **null**. The **null** value indicates that the input request must be suppressed (i.e., not forwarded to SP). We say that a generalization algorithm is a *defense algorithm* against a given attack Att_C if the execution of the algorithm with any input request r and re-identification threshold h returns either **null** or a request that is safe with respect to Att_C and h.

As mentioned in the introduction, to formally evaluate the correctness of a generalization algorithm, it is necessary to evaluate the safety of the output request which in turn relies on the context assumption. In the remainder of this section, we specify the contexts that we consider in this paper.

2.2 Knowledge of user position (C_{pst}, C_{st})

In this section we model the knowledge that the adversary could have about users location. First, we model context C_{pst} (pst means "partial spatio-temporal knowledge") in which the adversary knows the exact location of the users only when they are in certain regions. In order to model this context, analogous to the proposal in [3], we partition the geographical area monitored by the LTS into two parts: A_v and A_h. While A_v represents the area where users are "visible" and can be identified, A_h represents all the locations where users are "hidden", and hence cannot be identified. More specifically, if a user is in A_v, then we assume that an adversary can know the exact location of the user. In contrast, if a user i is in A_h, we assume that an adversary cannot know i's exact location although he does know that i is in A_h.

In [3] we showed how a uniform attack can be specified in context C_{pst} in the static case which, for the sake of brevity, we omit here. Instead, in Sections 2.3 and 2.4, we specify the attacks under the combination of context C_{pst} with other contexts.

In this paper we also consider context C_{st} (st means "spatio-temporal knowledge") that is explicitly or implicitly assumed in most LBS anonymity research, exemplified by [6, 5]. This context is the special case of context C_{pst} in which A_v corresponds to the entire area monitored by the LTS.

2.3 Public generalization function (C_g)

It is important to assume that the adversary may know the generalization function g itself [8, 3]. This context C_g, follows a prevalent practice in security research. As we proved in [11], many defense algorithms proposed in the literature in context C_{st} are not defense algorithm in context C_{st+g} i.e., the context in which the adversary has, in addition to complete location knowledge, the knowledge of the generalization function used by the LTS. In the following, we provide the definition of the anonymity set in the context C_{pst+g}. For simplicity, instead of providing a general definition that is valid for all generalization functions, we assume a special family of generalization functions that we call *segregated generalization functions*. A segregated generalization function does not generalize any request that is issued from a hidden location, and computes the generalization of the requests issued from a visible location only considering the locations of the other visible users.

For segregated generalization functions, the anonymity set of a request issued from a hidden location is composed of the set of users whose location is hidden. For a request issued from visible locations, the anonymity set for the generalized request r' consists of all the visible users i such that if i were the issuer, then her request would be generalized to the same r'. Note that under C_{st}, all generalization functions are segregated ones as $A_h = \emptyset$.

These anonymity sets can be formally specified through a function $o()$, such that $o(r', i)$ denotes the original request r that is issued by user i from $loc_i(r'.Tdata)$, at the time $r'.Tdata$ and with service specific data $r'.SSdata$. 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 location of i at the request time. The anonymity set under C_{pst+g} can now be defined as

$$AS_{C_{pst+g}}(r') = \begin{cases} \{i \in I | loc_i(r'.Tdata) \in A_h\} \\ \qquad\qquad \text{if } r'.Sdata \in A_h, \\ \{i \in I | g(o(i, r')) = r'\} \\ \qquad\qquad \text{otherwise.} \end{cases}$$

where g is the segregated generalization function.

Kalnis et al. [8] and Bettini et al. [3, 11] studied defense algorithms in context C_{st+g}. The idea is to divide the users into regions without consulting the location of the issuer of the given request r. Divisions should be iteratively executed in such a way that, if a further division results in few than k users in any region, then it is not executed and division terminates. Finally, the region that contains $r.Sdata$ is used as the generalized area. It can be formally shown that the knowledge of this defense function is useless to an adversary. In Sections 3 and 4, when we will define defense functions, we will use a similar idea.

2.4 Linked requests (C_{pid})

The focus of this paper is on privacy defense against correlations that an adversary could perform on the requests issued by a single user. In particular, we consider the case in which the adversary can deduce with some certainty that a set of requests are issued by the same user. 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 under context C is modeled through the L_C function that associates, to each generalized request r', the set of the linked generalized requests, denoted $L_C(r')$.

