Università degli Studi di Milano
Dipartimento di Informatica e Comunicazione

Rapporto interno N. 22-07

Privacy Protection through Anonymity in Location-based Services

Sergio Mascetti

Contents

Chapter 1

Introduction

1.1 Problem description

Location-based services (LBS) refer to those information services that deliver differentiated information based on the location from where a user issues the request. This kind of services has recently attracted much interest from both industry and research. Currently, the most popular commercial service is probably car navigation, but many other services are being offered and more are being experimented, as less expensive location aware devices are reaching the market.

Consciously or unconsciously, many users are ready to give up one more piece of their private information in order to access the new services. Many other users, however, are concerned with the privacy problem that could arise. Indeed, in LBSs the user location information necessarily appear in a request sent to the service providers. A privacy problem arises when the user is concerned with the possibility that an attacker may connect the user's identity with the information contained in the service requests, including location and other information. In general, the association between the real identity of the user issuing an LBS request and the request itself as it reaches the service provider can be considered a privacy threat.

An obvious defense against privacy threats is to eliminate from the re-

quest any data that can directly reveal the issuer's identity, possibly using a pseudonym whenever this is required (e.g., for billing through a third party). Unfortunately, simply dropping the issuer's personal identification data may not be sufficient to anonymize the request. For example, the location and time information in the request may be used, with the help of external knowledge, to restrict the possible user to a small group of issuers. A similar problem is well-known for the release of data in databases tables [44]. In this case, the problem is to protect the association between the identity of an individual and a tuple containing her sensitive data.

A possible solution consists in decreasing the precision of the location information sent to the service provider. In practice, the issuer of the request does not communicate her exact location but, instead, she provides a generalized area that includes the location where she is actually located.

Example 1 *Alice submits an LBS request from her device asking for a nearby vegetarian restaurant. She would rather prefer not to reveal that she is a vegetarian. For this purpose her credentials in the request to the service provider are substituted with a pseudo-ID and the network address of her device is also appropriately masked.*

Suppose the anonymized *request is obtained by an attacker. If this attacker happens to know that Alice was the only person to be present at the location and time indicated in the request (possibly through sighting, cameras, presence data in company buildings or the alike), the request is associated to her identity and Alice's private information is disclosed.*

In order to prevent this privacy violation, Alice avoids sending her exact location and, instead, she provides the service provider with a generalized area *that includes her location and the location of other* $k-1$ *users. The service provider replies to this* generalized request *with the set of vegetarian restaurants that are the closest one to any point of the generalized area.*

If the generalized area is sufficiently small, the service result will probably consist of a small set of restaurants among which there is the one that would have been returned if the exact location were provided. The advantage of

using the generalized area is that, since it contains at least k users, even if the attacker can obtain Alice's request and even if he knows the location and the identity of each of the k users, he is not able to identify the issuer with probability greater than $1/k$ *and therefore he is not able to understand that Alice is vegetarian (with probability greater than* $1/k$*).*

The generalization technique used in Example 1 implicitly assumes that the information about the spatio-temporal position of a sufficient number of potential users of the service is available to the entity performing the generalization. The typical scenario assumes the existence of a *Location-aware Trusted Server* (LTS) that can gather this information; the LTS receives the LBS requests from the users, it performs the appropriate generalization (also hiding explicitly identifying values), and it forwards the generalized request to the target service provider. The answer from the SP is also routed through the LTS to be redirected to the specific user with a refined result when possible [29].

In the above scenario, the generalization algorithm employed by the LTS has three goals: (i) to guarantee the user's privacy by insuring that a sufficiently large number of potential users are not distinguishable from the issuer, (ii) to preserve the quality of service by minimizing the size of the generalized area, and (iii) to be efficient, since it must be computed on-line.

To illustrate the above three goals, consider a yellow pages LBS. If a user's request is not generalized, the SP will be able to provide a very accurate service pointing to the closest relevant resource considering the exact position of the user; However, by revealing that information, the user's privacy may be compromised. On the other hand, if the user's position is generalized to a very large area, many more resources would be selected as potentially relevant to the user; This could lead to a waste of computation time and bandwidth, as well as to the possible delivery of useless information to the user.

1.2 Contribution

Most of the generalization algorithms proposed in previous works are not formally proved to be correct (see Section 6 for related work). This is due to the lack of a formal framework to define when a generalized request is safe. In this dissertation **we provide a unifying framework** that makes it possible to define the safety of a request and, consequently, to prove the correctness of a generalization algorithm. The framework clarifies the role of the external knowledge available to the attacker as well as of his reasoning abilities. Indeed, we note that the fact that certain generalization algorithms previously proposed in the literature are considered "unsafe" is simply due to different assumptions on the attacker's knowledge and reasoning abilities. We call "context" a set of these assumptions. The formalization we provide makes it possible to formally prove the safety of generalization algorithms under different contexts.

Based on the formal framework, **we propose a new methodology** to design safe generalization algorithms. The overall idea is that the context should first be defined, then it is possible to specify the attack that can be performed in that context and hence to state when a request is safe or not. Finally, a generalization algorithm should be defined to compute only requests that are safe in the chosen context.

In this dissertation we also **classify existing generalization algorithms** with respect to the context in which each of them is safe and we **propose new generalization algorithms** in contexts considered in previous works as well as in new contexts. We implemented most of the algorithms proposed in the literature as well as the new algorithms in order to **perform an extensive experimental study** on the performance of the algorithms in terms of computation time and size of the generalized area.

Results from this dissertation appear in the following publications: [7, 8, 37, 6]. Whilst much of this thesis is joint work with my supervisors, I made significant contributions to Chapter 2, in particular in the definition of the formal framework, and in Chapters 3 and 4, in particular in the specification

of the attacks and in the design of the generalization algorithms.

1.3 Outline

In Chapter 2 we discuss the privacy threads in LBS and we present the formal framework that will be used throughout this dissertation. In Chapter 3 we model the two contexts that were implicitly considered in most of previous works and a new context that, to the best of our knowledge, had never been considered in previous works. In each context, the attacks are specified, the existing generalization algorithms are analyzed and new ones are proposed. We conclude the chapter showing the results of our experimental results. The three contexes considered in Chapter 3 are extended in Chapter 4 where we specify new attacks, we present the corresponding generalization algorithms and show the results of our experimental results. A discussion is presented in Chapter 5 while related work are reported in Chapter 6. In Chapter 7 we conclude the dissertation discussing the open problems and summarizing the contributions of this work.

Chapter 2

A model of privacy in location based services

2.1 Identification of privacy threats

In general, there is a privacy threat when an attacker is able to associate the identity of a user to information that the user considers private. In the case of LBS, this *sensitive association* can be possibly derived from requests issued to service providers. More precisely, the identity and the private information of a single user can be derived from requests issued by a group of users. Figure 2.1 shows a graphical representation of this general view of privacy threats in LBS.

In order to infer the sensitive association, the attacker can exploit some *external knowledge* that is not transmitted with the requests. This information can be used, for example, to discover the identity of the issuer even if this information is not explicitly provided in the request or to derive private information associated with a particular location.

The assumption about the external knowledge that is available to the attacker strongly affects the defense techniques used to protect user's privacy. More generally, a privacy preserving technique can be provided once a *context* is fixed that includes the assumptions about the external knowl-

edge that is possibly available to the attacker and the assumptions about his reasoning abilities.

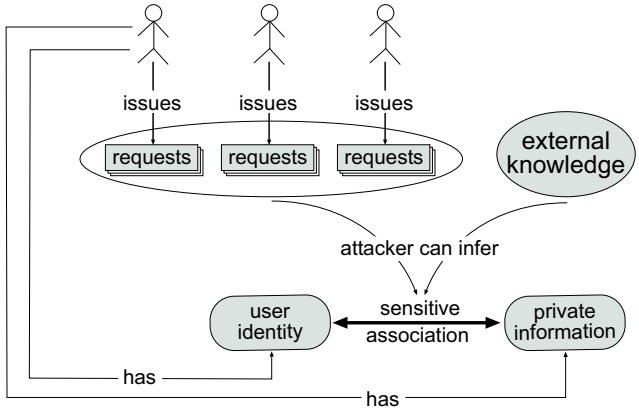


Figure 2.1: General privacy threat in LBS

2.1.1 The static case

Most of the approaches presented so far in the literature [24, 39, 29, 23, 8, 37] have proposed techniques to ensure a user's privacy in the case in which the attacker can acquire a single request issued by that user. More specifically, these approaches assume that:

- the attacker is not able to *link* a set of requests i.e., to understand that the requests have been issued by the same (anonymous) user;
- the attacker is not able to derive private information about the issuer of a request from the requests issued by other users.

In general, we can distinguish privacy threats according to two orthogonal dimensions: a) threats in *static* versus *dynamic* cases, b) threats involving requests from a single user (*single-issuer* case) versus threats involving requests from different users (*multiple-issuer* case).

Figure 2.2 shows a graphical representation of the privacy threat in the static, single-issuer case. In this case, in order to prevent the disclosure

of the sensible association, it is sufficient to prevent the attacker from inferring either user's identity or user's sensitive information. The ongoing research in this field is tackling these two subproblems: prevent the attacker from inferring the user's identity and prevent the attacker from inferring the user's private information. Despite the solution of one of the two subproblems is sufficient to guarantee user's privacy, we argue that the solution of both subproblems could enhance better techniques for privacy protection. Indeed, the obfuscation of requests parameters usually involved in privacy protection techniques implies a degradation of the quality of service. A location based privacy preserving system that implements solutions for both the subproblems can combine them in order to optimize quality of service while preserving privacy.

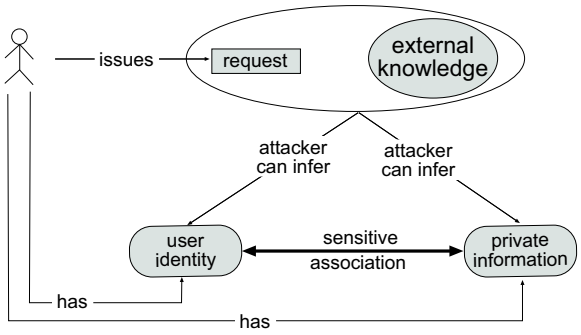


Figure 2.2: The static, single-issuer case

Example 2 shows that, in the multiple-issuer case, an attacker can infer the sensitive association for a user even if the identity of that user is not revealed to the attacker.

Example 2 *Suppose a user u issues a request that is generalized into r'. Assume that, considering r', an attacker can only understand that the issuer of r' is one of the users in the set S of potential issuers. However, if all of the users in S issue requests from which the attacker can infer the same sensitive information inferred from r', then the attacker can associate that sensitive information to u.*

In the area of privacy in databases, this kind of attack is known as *homogeneity attack* [35]. The problem in the area of LBS is depicted in Figure 2.3. Note that, differently from the general case (Figure 2.1), in the static, multiple-issuer case, a single request for each user is considered.

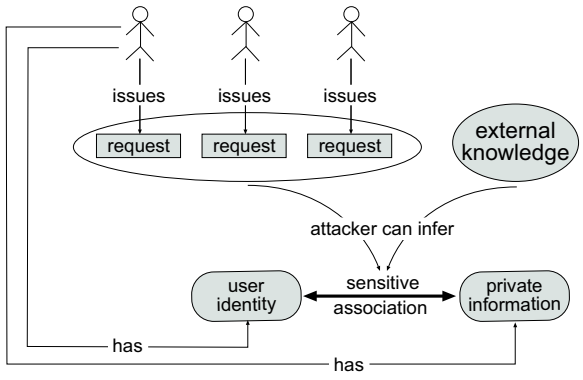


Figure 2.3: The static, multiple-issuer case

2.1.2 The dynamic case

In contrast with the static case, in the dynamic case it is assumed that the attacker is able to recognize that a set of requests has been issued by the same user. Researchers [5, 27, 13] have considered such a possibility. We call this *linking*. Several techniques exist to *link* different requests to the same user, with the most trivial ones being the observation of the same pseudo-id in the requests. We call *request trace* a set of requests that the attacker can correctly associate to a single user.

Figure 2.4 shows a graphical representation of the dynamic case. The corresponding techniques to preserve privacy are facing two problems. First, preventing the attacker from linking the requests (called *linking problem*); Indeed, the longer is a trace, the higher the probability of the issuer to loose her privacy. Second, preventing the attacker from understanding the sensitive association from a request trace.

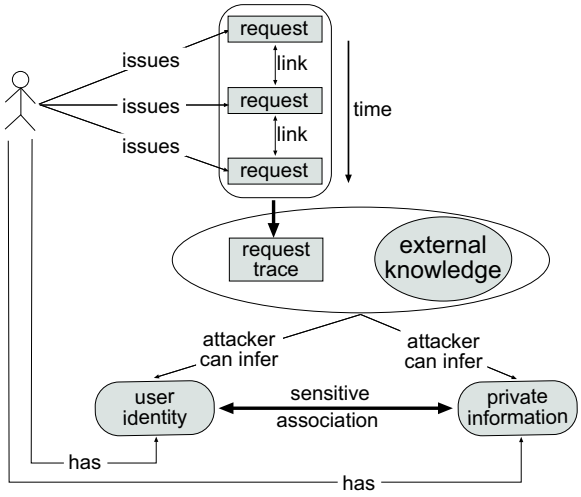


Figure 2.4: The dynamic case

2.1.3 The reference scenario

Figure 2.5 shows our reference scenario that involves three entities:

- The **User** invokes or subscribes to location-based remote services that are going to be provided to her mobile device.
- The **Location-aware Trusted Server (LTS)** stores precise location data of all its users, using data directly provided by users' devices and/or acquired from the infrastructure. It also has the ability to efficiently perform spatio-temporal queries to determine, for example, which or how many users are in a certain region.
- The **Service Provider (SP)** fulfills user requests and communicates with the user through the LTS. Both pull and push communication service models are possible; We concentrate on the former but the framework we present can be easily extended to deal with the latter too.

In our model each request is processed by the LTS into a request with the same logical components but appropriately generalized. Under the condition that user's privacy is guaranteed, this generalization should be as little as

possible to ensure the best service quality for the user. Requests, once forwarded by the LTS, may be acquired by potential attackers in different ways: they may be stolen from SP storage, voluntarily published by the trusted parties, or may be acquired by eavesdropping on the communication lines. On the contrary, the communication between the user and the LTS is considered as trusted, and the data stored at the LTS is not considered accessible by the attacker.

Most of the approaches proposed in the literature [22, 24, 29, 39, 8, 37] to protect LBS privacy consider scenarios that can be easily mapped to the one depicted in Figure 2.5. Actually, scenarios where no location-aware intermediate entity is present have also been considered. For example, in [30] a direct communication between the user and the service provider is assumed, and the defense function is computed on the client system. However, in this model it is not possible to assume that the client has any awareness of the exact location of other clients; hence the generalization techniques proposed in this and in other papers would not be applicable. On the contrary, Ghinita et al. propose a peer to peer architecture in which no centralized entity is required [23]. In this case, the exact location information, required to compute the generalized request, is shared by each user with other users in the network. The problem with this approach is that, in general, the other peers of the network are not trusted and hence providing them with the exact location information may compromise user's privacy.

We believe that the current business models of mobile operators naturally support the existence and functionality of an entity like the LTS. Indeed, mobile users implicitly trust the operator infrastructure even if they know that very accurate information about their location and service requests is stored. Moreover, in most countries each operator has a very large number of customers, and hence a collection of data that may be more than sufficient to implement some of the defense techniques we are proposing.

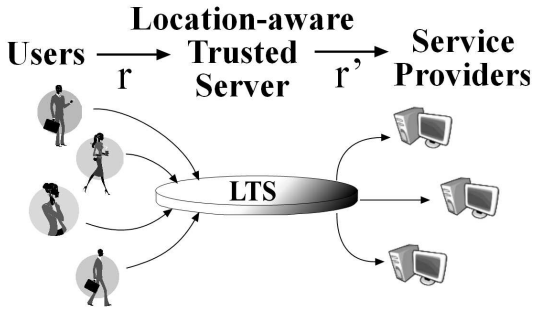


Figure 2.5: A general reference scenario

2.2 Definition of a formal framework for privacy preservation

2.2.1 Privacy protection through anonymity

As depicted in Figure 2.1, a privacy threat occurs when an attacker is able to obtain a user's sensitive association. When a LBS requires each request to contain explicit full identification of the user, the sensitive association can only be protected by avoiding the explicit and implicit release of the second component of the association: *private information*. However, most LBS either do not require full identification or they admit the use of pseudonyms for billing and/or personalization. In these cases, preserving the anonymity of the issuer is a successful technique to avoid releasing a sensitive association, while still providing precise service invocation parameters.

Note that the anonymity problem in LBS has at least two distinguishing aspects with respect to the analogous problem in the release of data from databases [44]. First, the fact that each request contains data about the location of the user at the time of request, introduces spatio-temporal data as a new kind of potential *quasi-identifier*, and it is well known that the effective management of this kind of data requires specific techniques. Second, anonymity in databases has been mainly studied considering a one-time publication of a given set of records, while the problem in LBS is inherently dynamic: the position of users is continuously changing and this has to be

taken into account each time a request has to be anonymized. Moreover, inferencing based on previously anonymized requests can be used by the attacker.

Anonymity as a LBS privacy protection technique has been only recently investigated. Several research contributions (among which [24, 39, 29, 8]) have proposed techniques that aim at enforcing the issuer of a request to be *anonymous*, in the sense that an attacker, that can acquire the requests, must not be able to associate each request to its issuer with likelihood greater than a threshold value. Unfortunately, most of these papers lack of a clear understanding of which techniques can be proved to be *safe* under which conditions. To solve this problem, a formal framework was proposed in [8]. In this dissertation we adopt this framework and we describe it in details in the remaining of this chapter.

2.2.2 Formal definition of requests

In this section we formally characterize the *original* requests that the user issues to the LTS as well as the *generalized* requests that the LTS forwards to the SP.

We denote with R the set containing all the possible original requests and with R' the set that contains the possible generalized requests that the LTS would forward to the SP. We also assume that R' contains the special element r_{null} that represents a request that should not be forwarded to the SP. If not differently stated, we indicate with r an original request of R and with r' a generalized request in R'.

The logical format of an original or generalized request is represented by the following triple:

$$\langle IDdata, STdata, SSdata \rangle$$

- **IDdata**: in an original request is the exact user identity; in a generalized request it is either empty or contains a pseudo-id.
- **STdata**: contains spatial-temporal information about the location of

the user performing the requests, and the time the request was issued.

- **SSdata**: the *service specific* data, that are the parameters characterizing the request service and the service provider

Given an original or generalized request r, we indicate with $r.IDdata$ ($STdata$, $SSdata$ respectively) the attribute $IDdata$ ($STdata$, $SSdata$ respectively) of r.

Since we are particularly interested in the spatio-temporal attribute of a request, we provide more details about the $STdata$ field of a request. First of all, for the sake of simplifying notation, we use **Sdata** and **Tdata** to denote the spatial and temporal part of $STdata$, respectively. Given an original request r, $r.Tdata$ is the time r was issued while $r.Sdata$ is the exact location of the issuer of r at time instant $r.Tdata$. In this dissertation, we assume that $r.Sdata$ is a point in the 2-dimensional space.

Despite some generalization techniques proposed in the literature (e.g., [24, 22]) makes it possible to generalize also along the temporal dimension, in this dissertation we consider the generalization along the spatial dimension only. As a consequence, given a generalized request r', $r'.Sdata$ is a region in the 2-dimensional space and $r'.Tdata$ is a time instant. We also assume that $r'.Sdata$ is a rectangle with axis parallel edges. The resons of this assumption are discussed in Section 5.2.

In the rest of this dissertation we indicate with I the set of user identities and with $issuer(r)$ the user who issued the original or generalized request r. We also indicate with A_{tot} the total area where the LTS provides privacy protection. We assume that a user can issue a request only from the location where she is located, so the spatial component of an original request is the location of the issuer of that request at the time the request is issued. Given $loc_i(t)$ the location of user i at time instant t, this is formalized in Property 1.

Property 1 *For each $i \in I$ and each $r \in R$ such that $issuer(r) = i$, $loc_i(r.Tdata) = r.Sdata$.*

Appendix B contains a summary of the notation used in this dissertation.

2.2.3 Formal definition of generalization functions

In this dissertation we focus our attention on the technique in which the LTS uses a spatial *generalization* function to transform an original request into a generalized one.

A deterministic generalization can be defined as a function $g : R \rightarrow R'$, such that, for each original request r, r and $g(r)$ differ in the $IDdata$ and $Sdata$ fields only. Indeed, during the generalization, the value $r.IDdata$ is either deleted or substituted with a pseudo-id. Moreover, we only consider generalization functions that *preserve the location* i.e., such that the generalized location contains the location of the original request. In terms of our notation, $r.Sdata \in g(r).Sdata$.

If the LTS uses randomized generalizations, the generalization function maps a request $r \in R$ to a probability distribution on R'.

Definition 1 *Given $DIST(R')$ the set of all possible probability distributions on R', $g : R \rightarrow DIST(R)$ is a* randomized **generalization function** *if for each $r \in R$ and $r' \in R'$ with $r' \neq r_{null}$, the fact that r' has a non-zero probability in $g(r)$ implies that $r.Sdata \in r'.Sdata$, $r.Tdata = r'.Tdata$ and $r.SSdata = r'.SSdata$.*

For simplicity of notation, we assume that R and R' are discrete sets; it is not difficult to extend our results for continuous R or R' sets. If the LTS uses a randomized generalization function, when it receives a request r from the user, it randomly chooses a generalized request r' from R' following the probability distribution $g(r)$ and, if $r' \neq r_{null}$, forwards it to the SP.

Clearly a deterministic generalization function is a special case of a randomized generalization function. In the sequel, we will use "generalization function" to mean either a randomized or a deterministic one based on the context.

In the following we need to compute, from a generalized request r', the original request r that is the potential original request from which r' is obtained through a generalization. More formally, we denote with $o(r', i, l)$

the original request r that is issued by user i from a location l at the time $r'.Tdata$ with service specific data $r'.SSdata$. In practice $o(r', i, l)$ is the potential original requests issued by i from l that is possibly generalized to r'.