As motivated in the introduction, in this paper we concentrate on the context C_{pid} in which the adversary is only able to link each generalized request with the requests issued with the same PID. Formally, in context C_{pid}, the following linking function is given: $L_{C_{pid}}(r') = \{r'' \in R' | r''.IDdata = r'.IDdata\}$.

Note that, in context C_{pid} alone, the adversary has no re-identification abilities and hence there are no attacks that he can perform to violate users' anonymity. However, when the knowledge of C_{pid} is used in combination with the knowledge available in other contexts, like C_{st+g} or C_{pst+g}, the attack becomes more powerful, as shown in Example 2.

Unlinking methods have been proposed in the literature to contrast specific linking techniques (see, e.g., [2]). 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 of C_{pid} is that unlinking decreases the quality of service because it prevents the SP from providing a personalized service. Therefore, it is desirable for a generalization function to limit the number of PIDs used for each issuer.

Under context $C_{pst+pid+g}$, since we assume segregated generalization functions, the anonymity set of a generalized request r' is composed of the users i such that (1) i is in A_h when a request r'' in $L_{C_{pid}}(r')$ is issued from A_h, and (2) for each request $r'' \in L_{C_{pid}}$ issued from A_v at time t, $loc_i(t)$ is changed by the generalization function into $r''.Sdata$. Intuitively, if and only if both (1) and (2) are satisfied, i can be the possible issuer of all the requests in $L_{C_{pid}}(r')$ (which includes r'). Formally, we have

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

3 Defenses when user positions are known

In this section we briefly describe some of the generalization algorithms we developed to defend against the attack in context $C_{st+pid+g}$, i.e., the adversary has the complete location information, is able to link requests, and knows the generalization function used by LTS. As we shall see in our experimental results, the algorithms presented in this section require the use of a new PID every few requests and hence, in practice, they do not allow the SP to provide a personalized service. For this reason, and due to space limitations, we do not describe the algorithms in details.

3.1 A greedy algorithm

As we show in Example 2, the generalization algorithms proposed in the literature to provide protection in the static case, i.e., for context C_{st+g}, do not guarantee users' privacy in the dynamic case. However, it is not difficult to extend such an algorithm to the dynamic case. The idea is

to use a static defense algorithm to compute the generalization r' of the first request issued by a user and to store the anonymity set $AS(r')$. Then, for each request r_i linkable to r', generalize the area of r_i with the minimum bounding rectangle (MBR) of the locations of the users in $AS(r')$ at time $r_i.Tdata$. Intuitively, this algorithm maintains a single anonymity set for all the linkable requests, and each user in the anonymity set could be the request issuer. Consider Example 2: Alice's second request would be generalized to the MBR of the current locations of Alice, Bob, and Carl, so any of these three users could be the issuer of both requests.

In our experiments, described in Section 5, we implemented a version of the greedy algorithm that exploits the $grid$ algorithm (see [10]) to compute the generalization of the first request. The algorithm is called *greedyGrid*.

3.2 A provident algorithm

The $greedyGrid$ algorithm has the problem that, while the area of the first request 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. 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. Consequently, in the version of the *greedyGrid* algorithm that we implemented we added a condition: if the perimeter of the generalized area is larger than P_{max}, unlinking is required. This implies that a new PID must be used in the generalized request and that the generalization should be computed as if it was the first one. As we shall see in the experimental results, even for large values of P_{max}, with the $greedyGrid$ 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 algorithm called $ProvidentHilb$ (see [10] for a detailed description of the algorithm). The idea of the algorithm 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 provide defense with threshold h. 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}.

3.3 An optimal algorithm

As we shall see in our experimental results, although $ProvidentHilb$ makes it possible to use fewer PIDs compared to $greedyGrid$, still a change of PID is required in every few requests. In order to evaluate how effective any practical algorithms can be, we designed an *Optimal* algorithm that, given a set of request temporally ordered, computes the generalization of the longest trace of successive requests in the set, starting with the first request, such that the generalized area of each request has a perimeter smaller than P_{max} and the intersection of the anonymity sets for the generalized requests in the trace is no smaller than $1/h$.

Optimal has two main differences with respect to $greedyGrid$ and $ProvidentHilb$. First, the input of the algorithm is all the requests, while $greedyGrid$ and $ProvidentHilb$ receive a single request each time. In a sense, *Optimal* "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 *Optimal* algorithm computes a defense function under C_{st+pid} but not under $C_{st+pid+g}$, as we only require a large intersection of the anonymity sets for the requests in the trace. Indeed, using this algorithm, if the adversary knew the generalization function, he could be able, in some cases, to violate user's privacy.

4 Defense under partial knowledge of user locations

In this section we provide a defense algorithm against attacks in context $C_{pst+pid+g}$. For clarity, we divide the presentation of the algorithm into two parts: in Section 4.1 we describe the $StaticPST$ defense algorithm for the static case while in Section 4.2 we use $StaticPST$ as a procedure in the definition of the $ProvidentHider$ defense algorithm.

4.1 Defense algorithm in the static case

Algorithm 1 shows $StaticPST$, a defense algorithm against the attack in context C_{pst+g}. The algorithm computes the anonymity set among the users specified in the input parameter I'. When the algorithm is used to generalize requests in the static case, this input is always set to I. As we show in Section 4.2, when this $StaticPST$ is used as a step of the generalization in the dynamic case, a subset of I may be used for this parameter.

As a segregated defense algorithm, Algorithm 1 is to generalize the location of the requests issued from visible locations only. Indeed, if a request is issued from a hidden location, the algorithm only checks if the number of users in I' that are in a hidden location at the time of the request is smaller than $1/h$. If this is the case, the algorithm suppresses the request. Otherwise, the algorithm returns the request without generalizing user location (Lines 1 to 4). On the contrary, if the request is issued from a visible location, all the visible users in I' are partitioned using the $HilbPart$ procedure. This procedure partitions the set of users given in input into buckets that contain at least $1/h$ users each

Algorithm 1 $StaticPST$

Input: an original request r, a value $h \in (0, 1]$, a value P_{max}, a set of users I'.
Output: **null** or a generalized request that is safe against $Att_{C_{pst+g}}$ with re-identification threshold h.
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** $(|AS| < 1/h)$ **then return null**
4: $S = r.Sdata$
5: **else** *// $issuer(r)$ is visible*
6: $I'' = \{i \in I' \text{ s.t. } loc_i(r.Tdata) \in A_v\}$
7: $part = HilbPart(I'', h, P_{max}, r.Tdata)$
8: $AS =$ the block of $part$ that contains $issuer(r)$
9: $S =$ MBR of the locations of users in AS at time $r.Tdata$
10: **if** $(Perimeter(S)) > P_{max}$ or $|AS| < 1/h)$ **then**
11: **return null** *// suppress request*
12: **end if**
13: **end if**
14: $r' = r$
15: $r'.IDdata =$ **null**
16: $r'.Sdata = S$
17: **return** r'

(given that the input set has at least $1/h$ users). The algorithm then computes S as the MBR of the locations of the users in the block that contains the issuer. If S has perimeter larger than P_{max}, then the input request is suppressed. Otherwise, the algorithm returns a request that has S as the generalized location. Theorem 1 proves the correctness of the algorithm.

Theorem 1 *$StaticPST$ is a defense algorithm against $Att_{C_{pst+g}}$.*

A detailed description of $HilbPart$ is omitted due to space limitation and can be found in [10]. In short, the goal of this partitioning procedure is to include, in each bucket, as many users as possible, provided that the perimeter of the MBR of the users' locations at the time of the request is below P_{max}. The reason for this is clarified in Section 4.2.

4.2 Defense algorithm in the dynamic case

The main idea of the $ProvidentHider$ algorithm (Algorithm 2) is similar to the idea of the $ProvidentHilb$ algorithm i.e., to include in the anonymity set as many users as possible. The main difference is that $ProvidentHider$ is designed to provide protection in the $C_{pst+pid+g}$ context and hence it can exploit the partial knowledge of the adversary in order to avoid, when possible, the change of PIDs.