2.2.4 Formal definition of attacks

The purpose of a generalization function is to render requests *safe* from privacy breaches by fuzzifying the location information. We claim that the safety of a generalization function can only be formally evaluated if the re-identifying abilities of the attacker are clearly stated. We use C to denote the *context* of a possible attack that we describe with a set of assumptions:

- The assumptions about the explicit external knowledge that the attacker could obtain; this could be a public knowledge (e.g., a voter list) or confidential information (e.g., the identity of a user in a given location).

- The assumptions about the reasoning abilities of the attacker; For example, the assumption about the fact that the attacker is (or is not) able to reason with multiple requests issued by the same user in different time instants.

Clearly, C changes depending on the applicative context and on the level of conservativeness of the chosen model. In this paper we assume for all considered contexts that the attacker has basic reasoning abilities like, for example, the ability of joining relations and to check for spatial inclusion.

Given a context C, the attacker aims to infer, from a generalized request, the identity of the user that issued it. We model an attack as the likelihood of associating a generalized request to a specific identity. In practice, an attack on a request r' is a probabilistic distribution on I.

Definition 2 *Given a context C, an* attack *based on context C is a function*

$Att_C : R' \times I \rightarrow [0,1]$ *s.t. for each* $r' \in R'$,

$$\sum_{i \in I} Att_C(r', i) = 1$$

Given an attack Att_C and a generalized request r', we indicate with AS_C the function that associates to each generalized request r' the *Anonymity Set of* r' i.e., the set of users that have a non zero 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 2, it is possible to specify attacks in which, given a request r', the users in the anonymity set of r' have different probabilities of being the issuer of r', as shown in Example 3.

Example 3 *Consider a location based yellow pages service and the following context C:*

- *attacker could acquire the knowledge of the location of each user;*
- *attacker could acquire the knowledge of the users' profiles (possibly obtained during the registration phase);*

Suppose that Alice issues a request asking for the closest shop where she can find classical music. The LTS receives the request and deletes the information that could directly lead to Alice's identity (her name, for example). Moreover, the exact location of Alice is generalized into an area. Then, the resulting generalized request r' *is forwarded to the SP.*

If an attacker obtains r'*, he first uses the location knowledge to restrict the set of possible issuers to the users whose location is in the region specified in* r'*. Suppose this set is composed by three users: Alice, Bob and Carl. Then, the attacker uses the knowledge of the users' profiles, to discover that Alice's musical interests are closer to the classical genre than those of Bob and Carl. Hence, this attack is characterized by a higher likelihood for Alice being the issuer of* r'*, than Bob or Carl.*

A special case of the general definition of attack is the one in which all the users in the anonymity set have the same probability of being the issuer. We call these attacks *uniform*.

Definition 3 *An attack Att_C is* uniform *if, for all request $r' \in R'$, for each pair of users $i, i' \in AS_C(r')$, $Att_C(i, r') = Att_C(i', r')$.*

The relationship between a uniform attack and the anonymity set is formalized by the following proposition.

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

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

For the sake of simplicity, 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. In the following we indicate with $UAtt_C$ the uniform attack characterized by AS_C.

2.2.5 Formal definition of defenses

Once we fix a context C and the corresponding attack Att_C, we can evaluate the safety of a request by measuring how likely is an attacker that exploits Att_C to recover the identity of the issuer of a generalized request.

Definition 4 *Given an attack Att_C and a value $h \in (0, 1]$ a request r' is* safe against Att_C, with re-identification threshold h *if $Att_C(r', issuer(r')) \leq h$.*

If a request is not safe, we say it is unsafe.

In other words, Att_C associates r' to the issuer of r' with a probability. If this probability is below (above, respectively) a given threshold h, then we say r' is safe (or unsafe, respectively).

The task of the LTS is to avoid to forward to a SP a unsafe request. We call *defense function* a generalization function that generates only requests that are safe against a given attack.

Definition 5 *Let Att_C be an attack and h a value in $(0, 1]$. A generalization function g is a* defense function against Att_C with re-identification threshold h *if for each original request $r \in R$ such that $g(r)$ is defined, $g(r)$ is a safe request against Att_C with re-identification threshold h.*

2.2.6 Algorithms to compute the generalization function

One of objectives of this research area is to provide algorithms to compute defense functions. In the following of this dissertation we call *generalization algorithm* an algorithm that takes as input, possibly in addition to other data, an original request and the re-identification threshold and that returns in output a generalized request. Since in this dissertation we focus our attention on the generalization of the spatial attribute only, the aim of the algorithms that we illustrate is to compute the generalized area. Once this region is computed, the generalized request is obtained through the *finalizeGeneralization* procedure that takes as input an original request r, a generalized area A and returns a generalized request r' with the same time and service specific data as r, with the explicit identifiers (e.g., name of the user, IP address, etc...) properly masked and with the exact location substituted with A.

The objective of the generalization algorithms is to guarantee user's anonymity. A generalization algorithm that is proved to compute a defense function against a given attack is called a *defense algorithm.*

Definition 6 *Let gen be a generalization algorithm, Att_C an attack and h a value in $(0, 1]$. We say that gen is a* defense algorithm against Att_C *if the execution of gen with h as re-identification threshold computes a defense function against Att_C with re-identification threshold h.*

2.2.7 Extension of the framework for the dynamic case

The framework we defined models the static case. A straightforward extension is required to model the dynamic case; Indeed, since the attacker

can identify traces of requests, the generalization of each request depends on how previous requests from the same user have been generalized.

In the following of this dissertation, we denote with Θ the set of all possible traces i.e., the set in which each element is a set of generalized requests issued by the same user. Formally, $\Theta \in 2^{R'}$ is such that, for each set $\tau \in \Theta$, for all pairs of requests $r'_1, r'_2 \in \tau$, $issuer(r'_1) = issuer(r'_2)$. The elements $\tau \in \Theta$ represents the requests already issued by a user. Since each set of requests $\tau \in \Theta$ is issued by a single user, we indicate with $issuer(\tau)$ the issuer of the requests.

We can now define a *deterministic generalization function (in the dynamic case)* as a function $g : R \times \Theta \rightarrow R'$. Intuitively, $g(r, \tau)$ is the request r' that generalizes r when τ is the set of requests already issued by $issuer(r)$. We can similarly extend the definition of randomized generalization function.

Definition 7 *Given $DIST(R')$ the set of all possible probability distributions on R', $g : R \times \Theta \rightarrow DIST(R')$ is a* randomized generalization function (in the dynamic case) *if for each $r \in R$, $\tau \in \Theta$ and $r' \in R'$ with $r' \neq r_{null}$, the fact that r' has a non-zero probability in $g(r, \tau)$ implies that $r.SData \in r'.SData$, $r.Tdata = r'.Tdata$ and $r.SSdata = r'.SSdata$.*

The definitions of attack, safe request and defense function defined for the static case do not need to be extended for the dynamic case. On the contrary, the definitions of generalization algorithm and of defense algorithm need to be extended by stating that the input includes, in addition to the original request r, the re-identification threshold h and possibly other paramethers, the set τ of generalized request already issued by the issuer of r.

Chapter 3

Techniques to enforce anonymity in the static case

In this chapter we specify three contexts in the static case. Contexts C_{st} and C_{st+g} (presented in Sections 3.1 and 3.2, respectively) had been considered, implicitly or explicitly, in most of the previous works [24, 22, 39, 29, 8, 37, 23, 6]. Context C_{ast+g}, defined in Section 3.3, has not been considered in any previous work, to the best of our knowledge. For each of the three contexts, we specify the corresponding attack, we describe the existing defenses (if exist) and propose new generalization algorithms.

We conclude this chapter with the experimental results to empirically compare the generalization algorithms proposed for contexts C_{st} and C_{st+g}.

3.1 Context C_{st}: a model for spatio-temporal anonymity

3.1.1 Context definition

Most of the papers in this research area did not explicitly state the assumptions about the external knowledge available to the attacker and about his reasoning abilities. However, many papers [24, 22, 39, 29] implicitly considered a common context that we refer to as C_{st} and that we model in this section.

The idea of the C_{st} context is that users can be identified through their location, and therefore user's privacy is endangered if a generalized request contains precise information about issuer's location. For example, an attacker can understand user's identity from user's location through the physical observation of the user, or because the user is the only one that can be in that location (a suburban house, for instance).

In context C_{st} it is assumed that the attacker could acquire the knowledge about the exact location of each user and that the attacker is not able to reason with more than one request. It may seem unrealistic that the attacker knows the location of each user; however, if an attacker can possibly know the location of one user (not too outrageous an assumption), it is quite natural to assume the worst case, namely he knows the location of all users. This assumption may be relaxed by saying that the attacker can only have an approximate knowledge of the locations where some users are located. This knowledge is modeled in Section 3.3.

The uniform attack based on context C_{st} is characterized by the anonymity set of users who are located within the area specified in the generalized request.

Definition 8 *Let C_{st} be the context in which it is assumed that for each identity $i \in I$ the attacker knows the association of i with the location of i.*

For each generalized request r', the anonymity set based on context C_{st} is:

$$AS_{C_{st}}(r') = \{i \in I | loc_i(r'.Tdata) \in r'.Sdata\}$$

The idea is that the more users are located inside $r'.Sdata$, at the time the request is issued, the more anonymous is the issuer of r'.

Definition 9 *In context C_{st}, a generalized request r' is said to be* k-anonymous *if $|AS_{C_{st}}(r')| \geq k$.*

The solutions to the anonymity problem in context C_{st} provided in [22, 39, 29, 24] guarantee anonymity by generalizing the $Sdata$ field of the

incoming request and forwarding to the SP only requests r' such that r' is k-anonymous in context C_{st}.

Theorem 1 shows that, considering context C_{st}, a generalization function that generates k-anonymous requests is a defense function, and that the parameter k is the inverse of h. The proof of the theorem, as well as all the other proofs of the theorems in this dissertation, can be found in Appendix A.

Theorem 1 *Let k be an integer and $g()$ a generalization function such that, for each original request r, $g(r)$ is k-anonymous in context C_{st}. Then, $g()$ is a defense function against $UAtt_{C_{st}}$ with threshold $1/k$.*

3.1.2 Defense algorithms against $Att_{C_{st}}$ proposed in the literature

In the following, we describe three generalization algorithms proposed in the literature. From Theorem 1, it easily follows that the algorithms are defense algorithm against $Att_{C_{st}}$. For each algorithm, we also discuss its temporal complexity.

Interval Cloaking Generalization Algorithm

The first algorithm to compute a generalization function that appeared in the literature was named *IntervalCloaking* [24]. The idea of the algorithm is to iteratively divide the total region A_{tot} where the LTS provides privacy protection. At each iteration the current area q_{prev} is partitioned into quadrants of equal size. If less than k users are located in the quadrant q where the issuer of the request is located, then q_{prev} is returned. Otherwise, iteration continues considering q as the next area.

In order to evaluate the time complexity of the algorithm it is necessary to make some assumptions about the data structures. In our implementation of the algorithm, we used a data structure consisting of a quadTree in which each leaf has a pointer to a user, and each internal node n stores the number

of users "contained" in n i.e., the number of users stored in the subtree that has n as root. The generalization algorithm traverses the quadTree from the root to the first internal node that contains at least k users. Each iteration of the algorithm is performed in constant time and the number of iterations is bounded by the height of the quadTree. In the worst case, the height of the tree is linear in the cardinality of the set I of users hence the algorithm has a worst case time complexity of $O(|I|)$. However, if users are uniformly distributed, the height of the tree is logarithmic in the number of users hence the algorithm has a worst-case time complexity of $O(\log(|I|))$.