In the algorithm, variable I' represents the set of users among which the generalization is to be computed while τ represents the set of requests issued by $issuer(r)$ that is linkable to r. If τ is empty, then it means that the generalization of r is not going to be linkable with any request. In this

Algorithm 2 $ProvidentHider$

Input: an original request r, a value $h \in (0, 1]$, a set τ of generalized requests, a value P_{max}.
Output: **null** or a generalized request that is safe against $Att_{C_{pst+pid+g}}$ with re-identification threshold h.
Method:

1: **if** $(\tau = \emptyset)$ **then** *// r is the first request*
2: $I' = I$
3: $pid =$ create a new PID
4: **else** *// r is not the first request*
5: $r'' =$ the last request in temporal order of τ
6: $I' = AS_{C_{pst+pid+g}}(r'')$ *// anonymity set of r''*
7: $pid = r''.IDdata$
8: **end if**
9: $r' = StaticPST(r, h, P_{max}, I')$
10: **if** $(r' \neq$ **null**$)$ **then**
11: $r'.IDdata = pid$
12: $\tau = \tau \cup \{r'\}$
13: **return** r'
14: **else if** $(\tau \neq \emptyset)$ **then**
15: $\tau = \emptyset$
16: **return** $ProvidentHider(r, h, \tau, P_{max})$
17: **else**
18: **return null** *// suppress the request*
19: **end if**

case, I' is set to I and a new PID is created. Otherwise, I' is assigned to the anonymity set of the last generalized request and the PID used in that request is stored in the variable pid. Then, $StaticPST$ is executed using I' in the input. If $StaticPST$ returns a non-null request r', then the correct pid is assigned to r', the request is added to τ and returned. If $StaticPST$ returns **null**, it means that it is not possible to generalize the input request r. Hence, if r is not the first request (i.e., $\tau \neq \emptyset$), unlink is performed by reassigning τ to $\emptyset$ and recursively calling the $ProvidentHider$ algorithm. Note that this implies that in the second execution of the algorithm a new PID is created and $StaticPST$ is executed with I as the input set of users. Finally, if $StaticPST$ returns **null** and r is the first request then it is not possible to generalize the input request and hence **null** is returned. Theorem 2 proves the correctness of the algorithm.

Theorem 2 *$ProvidentHider$ is a defense algorithm against $Att_{C_{pst+pid+g}}$.*

In the implementation of the $ProvidentHider$ we added two optimization steps. The first one is intended to maintain two PIDs for each user: one PID for the requests issued when the user is visible, the other for the request issued from a hidden location. In practice, when the request is issued from a visible (resp. hidden) location, the variable I' is assigned to the anonymity set of the last request r'' issued from a visible (resp. hidden) location. The second optimization is called "reuse of PID". The idea is that, instead

of considering the last PID only, the algorithm checks if it is possible to use a PID that was used in previous requests, and set τ accordingly, i.e., as the set of requests previously generalized with that particular PID. If the request is issued from a visible (resp. hidden) location, the algorithm tries to use all the PIDs that were used in the generalizations of requests issued from a visible (resp. hidden) location. We applied a similar optimization to the algorithms in the $C_{st+pid+g}$ context to have fair comparison.

5 Experimental results

In this section, we present the results of an extensive experimental evaluation of the algorithms presented in Sections 3 and 4. 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 request 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 artificial data. Users' locations were generated by the Siafu context simulator [9] that was set to simulate movements for $10,000$ 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.

Three main parameters affects each test: i) the P_{max} value; ii) the number of requests issued by each user; iii) the h value (for simplicity, in the following we use the parameter $k = \lceil 1/h \rceil$). The results presented in this section are obtained as the average computed for 100 users.

In Figure 1, we compare performances of *Optimal*, *greedyGrid* and *ProvidentHilb* algorithms in context $C_{st+pid+g}$, considering 30 requests for each user, and $P_{max} = 1000m$, which is the largest value we considered in our tests for this parameter. It can be noticed that, using the *greedyGrid* algorithm, even for small values of k, on average about 11 PIDs are required and 4 requests are suppressed. This means that, on average, about every 3 requests it is necessary to perform an unlink and more than

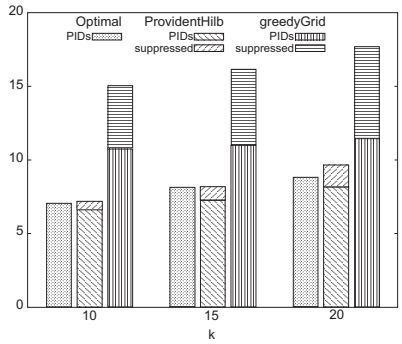


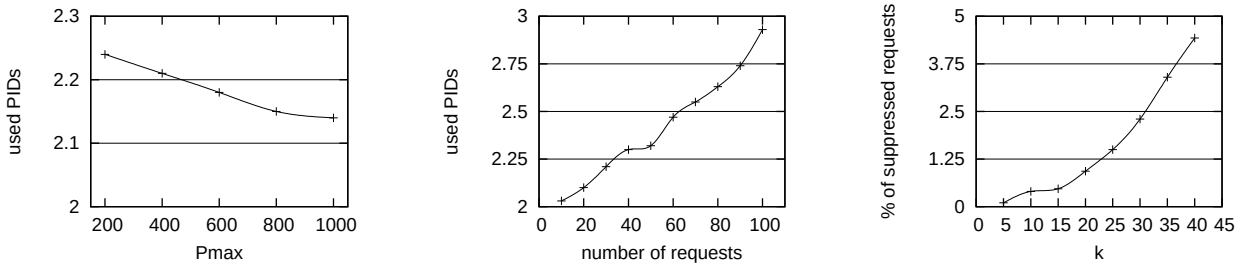
Figure 1. Average number of PIDs and suppressed requests for 30 requests in 10 days with $P_{max} = 1000$, context $C_{st+pid+g}$.

10% of requests are suppressed. In the figure it can also be observed that the performance of *ProvidentHilb* are not significantly different from those of *Optimal*. The main difference is that *ProvidentHilb* suppresses some requests while in our tests no request was suppressed using *Optimal*. This means that it is not possible to devise a defense algorithm against $Att_{C_{st+pid+g}}$ that significantly outperforms, in terms of number of PIDs, the *ProvidentHilb* algorithm. However, even using the *Optimal* algorithm, it is not possible to use, on average, less than one new PID every 5 requests (with $k = 10$).

We now discuss the performance of Algorithm 2 in context $C_{pst+pid+g}$. In our test setting, a user is considered "visible" when she is located in her workplace. Considering that each user works on average 8 hours per day, the adversary is aware of her location 30% of the time.

In Figure 2(a) we show the impact of P_{max} on the total number of PIDs. The values of P_{max} vary from $200m$ to $1000m$, with k fixed to 20 and the total number of requests fixed to 30. 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. We notice that, on average, the algorithm 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. Furthermore, we observed that, for values of P_{max} between $200m$ and $600m$, the percentage of suppressed requests is 0.33%, while for greater values of P_{max} we didn't observed any suppression in all of our tests.

Figure 2(b) shows the average number of used PIDs with respect to the total number of requests issued by each user when $k = 20$ and $P_{max} = 400m$. We can observe that, even if we consider a high frequency of requests (100 re-



(a) Average number of PIDs for 30 requests in 10 days with $k = 20$.

(b) Average number of PIDs in 10 days with k=20 and P_{max}= 400.

(c) Average percentage of suppressed requests with $P_{max} = 400$.

Figure 2. Experimental results under $C_{pst+pid+g}$

quests in 10 days), the average number of PIDs is still smaller than 3. This means that the algorithm scales well with respect to the frequency of the requests.

Finally, we studied the number of suppressed requests for different values of k. Figure 2(c) shows the percentage of suppressed requests for different values of k with P_{max} fixed to $400m$. We can observe that the value of k significantly affects the percentage of suppressed requests.

6 Conclusions and future works

In this paper, we studied privacy defense algorithms under a realistic assumption that user requests may be correlated. This correlation increases the difficulty for privacy preservation, especially when a conservative assumption is used in which the adversary knows the locations of all users at all times. This paper proposed a relaxation of this assumption, formally modeled it and showed via experiments 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.

References

[1] A. Agostini, C. Bettini, N. Cesa-Bianchi, D. Maggiorini, D. Riboni, M. Ruberl, C. Sala, and D. Vitali. Towards Highly Adaptive Services for Mobile Computing. In *Proc. of MOBIS*. Springer, 2004.

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

[3] C. Bettini, S. Mascetti, X. S. Wang, and S. Jajodia. Anonymity in location-based services: towards a general framework. In *Proc. of MDM*. IEEE Computer Society, 2007.