Casper Generalization Algorithm

Mokbel et al. [39] propose *Casper*, a framework for privacy protection that includes a generalization algorithm. In this paper we consider the "basic" data structure[1] used by Casper i.e., a balanced quadTree in which each node has a pointer to its parent, and users are stored in leaf nodes only. Moreover, the data structure consists of a table in which each user i is associated with the leaf node that contains i. The generalization algorithm starts from the leaf node that contains the issuer of the request, and iteratively traverses the tree towards the root until an area that contains at least k users is found. At each iteration, the algorithm considers the union of the area covered by the current node n and the horizontally (vertically, resp.) contiguous area covered by its sibling node. If only one of these two joined areas contains more than k users, that area is returned; if both of them contain more than k users, the one containing the minimum number is returned; otherwise, the algorithm proceeds with the next iteration. Similarly to *IntervalCloaking*, the worst case time complexity of *Casper* is linear in the height of the quadTree. However, in this case, this height is bounded by the logarithm of the number of leaf nodes if users are uniformly distributed, and it is at most linear in the same number, otherwise.

[1] For the purpose of this dissertation, there is no need to consider the "adaptive" data structure proposed in the paper.

nnASR Generalization Algorithm

Conceptually, one of the simplest ways to generalize a request is to compute the k Nearest Neighbor query among the users and return the MBR of the result. Following this idea, Kalnis et al. [29] propose the *nnASR* generalization algorithm that picks a random user i in the set of the $k-1$ users that are closest to the issuer, and returns the MBR of the set containing i, the issuer, and the $k-1$ users closest to i. In our implementation of the *nnASR* algorithm we used a kd-Tree to store users' locations, making possible to compute k Nearest Neighbor queries in logarithmic expected time with respect to the number of users.

3.2 Context C_{st+g}: a model for spatio-temporal anonymity when the generalization function is known to the attacker

3.2.1 Context definition

In this section we give a formal explanation to the counterattacks illustrated in the literature to the defense techniques first proposed in context C_{st}. In a nutshell, the counterattacks implicitly consider the generalization function as publicly known, while the defense technique assumed otherwise.

Assume an attacker obtains a generalized request r' and knows, in addition to the knowledge of C_{st}, the generalization function g used by the LTS. In this case, the attacker can compute, for each user $i \in I$, the *potential original request* r_i that would be send to the LTS if i was the issuer. Then the attacker can check if $g(r_i)$ equals r'. If it does, then i is a potential issuer, otherwise the attacker can understand that i is not the issuer of r'.

Example 4 *User i_1 issues a request r_1 to the LTS. The required anonymity threshold h is $1/3$, hence r_1 is generalized, using algorithm g, to a request r' such that i_1, i_2 and i_3 are located inside $r'.Sdata$.*

If an attacker obtains r', he can compute, for each user $i \in I$, the potential request r_i and then apply $g()$ to each r_i. Since r'.Sdata includes the locations of i_1, i_2 and i_3 only, for any user i not in $\{i_1, i_2, i_3\}$ the potential request r_i is generalized to a request $r'' \neq r'$. So, the only three user whose potential request could be generalized to r' are i_1, i_2 or i_3. Assume that r_3 is generalized to a request r''' that is different from r' while both r_1 and r_2 are generalized into r'. Figure 3.1 gives a graphical representation of this situation.

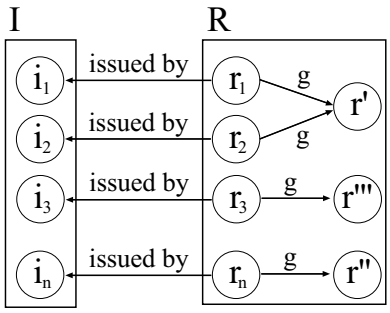


Figure 3.1: Example of attack in context C_{st+g}.

Since i_1 and i_2 have the same probability of being the issuer of r', the attacker can identify the issuer of r' with probability $1/2$, hence r' is unsafe with respect to this attack with the required threshold of $1/3$.

We indicate with C_{st+g} the context in which the attacker, in addition to the knowledge and reasoning abilities of the context C_{st}, also has the knowledge of the procedure that the LTS used to compute the generalization $g()$. In this context the following attack can be defined:

Definition 10 *Let C_{st+g} be the context C_{st} in which the attacker also knows the procedure used by the LTS to compute the generalization function $g()$.*

The attack based on context C_{st+g} is given by

$$Att_{C_{st+g}}(r', i) = \frac{P[g(o(r', i, loc_i(r'.Tdata)) = r']}{\sum_{j \in I} P[g(o(r', j, loc_j(r'.Tdata)) = r']}$$

Intuitively, the attack $Att_{C_{st+g}}(r', i)$ is the probability that an original request r issued by user i is generalized to r'. This probability is then normalized for all the users in I.

Theorem 2 proves that, $AS_{C_{st+g}}(r') \subseteq AS_{C_{st}}(r')$. This implies that $Att_{C_{st+g}}$ has more re-identification power than the uniform attack in context C_{st}. In addition, it suggests an efficient procedure to compute $Att_{C_{st+g}}(r', i)$. Indeed, given $r_i = o(r', i, loc_i(r'.Tdata))$ the potential request issued by user i, $P[g(r_i) = r'] = 0$ if the location of i is outside $r'.Sdata$.

Theorem 2 *For each generalized request r', $AS_{C_{st+g}}(r') \subseteq AS_{C_{st}}(r')$.*

Intuitively, Theorem 2 states that, given a generalized request $r' = g(r)$, the set of possible issuers of r' computed exploiting the knowledge of the function $g()$ is a subset of the set of users whose location is within $r'.Sdata$. This implies that, if $g()$ is known to the attacker, a request that is k-anonymous according to Definition 9, may nevertheless be associated to the issuer with likelihood greater than $1/k$.

Corollary 1 *Let r' be a generalized request. The k-anonymity of r' in context C_{st} is a necessary but not sufficient condition for r' to be a safe request against $Att_{C_{st+g}}$ with threshold $1/k$.*

The "necessary" part immediately follows Theorem 2, while the "not sufficient" part is shown with Example 5.

Corollary 1 shows that the notion of k-anonymity of a request is not sufficient in order to guarantee user's privacy when the generalization function is known to the attacker. In Example 5 we show, in terms of our framework, why the generalization function proposed in [24] is not a defense function against $Att_{C_{st+g}}$ with threshold $1/k$.

Example 5 *Figure 3.2 shows four user locations. Users u_1, u_2 and u_3 are close to each other, while u_4 is far away. Consider a generalization function $g()$ that gives the minimum rectangular area that contains the area of a request as well as at least other $k-1$ users. In our example, a request issued*

by u_1, u_2 or u_3 is generalized into a request that has A_2 as Sdata while a request issued by u_4 is generalized into a request that has A_1 as Sdata.

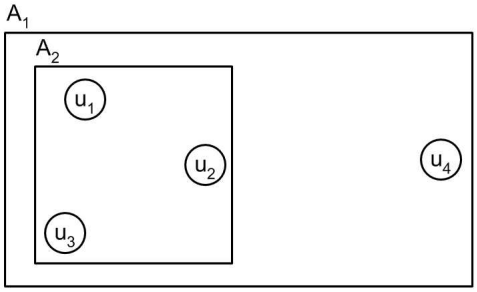


Figure 3.2: Example of attack when the generalization function is known to the attacker.

As we proved in Theorem 1, $g()$ is a defense function against the uniform attack characterized by $AS_{C_{st}}$ with threshold $1/3$. On the contrary, if we assume that the attacker knows the generalization function, then $g()$ is not a defense function against the uniform attack characterized by $AS_{C_{st+g}}$ for any threshold in $(0, 1)$. Indeed, let r be a request issued by u_4 and r' the generalization of r such that $r'.Sdata = A_1$. Since any request issued by u_1, u_2 or u_3 would be generalized into a request different from r', the only possible issuer of r' is u_4.

3.2.2 A class of defense algorithms

In this section we show how the formal framework defined in Section 2.2 can be useful to define a defense function against a given attack. In particular, we consider $Att_{C_{st+g}}$, as defined in Section 3.2.1, and we show how to define a class of algorithms to compute a defense function.

Since the attacker's knowledge assumed in C_{st+g} subsumes the attacker's knowledge in C_{st}, and in the two contexts the attacker has the same reasoning abilities, the defense function against $Att_{C_{st+g}}$ should also provide protection against $UAtt_{C_{st}}$. This consideration suggests us a two steps procedure to define the algorithm that computes the defense function against $Att_{C_{st+g}}$:

1. define algorithm Gen_{st} that provides protection against $UAtt_{C_{st}}$;

2. alter Gen_{st} to obtain Gen_{st+g} that provides protection against $Att_{C_{st+g}}$.

We define algorithm Gen_{st} inspired by the *IntervalCloaking* generalization algorithm [24] described in Section 3.1.2. The idea of Gen_{st} is to iteratively restrict the anonymity set that is stored in the variable AS initialized to I. At each iteration AS is partitioned. If the block b that contains the issuer contains at least k users, then AS is set to b and the iteration continues. Otherwise, the algorithm terminates returning the generalized request that has the MBR (Minimum Bounding Rectangle) of the locations of the users in AS as the generalized location. Gen_{st} does not automatically provide protection against $Att_{C_{st+g}}$ due to a problem similar to that described in Example 5. A solution consists in: i) imposing the partitioning function to be independent from the issuer's location and ii) modifying the termination condition of Gen_{st} so that the algorithm terminates if any of the blocks into which AS is partitioned contains less than k users. The resulting algorithm is Gen_{st+g} and is shown as Algorithm 1.

Algorithm Gen_{st+g} iteratively restricts the set AS of candidate identities by successively partitioning it. The iteration terminates when a further partitioning would contain at least one block with less than k users. Finally, the algorithm returns the set of the users in the anonymity set.

Note that Algorithm 1 does not specify how the users should be partitioned during each iteration. Provided that the partition is computed independently from the issuer, any procedure to partition the set of users can be used. Hence, in practice, Algorithm 1 is a class of algorithms that can be instantiated providing a specific procedure to compute the partition.

The correctness of Algorithm 1 follows from the fact that, given $r' = Gen_{st+g}(r, h)$, the set of users whose location is inside $r'.Sdata$ is such that, if any of them issues a request r_1, then $Gen_{st+g}(r_1, h)$ would return a generalized request with the same area as r'. In order to have this property, the partitioning procedure should be computed independently from r. This is the intuition of the proof of Theorem 3 that formalizes the correctness of

Algorithm 1 Gen_{st+g}

Input: an original request r, a value $h \in (0, 1]$.
Output: a generalized request that is safe against $Att_{C_{st+g}}$ with re-identification threshold h.
Method:

1: $AS = I$;
2: $cont$ =**true**
3: **while** ($cont$) **do**
4: partition AS into $AS_1, \ldots, AS_n$;
5: **if** ($\exists j \in \{1, \ldots, n\}$ such that $|AS_j| < 1/h$) **then**
6: $cont$ =**false**;
7: **else**
8: $AS = AS_j$ where AS_j is the block in $AS_1, \ldots, AS_n$ that contains $issuer(r)$;
9: **end if**
10: **end while**
11: **return** $finalizeGeneralization(r, MBR(AS))$

the algorithm.