[4] C. Bettini, X. S. Wang, and S. Jajodia. Protecting privacy against location-based personal identification. In *Proc. of SDM*, volume 3674 of *LNCS*. Springer, 2005.

[5] B. Gedik and L. Liu. Location privacy in mobile systems: A personalized anonymization model. In *Proc. of ICDCS*. IEEE Computer Society, 2005.

[6] M. Gruteser and D. Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *Proc. of MobiSys*. The USENIX Association, 2003.

[7] M. Gruteser and X. Liu. Protecting privacy in continuous location-tracking applications. *IEEE Security & Privacy*, 2(2):28–34, 2004.

[8] P. Kalnis, G. Ghinita, K. Mouratidis, and D. Papadias. Preserving anonymity in location based services. TR B6/06, National University of Singapore, 2006.

[9] M. Martin and P. Nurmi. A generic large scale simulator for ubiquitous computing. In *Proc of MobiQuitous*. IEEE Computer Society, 2006.

[10] S. Mascetti. *Privacy Protection through Anonymity in Location-based Services*. PhD thesis, University of Milan, 2007. TR n. 22-07.

[11] S. Mascetti and C. Bettini. A comparison of spatial generalization algorithms for LBS privacy preservation. In *Proc. of PALMS*. IEEE Computer Society, 2007.

[12] Mohamed F. Mokbel, Chi-Yin Chow, and Walid G. Aref. The new casper: query processing for location services without compromising privacy. In *Proc. of VLDB*. VLDB Endowment, 2006.

For the sake of the reviewers only

Proof of Theorem 1

Proof. We need to show that, for each input request r, each P_{max} value, each threshold $h \in (0,1]$, and each set of users I', if $StaticPST(r,h,P_{max},I')$ does not return **null**, then it must return a request r' that is safe with respect to $Att_{C_{pst+g}}$ and threshold h.

Two cases arise, namely (1) $r.Sdata \in A_h$ and (2) $r.Sdata \in A_v$. Consider case (1), and let $AS = \{i \in I' | loc_i(r.Tdata) \in A_h\}$. In this case, if $StaticPST$ does not return **null**, we must have $|AS| \geq 1/h$ and algorithm returns $r' = r$. Since $r'.Sdata = r.Sdata \in A_h$, by definition, $AS_{C_{pst+g}}(r') = AS$. Since $|AS_{C_{pst+g}}(r')| = |AS| \geq 1/h$, by Definition 1 and Proposition 1, r' is safe with respect to $Att_{C_{pst+g}}$ and threshold h.

Consider case (2), and let $I'' = \{i \in I | loc_i(r.Tdata) \in A_v\}$. Since $r.Sdata \notin A_h$, $StaticPST$ does not return **null** only if the block AS in $HilbPart(I'', h, P_{max}, r.Tdata)$ that contains $issuer(r)$ has a cardinality at least $1/h$, i.e., $|AS| \geq 1/h$. (Note that $AS \subseteq I''$ as $HilbPart$ partitions the users in I''.) Under this condition, the algorithm returns the request r' with the parameter $r'.Sdata$ set to the MBR of the locations of all the users in AS. Since $issuer(r)$ is in AS, we have $r.Sdata \in r'.Sdata$. Since $HilbPart$ partitions the users independently from $issuer(r)$, we can see that $StaticPST(o(i',r),h,P_{max},I') = r'$ for each $i' \in AS$. We now show that r' is safe with respect to $Att_{C_{pst+g}}$ and h. Since $r.Sdata \subseteq r'.Sdata$, $r.Sdata \notin A_h$, we have $r'.Sdata \notin A_h$. By definition,

$$AS_{C_{pst+g}}(r') = \{i \in I' | g(o(i,r')) = r'\},$$

where g is $StaticPST$. As we have reasoned above, for each $i' \in AS$, we have the property that $g(o(i',r')) = r'$ (note $o(i',r) = o(i',r')$ as the difference between r and r' is only in their $Sdata$), and hence $AS \subseteq AS_{C_{pst+g}}(r')$. We then have $|AS_{C_{pst+g}}(r')| \geq |AS| \geq 1/h$. By Definition 1 and Proposition 1, r' is safe with respect to $Att_{C_{pst+g}}$ and threshold h. □

Proof of Theorem 2

Proof. We prove by induction on the size of τ that, for each input request r, each value of P_{max} and each $h \in (0,1]$, if $ProvidentHider(r,h,\tau,P_{max})$ does not return **null**, then it returns a request $\bar{r}$ that is safe with respect to $Att_{C_{pst+pid+g}}$ and threshold h.

Induction basis, $\tau = \emptyset$) In this case, $ProvidentHider(r,h,\tau,P_{max})$ computes $r' = StaticPST(r,h,P_{max},I)$. If r' is **null**, the algorithm returns **null**. Otherwise, the algorithm returns $\bar{r}$ which is equal to r' except for $IDdata$ that is set to a new PID. Hence, $L_{C_{pid}}(r') = \{r'\}$. Consequently,

$$AS_{C_{pst+pid+g}}(\bar{r}) = \{i \in I | \\ ((\bar{r}.Sdata \in A_h \wedge loc_i(\bar{r}.Tdata) \in A_h) \vee \\ (\bar{r}.Sdata \notin A_h \wedge g(o(i,\bar{r})) = \bar{r}))\}$$

By Theorem 1, $StaticPST$ is a defense algorithm against $Att_{C_{pst+g}}$, and hence if r' is not-**null**, we have $|AS_{C_{pst+g}}(\bar{r})| \geq 1/h$. It follows, by definition of $AS_{C_{pst+g}}$, that $|\{i \in I \text{ s.t. } loc_i(\bar{r}.Tdata)\}| \geq 1/h$ if $\bar{r}.Sdata \in A_h$, $|\{i \in I \text{ s.t. } g(o(i,\bar{r})) = \bar{r}\}| \geq 1/h$ otherwise. Therefore, $|AS_{C_{pst+pid+g}}(\bar{r})| \geq 1/h$ and hence, by Definition 1 and Proposition 1, $\bar{r}$ is safe with respect to $Att_{C_{pst+pid+g}}$ and threshold h.

Induction step) Assume $\tau \neq \emptyset$ and let r'', the last request in temporal order of τ, be safe with respect to $Att_{C_{pst+pid+g}}$ and threshold h. Two cases arise, the execution of $StaticPST(r,h,P_{max},I')$ either (1) returns **null** or (2) returns a request r'.

Consider case (1). In this case, $ProvidentHider$ performs an unlinking by setting τ to $\emptyset$, and the result of a recursive call to $ProvidentHider$ is returned. We have proved in **induction basis** that in this case if $ProvidentHider$ does not return **null**, then it returns request that is a safe with respect to $Att_{C_{pst+pid+g}}$ and threshold h.

Consider case (2). In this case, $ProvidentHider$ returns the request $\bar{r}$ which is equal to r' except for $IDdata$ that is set to the PID of r''. We now only need to show that $|AS_{C_{pst+pid+g}}(r')| \geq 1/h$. By definition,

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

Since $AS_{C_{pst+pid+g}}(r'') = I'$, we have

$$AS_{C_{pst+pid+g}}(\bar{r}) = I' \cap \{i \in I | \\ ((\bar{r}.Sdata \in A_h \wedge loc_i(\bar{r}.Tdata) \in A_h) \vee \\ (\bar{r}.Sdata \notin A_h \wedge g(o(i,\bar{r})) = \bar{r}))\}$$

that is equivalent to

$$AS_{C_{pst+pid+g}}(\bar{r}) = \{i \in I' | \\ ((\bar{r}.Sdata \in A_h \wedge loc_i(\bar{r}.Tdata) \in A_h) \vee \\ (\bar{r}.Sdata \notin A_h \wedge g(o(i,\bar{r})) = \bar{r}))\}$$

Now if $StaticPST(r,h,P_{max},I')$ does not returns **null**, then either $\bar{r}.Sdata \in A_h$ and $|\{i \in I' \text{ s.t. } loc_i(\bar{r}.Tdata) \in A_h\}| \geq 1/h$ or $\bar{r}.Sdata \notin A_h$ and $|\{i \in I' \text{ s.t.} g(o(i,\bar{r})) = \bar{r})\}| \geq 1/h$. In either case, $|AS_{C_{pst+pid+g}}(\bar{r})| \geq 1/h$. □