Theorem 3 *If the partition procedure of Line 4 of Algorithm 1 is deterministic and has AS as the only parameter, then algorithm* Gen_{st+g} *is a defense algorithm against* $Att_{C_{st+g}}$.

3.2.3 A defense algorithm proposed in the literature

To the best of our knowledge, the first defense algorithm against $Att_{C_{st+g}}$ was proposed by Kalnis et al. [29], and it was called *hilbASR*. The algorithm is an instance of Gen_{st+g} in which the partitioning is obtained exploiting the Hilbert space filling curve[2] to define a total order among users' locations. A data structure is then used to store users in the order defined through the Hilbert space filling curve. Intuitively, the *hilbASR* generalization algorithm partitions the data structure into blocks of k users: the first block from the user in position 0 to the user in position $k-1$ and so on (note that the last

[2]A space filling curve transforms 2-D coordinates into 1-D coordinate. With high probability, two points that are close to each other in the 2-D coordinates are also close to each other in the transformed 1-D coordinate.

block can contain up to $2 \cdot k - 1$ users). The algorithm then returns the block that contains the issuer. The worst case time complexity of hilbASR is $O(log(|I|))$. Indeed, the algorithm does not actually compute all the blocks of the partition, but instead only finds the block that contains the issuer. Therefore, the only non-constant operation that is required is to find the issuer in the data structure, and this operation can be completed in logarithmic time.

3.2.4 *DichotomicPoints* defense algorithm

In the preliminary paper [37], Mascetti et al. propose a defense algorithm against $Att_{C_{st+g}}$ called *dichotomicPoints* that is shown in Algorithm 2. The algorithm iteratively partitions the set of users into two sets. The idea is that the users are totally ordered according to their locations considering first one axis, then the other, and if necessary even the user identifier[3]; Then, considering the user u whose position is in the middle of the totally ordered set of positions[4], the algorithm partitions the users into two blocks: the ones before u, and the remaining ones. Iteration continues considering only the users in the block that contains the issuer.

At each iteration, in order to chose which axis to consider first for the total ordering, the algorithm computes the maximum and minimum values of users' locations projected on each axis. Then, the axis having the maximum value of the difference is chosen (Lines 3 to 13). The *dichotomicPoints* algorithm terminates when the block that contains the issuer has cardinality smaller than $2 \cdot k$; Then it returns the generalized request having the MBR of the users in the block as generalized location.

The idea of computing a generalization by iteratively splitting the users along one of the dimensions also appeared in the *Anonymize* algorithm presented in [33] to generalize database relations. Despite this similarity, the two algorithms have been independently designed.

[3]We assume that each user has a unique user identifier.

[4]When there is an even number r of users, user u is the one in position $r/2$.

Algorithm 2 *dichotomicPoints*

Input: an original request r, a value $h \in (0, 1]$.
Output: a generalized request that is safe against $Att_{C_{st+g}}$ with re-identification threshold h.
Method:

1: the array AS is initialized with the identities of all users (I)
2: **while** ($|AS| \geq 2/h$) **do**
3: $min_x = min_{i \in AS}(loc(i).x)$
4: $max_x = max_{i \in AS}(loc(i).x)$
5: $min_y = min_{i \in AS}(loc(i).y)$
6: $max_y = max_{i \in AS}(loc(i).y)$
7: **if** $((max_x - min_x) \geq (max_y - min_y))$ **then**
8: $firstOrder = x$
9: $secondOrder = y$
10: **else**
11: $firstOrder = y$
12: $secondOrder = x$
13: **end if**
14: sort AS according first to $firstOrder$, then to $secondOrder$, and eventually to user identifiers.
15: $pivot = \lfloor |AS|/2 \rfloor$
16: $issuerIndex =$ the index such that $AS[issuerIndex] = issuer(r)$
17: **if** ($issuerIndex < pivot$) **then**
18: $AS = AS[0] \ldots AS[pivot - 1]$
19: **else**
20: $AS = AS[pivot] \ldots AS[|AS| - 1]$
21: **end if**
22: **end while**
23: **return** $finalizeGeneralization(r, MBR(AS))$

Example 6 *Consider the positions of* 20 *users, as shown in Figure 3.3(a). Assume that user i issues a request and that the re-identification threshold is* 1/2. *The* dichotomicPoints *algorithm computes the necessary generalization in three iterations of its main cycle. Each of the three Subfigures 3.3(a) 3.3(b), and 3.3(c) shows the set of positions included in the current AS array computed by that iteration (as black filled circles), as well as the rest of the users' positions.*

The array AS is initialized to the identities of the 20 *users in our ex-*

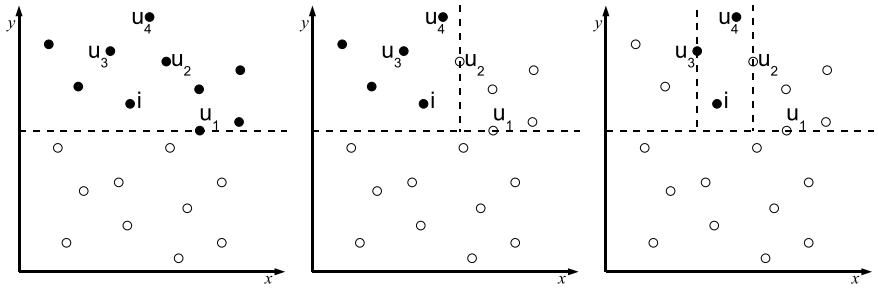
ample. In the first iteration, the difference between the maximum and minimum value over the set of all projections of user positions on the x axis is compared with the difference between the maximum and minimum value on the y axis. Since in this example the difference along the y axis is larger, the array AS is sorted considering the values on this dimension first. The pivot variable is assigned with the value 10*, which in AS happens to be the index of user u_1. Since the issuer i has index* 12*, AS is reassigned to contain only the identities of u_1 and of the users with greater indexes (i.e., $AS = AS[10], \ldots, AS[19]$).*

In the second iteration, considering the projections on the x and y axis of the positions of the 10 *users currently in AS, the x axis is selected as $firstOrder$; the reason can be clearly seen observing the black filled circles in Figure 3.3(a) that also represents the set of positions considered in the second iteration. Hence, this time AS is sorted considering first projections of positions on x. In the resulting array, the pivot is computed as* 5*, which in Figure 3.3(b) corresponds to user u_2. Since now the issuer i has index* 3*, AS is reassigned to contain only the identities of the users with indexes smaller than* 5*, i.e., the users located at the left of u_2. ($AS = AS[0], \ldots, AS[4]$).*

In the third iteration, AS is again sorted considering first projections of positions on x. The pivot is 2*, which in Figure 3.3(c) corresponds to user u_3. Since the issuer i has index* 3*, AS is reassigned to contain $AS[2]$, $AS[3]$, and $AS[4]$, i.e., the identities of the users with indexes greater than or equal to* 2*, discarding the users whose location is at the left of u_3.*

Since AS now contains less than $2 \cdot k = 4$ users, the algorithm terminates returning users i, u_3 and u_4.

Note that *dichotomicPoints* terminates when a block contains less than $2 \cdot k$ users and therefore any further partitioning would generate a block with less than k users. Therefore, *dichotomicPoints* is an instance of the class of defense algorithms presented in Section 3.2.2, hence it is a defense algorithm against $Att_{C_{st+g}}$.



(a) First iteration (b) Second iteration (c) Third iteration

Figure 3.3: Example of *dichotomicPoints* algorithm

Theorem 4 dichotomicPoints *is a defense algorithm against* $UAtt_{C_{st+g}}$.

The data structure that we used in the implementation of *dichotomicPoints* consists of two arrays, $order_x$ and $order_y$, containing the users ordered according to the horizontal and vertical axis, respectively. At each iteration, the user locations that are not in the same block as the issuer are removed from the two arrays. So, at each iteration it is necessary to find the user location in the middle of the correct array, to count how many users will be in each block and to remove the users that are not in the same block as the issuer. The first two operations can be performed in constant time, while the last one requires a time linear in the size of the two arrays. Since the number of iterations is logarithmic in the number of users and each iteration requires time linear in the number of the users, the worst case time complexity of the algorithm is $O(|I| \cdot \log(|I|))$.

3.2.5 *Grid* defense algorithm

A different defense algorithm against $Att_{C_{st+g}}$ is shown in Algorithm 3 and it is called *grid*. This algorithm partitions the set of users in two steps. During the first step, users are totally ordered considering their location along the x axis, then along the y axis and eventually according to their user identifier. This ordered set of users is then partitioned into blocks of consecutive users,

each block having the same number of users except the last block that may contain more users than the previous ones. Only the users that are in the same block as the issuer are considered in the second step in which users are ordered considering first their location along the y axis, then their location along the x axis and eventually according to their user identifier. Using this ordering, users are partitioned similarly to what is done in the first step. Eventually, a generalized request is returned having as generalized location the MBR of the users that are in the same block as the issuer.

Algorithm 3 *Grid*

Input: an original request r, a value $h \in (0, 1]$.
Output: a generalized request that is safe against $Att_{C_{st+g}}$ with re-identification threshold h.
Method:

1: $k = \lceil 1/h \rceil$
2: the array AS is initialized with the identities of all users (I)
3: $nob = \left\lfloor \sqrt{|AS|/k} \right\rfloor$ {number of blocks}
4: **if** ($nob \leq 1$) **return** the set of all user identities (I)
5: **for** ($dim \in \{x, y\}$) **do**
6: **if** ($dim = x$) **then**
7: sort AS according first to the location on x, then to the location on y and eventually to the user identifier.
8: **else**
9: sort AS according first to the location on y, then to the location on x and eventually to the user identifier.
10: **end if**
11: $issuerIndex$ = the index such that $AS[issuerIndex] = issuer(r)$
12: $upb = \lfloor |AS|/nob \rfloor$ {users per block}
13: $ibi = \lfloor issuerIndex/upb \rfloor$ {issuer's block index}
14: **if** ($|AS|$ mod $nob = 0$ OR $ibi < nob - 1$) **then**
15: $start = ibi \cdot upb$
16: $end = start + upb - 1$
17: **else**
18: $start = (nob - 1) \cdot upb$
19: $end = |AS| - 1$
20: **end if**
21: $AS = AS[start] \ldots AS[end]$
22: **end for**
23: **return** $finalizeGeneralization(r, MBR(AS))$

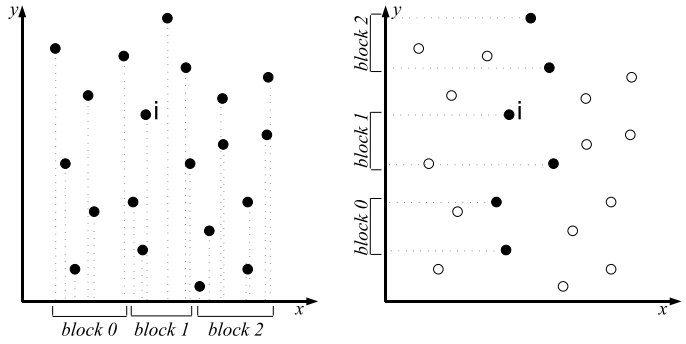
Example 7 *Figure 3.4(a) shows the same position of* 20 *users as considered in Example 6 with the issuer i of a request in the same position. We show the execution of the* grid *algorithm, with re-identification threshold equals to* $1/2$.

The array AS is set to contain the identities of the 20 *users. In the first step, AS is sorted considering first the values of users' locations on the x axis. Since the number of blocks (nob) into which AS should be partitioned is* $\left\lfloor \sqrt{20/2} \right\rfloor = 3$, *the number of users in each block (upb) is computed as* $\lfloor 20/3 \rfloor = 6$. *Then, since the issuer has index* 8 *in AS, the index of the block containing the issuer (ibi) is computed as* $\lfloor 8/6 \rfloor = 1$. *Consequently, AS is reassigned to the set of users in the same block as the issuer. More technically, start and end are computed as* 6 *and* 11, *and AS is set to contain only* $AS[6], \ldots, AS[11]$.

In the second step, AS, currently containing 6 *users (identified in Figure 3.4(b) as black filled circles) is sorted considering first the users' positions along the y axis. Since the value of nob cannot change after it is initialized, AS is again partitioned into* 3 *blocks. Considering that AS now contains* 6 *user identities, each of the new blocks has to contain* 2 *users. Since the issuer has now index* 3 *in AS, the index of the block that contains the issuer (ibi) is computed as* $\lfloor 3/2 \rfloor = 1$. *Therefore, AS is reassigned to contain only* $AS[2]$ *and* $AS[3]$, *and this is the set of users' identities returned by the algorithm.*

Theorem 5 proves that *grid* is a defense algorithm against $Att_{C_{st+g}}$. Intuitively, the correctness of the algorithm relies on the fact that the product of the number of blocks into which users are partitioned along the two dimensions is less than or equal to $\lfloor |I| \cdot h \rfloor$. This implies that the set I of users is partitioned at most into $\lfloor |I| \cdot h \rfloor$ blocks of about the same size, hence each block contains at least $1/h$ users. This is the reason why the number of blocks along each dimensions is $\left\lfloor \sqrt{|I| \cdot h} \right\rfloor$.

Theorem 5 grid *is a defense algorithm against* $Att_{C_{st+g}}$.



(a) First step (b) Second step

Figure 3.4: Example of *grid* algorithm

In order to compute the *grid* algorithm, we use a data structure that keeps the users totally ordered according to their location along the x axis, then along the y axis and eventually according to their user identifier. Therefore, during the first step of the algorithm it is not necessary to sort the set of users and the only non-constant operation is to find the index of the issuer. This operation has a worst case time complexity logarithmic in the size of $|I|$. During the second step, it is necessary to sort the users that, during the first iteration were in the same bucket as the issuer. Since these users are at most $2(|I|/\left\lfloor\sqrt{|I|\cdot h}\right\rfloor)-1$, in the worst case this operation can be performed in time $O(\sqrt{|I|/h}\cdot\log\sqrt{|I|/h})$. Hence, the worst case time complexity of *grid* algorithm is given by $O(\log|I|+\sqrt{|I|/h}\cdot\log\sqrt{|I|/h})$ that is equal to $O(\sqrt{|I|/h}\cdot\log\sqrt{|I|/h})$.

3.3 Context C_{ast+g}: modeling approximate knowledge of users' location

In this section we model the case in which the attacker has an approximate knowledge of users' locations. In the following, we present a formalism to model this kind of attacker's knowledge, we specify the attack based on

this knowledge and we present our preliminary results in the definition of a defense function.

3.3.1 Context definition

We model the case in which the attacker has an approximate knowledge of users' locations by assuming that the attacker knows, for each user, the probabilistic distribution of possible areas where the user is located. In Section 3.3.3 we show how this probabilistic distribution can be derived from publicly available external information.

To formally model this knowledge, we define a partition $\perp_S$ of the total area A_{tot} where the LTS protects users' privacy. Intuitively, $\perp_S$ represents the *bottom spatial granularity* and therefore we call its elements *spatial granules*. In the following we assume that each $2D$-region we refer to can be represented as the union of a finite number of spatial granules. Consequently, given a $2D$-region A, we indicate with $s \in A$ the set of granules of $\perp_S$ whose union yields the region A.

Once $\perp_S$ is defined, we model the attacker's external knowledge as the probabilistic distribution of the possible locations where each user is located.

Definition 11 *Given a set of users I and the bottom spatial granularity $\perp_S$, the* possible user locations function *$pul : I \times \perp_S \rightarrow [0, 1]$ is such that for each user $i \in I$, $\sum_{s \in \perp_S} pul(i, s) = 1$.*

Once a possible user locations function is defined for each spatial granule, it can be easily extended for each area A in A_{tot} as follows:

$$\forall i \in I, pul(i, A) = \sum_{s \in A} pul(i, s)$$

Given the possible user location function, we can define the probability that, a single user located in an area A has identity i. We denote this probability as $p(i, A)$. Intuitively, the value of $p(i, A)$ is given by the probability of i being in A, divided by the estimated number of users in A that is given by the sum, for each user in I, of the probability of i being located in A. If there

are no users having any probability of being located in A, we assume $p(i, A)$ to be equal to zero for each $i \in I$. Formally, given $enu(A) = \sum_{i \in I} pul(i, A)$

$$p(i, A) = \begin{cases} 0 & \text{if } enu(A) = 0 \\ \frac{pul(i,A)}{enu(A)} & \text{otherwise} \end{cases}$$

We can now define the attack that can be performed with an approximate knowledge of users' locations.

Definition 12 *Let C_{ast+g} be the context in which the attacker knows, i) the generalization algorithm used by the LTS, ii) the set I of users, and iii) the function $pul(i, s)$, for each $i \in I$ and each $s \in \perp_S$.*

Then, for each generalized request r', and each user $i \in I$, the attack based on context C_{ast+g} is:

$$Att_{C_{ast+g}}(r', i) = \frac{\sum_{s \in r'.Sdata}(p(i, s) \cdot P[g(o(r', i, s)) = r'])}{\sum_{j \in I} \sum_{s \in r'.Sdata}(p(j, s) \cdot P[g(o(r', j, s)) = r'])}$$

The idea of Definition 12 is that the probability of a user being the issuer of a request r' is given by the sum, for each granule $s \in r'.Sdata$, of the probability of the user issuing the request from s times the probability that an original request issued by i from s is generalized to r'. This value is then normalized to 1 (by the denominator) for all the users in I.

Example 8 *Let consider a set $\perp_S$ equals to $\{s_1, s_2, s_3, s_4, s_5\}$ and that a set of users I equals to $\{i_1, i_2, i_3\}$. The values of the pul function at time t are shown in Table 3.1.*

	i_1	i_2	i_3
s_1	1/3	1/36	1/4
s_2	1/3	1/36	1/24
s_3	1/4	1/36	1/24
s_4	1/18	1/4	4/9
s_5	1/36	2/3	2/9

Table 3.1: Values of the *pul* function

Let r' be a generalized request issued by i_1 with an anonymity threshold $h = 1/2$ such that $r'.Sdata = \{s_1, s_3\}$ and $r'.Tdata = t$. Moreover, let the generalization function g used by the LTS be such that $P[g(o(r', s_1)) = r'] = 1$ and $P[g(o(r', s_3)) = r'] = 1$. Then, the attack based on context C_{ast+g} associates r' to user i_1 with likelihood

$$\frac{1/3 + 1/4}{1/3 + 1/4 + 1/36 + 1/36 + 1/4 + 1/24} = 42/67 > 1/2$$

Therefore, r' is unsafe with respect to attack Att_{ast+g} with threshold $1/2$.

3.3.2 *PartitionArea* generalization algorithm

Algorithm 4 shows a generalization algorithm. The idea of the algorithm is similar to the idea of the *dichotomicArea* algorithm [37] and consists in iteratively partitioning a $2D$-region A that is initialized to A_{tot}. At each iteration, a partition of A is computed. For each block B, if any user could be identified, using context C_{ast}, as the issuer of a request having B as the generalized location, with likelihood greater than h, then the algorithm terminates returning a generalized request that has A as generalized location. Otherwise, iteration continues and the value of A is set to the block that contains the location of the issuer.

We believe that *PartitionArea* is a defense algorithm against C_{ast+g} because, if r' is the output of $PartitionArea(r, h)$, then for any other request r_1 issued from $r'.Sdata$, $PartitionArea(r_1, h)$ returns a request r'_1 that has the same generalized area as r'. We leave the formal proof as a future work.

Conjecture 1 PartitionArea *is a defense algorithm against $Att_{C_{ast+g}}$.*

The efficiency of Algorithm 4 relies on the complexity of the evaluation of the **if** statement at Line 6 and on the partition procedure used at Line 5. In our implementation, we used a partitioning procedure and we adopted a data structure that make it possible to execute Algorithm 4 in logarithmic time with respect to the number of granules of $\perp_S$ whose union yields A_{tot}.

For the sake of simplicity, in the following we assume the granules of $\perp_S$ to be squares of the same size, and A_{tot} to be a rectangle. Note that

Algorithm 4 PartitionArea

Input: an original request r, a value $h \in (0, 1]$.
Output: a generalized request that is safe against $Att_{C_{ast+g}}$ with re-identification threshold h.
Method:

1: $A = A_{tot}$
2: $continue =$ **true**
3: $i = issuer(r)$
4: **while** (continue) **do**
5: $par = partition(A)$
6: **if** (exists a user $j \in I$ and a block B in par s.t. $p(j, B) > h$) **then**
7: $continue =$**false**
8: **else**
9: $A =$ the block B of par s.t. $r.Sdata \in B$
10: **end if**
11: **end while**
12: **return** $finalizeGeneralization(r, A)$

our solution can be easily adapted to work without these assumptions. The partitioning procedure $partition(A)$ used at Line 5, partitions the parameter area A in two sub-areas along one of the two axis, choosing the axis on whose projection A is larger. Intuitively, if A is a rectangle covering x granules on one dimension and y granules on the other, assuming $x > y$, the two blocks resulting from the partition of A will have $\lfloor x/2 \rfloor \cdot y$ and $\lceil x/2 \rceil \cdot y$ granules, respectively.

Based on this partitioning function, we define the data structure $pTree$ composed by a binary tree in which each node $pNode$ points to a region A (we denoted this as $pNode.A$). The root of the tree points to A_{tot} and the leafs point to elements of $\perp_S$. Each internal node $pNode$ has two children, pointing to the two blocks of $partition(pNode.A)$. Each $pNode$ has also a pointer to the value $pMax$ (denoted by $pNode.pMax$), that is the maximum value of $p(i, pNode.A)$ for all the users $i \in I$.

Using this data structure, the *PartitionArea* algorithm can be computed by traversing the tree from the root to the leafs. The condition of the **if** statement can be evaluated in constant time, since it requires to compare h

with the values of $pMax$ for the two children of the $pNode$ that is currently being processed. Since the partition procedure is also executed in constant time, each iteration requires a constant time. The number of iteration is bounded by the height of the binary tree that logarithmic in the number of granules of $\perp_S$ whose union yields A_{tot}.

3.3.3 Specification of the *pul* function from publicly available information

In Section 3.3.1 the *pul* function is defined to specify an attack based on context C_{ast+g}. However, in practice, it is unlikely that external knowledge available to the attacker contains the values of this function. Nevertheless, the function can be derived from some other forms of external knowledge. In this section we show how the *pul* function can be derived from a partial function *ex* that represents the *explicit external knowledge* available to the attacker.

Example 9 shows how the *pul* function of Example 8 can be derived from publicly available information. For the sake of brevity, the example considers user i_1 only.

Example 9 *Assume an attacker knows that i_1 works in an area that corresponds to $s_1 \cup s_2$ and that i_1 lives in an area that corresponds to s_3. This information can be computed by first obtaining the addresses from public databases, like voter lists or lists of employees and then mapping the addressed to the physical location. Moreover, assume that the request the attacker obtain is issued during working hours.*

The attacker knows that, on average, during working hours, people have $2/3$ of probabilities of being at work and $1/4$ of probability of being at home.

If i_1 is at work, then he can be either in s_1 of in s_2. Since the attacker has no knowledge about users' distribution inside s_1 or s_2, he can only assume an uniform distribution. Hence, assuming s_1 and s_2 have the same size, if i_1 is at work, then she has $1/2$ of possibility of being in s_1 or s_2. Clearly, since i_1 has $2/3$ of possibilities of being at work, the probability of being in

s_1 is equal to the probability of being in s_2 that is equal to $1/3$. *Hence,* $pul(i_1, s_1) = pul(i_1, s_2) = 1/3$.

Since i_1 has $1/4$ *of probability of being at home,* $pul(i_1, s_3) = 1/4$.

If i_1 is not at work nor at home ($1/12$ *of probability), she can be in s_4 or s_5. Since s_5 is twice as big as s_4, it is twice more likely that i_1 is located in s_5 than in s_4. Consequently,* $pul(i_1, s_5) = 1/12 \cdot 2/3 = 1/18$ *and* $pul(i_1, s4) = 1/12 \cdot 1/3 = 1/36$.

Formally, we assume the *explicit knowledge* to be defined by the ex function.

Definition 13 *We define the* explicit knowledge *as a partial function* $ex : I \times 2^{\perp_S} \rightarrow [0, 1]$ *such that, for each* $i \in I$, *given* $def_i = \{A \subseteq A_{tot} | ex(i, A)$ *is defined*$\}$, $\bigcap_{A \in def_i} A = \emptyset$ *and* $\sum_{A \in def_i} ex(i, A) \leq 1$.

Intuitively, for each user, ex is a partial function that associates some non overlapping areas with the probability of the user being located in these areas.

Definition 14 shows how to extend the partial function ex to the complete function pul.

Definition 14 *Given the explicit knowledge function* ex, $def_i = \{A \subseteq A_{tot} | ex(i, A)$ *is defined*$\}$ *and* $undef_i = A_{tot} \setminus \bigcup_{A \in def_i} A$, *for all* $s \in \perp_S$ *and* $i \in I$, *the possible user location function* $pul(i, s)$ corresponding *to ex is:*

$$pul(i, s) = \begin{cases} ex(i, A) \cdot \frac{Area(s)}{Area(A)} & \text{if } \exists A \in def_i \text{ s.t. } s \in A \\ (1 - \sum_{A \in def_i} ex(i, A)) \cdot \frac{Area(s)}{Area(undef_i)} & \text{otherwise} \end{cases}$$

Intuitively, in the regions A where the explicit knowledge is provided, the probabilities are uniformly distributed among the spatial granules whose union yields A. Where the explicit knowledge is not provided, the propabilities that has not been assigned (if any) are uniformly distributed.

3.4 Empirical Evaluation of Generalization Algorithms

In this section we show the results of the experiments we performed for the defense algorithms in contexts C_{st} and C_{st+g}. We leave as a future work the empirical evaluation of the *PartitionArea* defense algorithm.

Since, in most of previous works [24, 39, 29] the degree of anonymity k was used to evaluate the anonymity of a request, in this section we use this parameter instead of the re-identification threshold h. However, we recall that $h = 1/k$.

3.4.1 Experimental setting

We performed an extensive experimental evaluation of the defense algorithms against $UAtt_{C_{st}}$ and $Att_{C_{st+g}}$. We used two synthetic datasets; In both cases, the total area of the considered map is about 100 km^2 and the maximum number of users is $500,000$. In the first dataset, users are uniformly distributed, while in the other, users' locations are generated by the moving object generator developed by Brinkhoff [16] that was set to generate users' locations in the streets of the city of San Francisco. In the experiments with $500,000$ users, the average density of users for km^2 is about $5,000$ which is in the same range of the real density of population in that area. We identified two main parameters for each test: the degree of anonymity k (i.e., the inverse of the re-identification threshold), and the total number p of users.

We implemented the algorithms using Java, and we performed our tests on a Linux machine with two 2,4Ghz Pentium Xeon processors and 4GB of shared RAM. All the output values presented in this section are obtained by running $1,000$ tests and taking the average.

In order to compare the regions returned by the generalization algorithms with the smallest possible generalized region, we implemented the

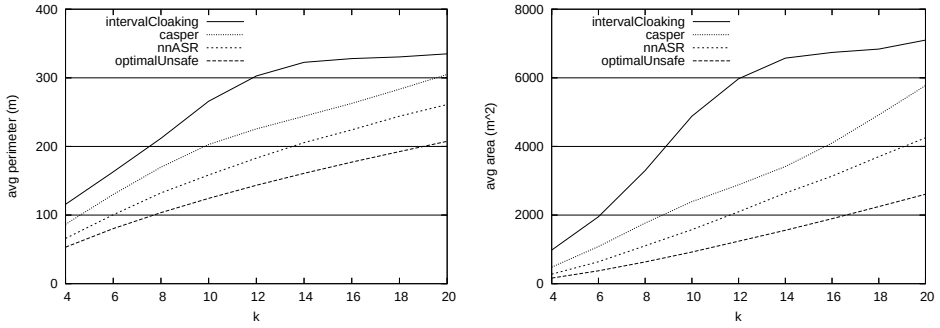
optimalUnsafe generalization algorithm[5]. This algorithm computes the set of $k-1$ users such that the perimeter of the MBR including these users and the issuer is minimal. The idea of *optimalUnsafe* is to search the best perimeter of the MBRs for each set containing the issuer and other $k-1$ users. Hence, the complexity of the algorithm is exponential in the number of users p; However, we developed several optimization techniques that make the algorithm in most cases computable in time linear in the size of p, and exponential in the size of k. This makes it possible to compute the optimal perimeter, as a reference value for the evaluation of the defense algorithms, for quite large values of p and practically relevant values of k. The algorithm we implemented computes, among all possible anonymity sets, the one having the smallest perimeter of the *MBR*. The choice of computing the optimal perimeter, instead of the optimal area, is driven by the fact that if optimal area is computed then the algorithm often returns a set of users having the same x value (or y value) but that are possibly far from each other.

3.4.2 Evaluation of the quality of service

Figure 3.5(a) and 3.5(b) show the average perimeter and area, respectively, of the region returned by four defense algorithms against $UAtt_{C_{st}}$ for different values of k. The principle behind the *nnASR* algorithm may induce the reader to think that the resulting region is minimal. Our empirical results show that this is not the case. On average, *nnASR* returns regions having a perimeter 25% larger than the one of the region returned by *optimalUnsafe*.

We also computed the average number of times in which *nnASR* returns the same result as *optimalUnsafe*. We noticed that this value rapidly decreases with the growing of k. For example, with $k = 4$ and $p = 500,000$, *nnASR* returns the region with the minimal perimeter in about 26% of the cases, while the percentage drops below 2% for $k = 20$ and the same number of users.

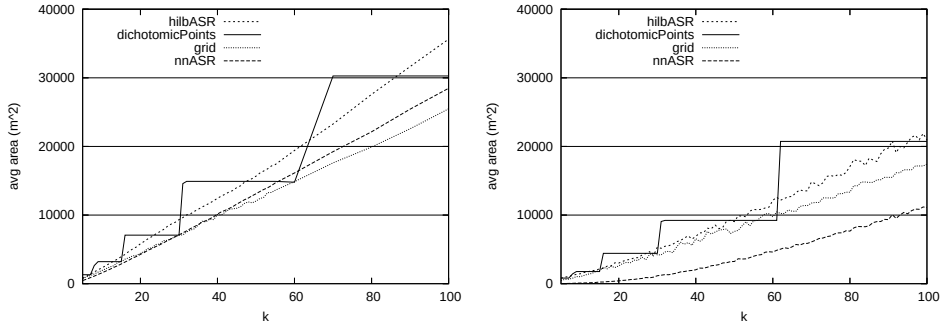
[5]The term "unsafe" in the name *optimalUnsafe* refers to the fact that the algorithm is a defense only against $UAtt_{C_{st}}$ and not against $Att_{C_{st+g}}$.



(a) perimeter (b) area

Figure 3.5: Average size of the generalized region with $p = 500,000$ for the defense algorithms against $UAtt_{C_{st}}$ in the non-uniform distribution.

Figures 3.6(a) and 3.6(b) consider three defense algorithms against $Att_{C_{st+g}}$ and one defense algorithm against $UAtt_{C_{st}}$, to compare with. They show the average area of the regions returned by these algorithms for different values of k, in both uniform and non-uniform cases. First, we can notice that, in the non-uniform distribution, the algorithms perform better. This is due to the fact that users are located in the streets only and therefore they tend to be closer to each other. We can also observe that in both cases the *grid* algorithm performs significantly better than the other two defense algorithms against $Att_{C_{st+g}}$. Comparing *grid* and *nnASR* we can notice that, in the uniform distribution, they perform similarly for values of k up to 45, while for higher values of this parameter, *grid* performs slightly better. On the contrary, in the non-uniform case, *nnASR* performs significantly better for every value of k. Indeed, in the non-uniform case it is possible that projecting the positions of a given set of users on one of the axis, some successive values on the axis happen to be far apart from each other. Due to its partitioning strategy, *grid* may have to include the corresponding positions in the same block, while *nnASR*, for the same set of users may be able to include only positions with values on that axis close to each other. This clearly does not happen in the uniform case, since the



(a) uniform distribution

(b) non-uniform distribution

Figure 3.6: Average area with $p = 500,000$ for the defense algorithms against $Att_{C_{st+g}}$.

values in the projection will always have a similar distance. Finally, we can observe that the growth of the size of the average area obtained using *hilbASR*, *grid* and *nnASR* is almost linear in k, while the growth obtained using *dichotomicPoints* is constant for certain intervals of k, with changes of values for some values of k. This is due to the fact that *dichotomicPoints* partitions the number of points until it finds a set containing less than k users. The number of iterations is given by: $\lceil \log(\frac{p}{k}) \rceil$. Therefore, there are executions of the algorithm with different values of k that iterate the same number of times, hence computing, at the last iteration, the same number of users. Consequently, these executions return regions with similar area.

In order to better illustrate this behavior, we show in Figure 3.7 the average area of the *MBR* for high values of k (up to 400). It can be noticed that *hilbASR* does not scale well for high values of k. Though this values of k may be assumed too high for some applications, in Chapter 4 we show how, in the dynamic case, it is necessary to compute some of the algorithms presented in this section for high values of k.

In Figure 3.8 we fix the value of k to 40 and we show the average area of the generalized region computed by the four algorithms with respect to the considered number of users. As expected, the average area decreases for

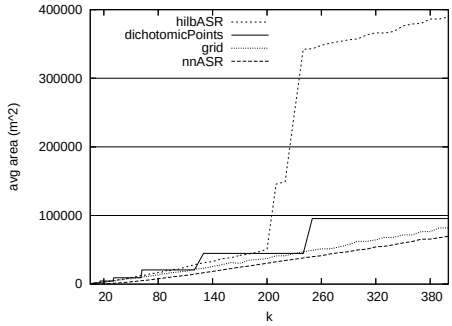


Figure 3.7: Average area with $p = 500,000$ for the non-uniform distribution and for large values of k.

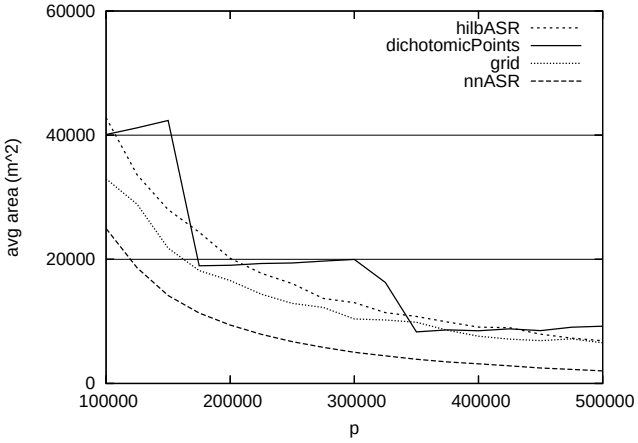


Figure 3.8: Average area with $k = 40$ for the non-uniform distribution.

a growing value of p. A slightly different behavior can be noticed for the *dichotomicPoints* algorithm: since, as we observed before, the number of divisions performed by *dichotomicPoints* is equal to $\lceil \log(\frac{p}{k}) \rceil$, if we increase the value of p, keeping the same number of divisions, we will have larger anonymity sets, hence larger areas of their *MBR*.

In Figure 3.9 we show the variance of the area of the *MBR* of the generalized regions computed by the *dichotomicPoints*, *hilbASR*, *grid* and *nnASR* algorithms. The growth of the *hilbASR* variance is not regular. We have observed that, in some cases, the execution of the *hilbASR* algorithm in the non-uniform distribution can result in an area that is up to 30 times larger

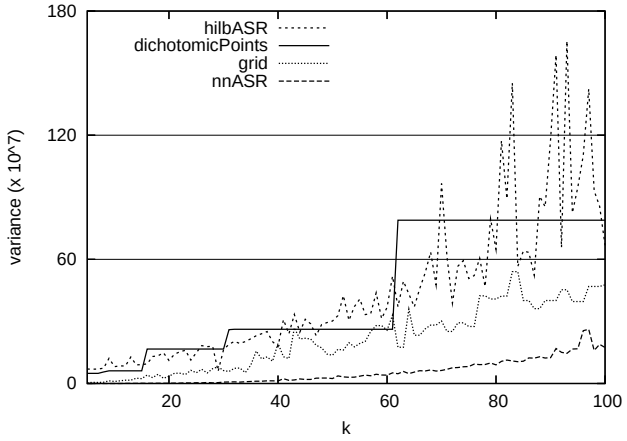
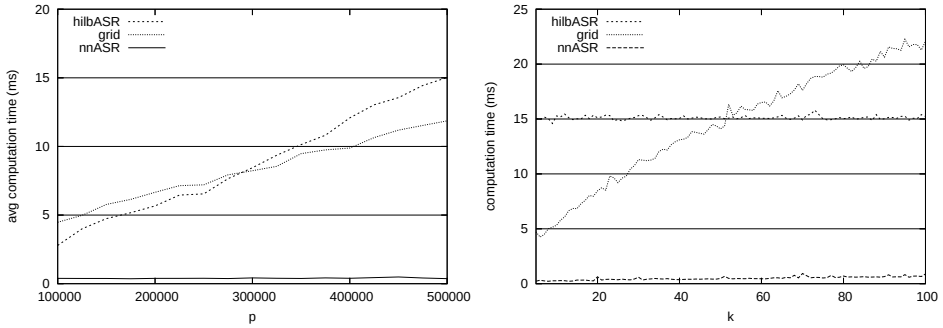


Figure 3.9: Variance of the algorithms with $p = 500,000$ for the non-uniform distribution.

than the average one. On the other hand, in most of the cases the algorithm generates areas smaller than the average one. The *dichotomicPoints* algorithm has a regular but still high variance, while *grid* algorithm has a smaller variance. As expected, the *nnASR* algorithm has a much more regular and a definitely better variance.

3.4.3 Performance evaluation

Figures 3.10(a) and 3.10(b) show the average computation time of the algorithms *nnASR*, *hilbASR* and *grid*. In the former figure, the value of k is fixed to 40 and the value of p varies from a minimum of $100,000$ to a maximum of $500,000$. In the latter, p is fixed to $500,000$ and k varies from 5 to 100. The results for the *dichotomicPoints* are not reported here, since this algorithm has a computation time about 50 times higher with respect to the other algorithm reported in the two figures. Indeed, for $k = 40$ and $p = 500,000$ the computation time of *dichotomicPoints* is about half a second. We can notice that the computation time of the *nnASR* algorithm is about one millisecond and we did not measure significant variation in this value for different values of k of p. The *hilbASR* and *grid* algorithms have similar computation time. As expected from the complexity analysis, the



(a) $k = 40$, varying p (b) $p = 500,000$, varying k

Figure 3.10: Average computation time for the non-uniform distribution.

computational time of both algorithms is affected by the number of users. On the contrary, we can observe that the computational time of *hilbASR* is not affected by the value of k while the computational time of *grid* increases linearly with the value of k. Again, this experimental finding is consistent with the worst case time complexity analysis of the two algorithms.

Chapter 4

Techniques to enforce anonymity in the dynamic case

4.1 Requests's linking

In the dynamic case, the attacker is able to understand that a set of requests is issued by the same user. We say that a set of requests is *linked* to a request r', if in a given context it is possible for an attacker to understand that all the requests in the set, including r', are issued by the same user. The set of requests that are linked to a request is called a *trace*.

The ability of the attacker to link requests is modeled as part of the context. Indeed a context C specifies how the attacker can compute the linking function L_C that associates, to each generalized request r', the set $L_C(r')$ of all the requests linked to r' under C. When a context C provides no information about how to compute the linking function, we assume that requests cannot be linked. This model makes it possible to provide a formal characterization of the intuitive notion of static and dynamic cases we used so far. Indeed, in the static case, each generalized request can be linked with no other requests but itself.

Definition 15 *Given a context C, we say that C is* a context in the static case *if for each generalized request r' , $L_C(r') = \{r'\}$. Otherwise we say that C is* a context in the dynamic case.

In general, the combination of data in different fields of LBS requests can lead an attacker to link a set of requests to a specific one. Consequently, different contexts can be defined to specify how the attacker can trace user's requests. However, in the following of this dissertation we concentrate on the context C_{pid} in which the attacker is able to link each generalized request with the requests issued with the same pseudo-identifier (pid). The reasons for this choice are:

- Service personalization is recognized as an important characteristic in internet services [31]. We argue that service personalization is even more important in location based services. In order to implement service personalization, SPs require a pid to be specified in each request. Indeed, these SPSS store a user profile that is explicitly provided by the user and/or derived by the SP. The pid is required to associate the issuer of a request to the correct user profile.

- The details on how to compute linking using spatio-temporal information (see, e.g., [25]) or service specific information are out of the scope of this dissertation. In context C_{pid} it is trivial to compute the linking function and to perform the *unlink* operation i.e., to make a request not linkable to any previous one. Indeed, a request is linked with all the requests already issued with the same pid and the LTS can unlink a request by using a pid that was never used in any previous request.

- Context C_{pid} can be easily modified to model the most conservative case i.e., a generalized request r' can be linked with the set of all requests issued by $issuer(r')$. Indeed, if we assume that the LTS uses a single pid for each user (and cannot change it), then each request can be linked with all the requests issued by the same user.

Formally, in context C_{pid} the following linking function can be defined:

Definition 16 *Let C_{pid} be the context in which the attacker knows that each pseudo-identifier is used by at most one user. In context C_{pid}, for each generalized request r', the attacker can compute the following linking function:*

$$L_{C_{pid}}(r') = \begin{cases} \{r'\} & \text{if } r'.IDdata = \textbf{\textit{null}} \\ \{r'' \in R' | r''.IDdata = r'.IDdata\} & \text{otherwise} \end{cases}$$

Note that, in context C_{pid} the attacker 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 it allows the attacker to perform attacks that are much more likely to identify the issuer of the request than the corresponding ones in the static case.

An important aspect of the C_{pid} context is that the unlinking operation (i.e., the change of the pid) decreases the quality of service because it prevents the SP from providing a personalized service. Therefore, unlinking should only be used when the generalization (without unlinking) would produce a generalized region larger than a given threshold S_{max}. Intuitively, S_{max} is a parameter that represents the maximum size of the generalized region above which the service becomes meaningless and cannot be provided.

4.2 Context C_{st+pid}: a model for spatio-temporal anonymity with linking

4.2.1 Context definition

In this section we consider context C_{st+pid} in which the attacker knows the location of each user in each time instant and is able to link requests issued with the same pid. In this context, given a trace of requests, the set of possible issuers is given by the users that can have possibly issued all the requests in the trace. In other words, the anonymity set of each generalized

request r' is given by the set of users i such that, for each request r'' linked with r', the location of i at time $r''.Tdata$ is inside $r''.Sdata$. Definition 17 formalizes this concept.

Definition 17 *Let C_{st+pid} be the context in which the attacker has the knowledge and reasoning abilities of contexts C_{st} and C_{pid}. For each generalized request r' the anonymity set based on context C_{st+pid} is:*

$$AS_{C_{st+pid}}(r') = \bigcap_{r'' \in L_{C_{pid}}(r')} AS_{C_{st}}(r'')$$

Note that Definition 17 is a proper extension of Definition 8. Indeed, if no linking is possible (e.g., if the LTS always performs unlinking), then for each generalized request r', $L_{C_{pid}}(r') = \{r'\}$. Therefore:

$$AS_{C_{st+pid}}(r') = \bigcap_{r'' \in L_{C_{pid}}(r')} AS_{C_{st}}(r'') = AS_{C_{st}}(r')$$

4.2.2 *Greedy-nnASR* Generalization algorithm

The first algorithm we present is a greedy algorithm based on two main ideas:

- the same anonymity set AS should be maintained for the requests in the trace;
- the anonymity set is computed with the $nnASR$ algorithm (see Section 3.1.2) that is a defense algorithm with respect to $UAtt_{C_{st}}$.

Algorithm 5 takes in input the value of the spatial threshold S_{max} in addition to the original request r, the re-identification threshold h and the set of previous requests τ. The algorithm first checks if r is the first request issued by $issuer(r)$. If so, a request generalized with the $nnASR$ algorithm is returned. Otherwise, *Greedy-nnASR* computes the anonymity set AS of the first request issued with the last used pid and then assigns to the variable A the value of the minimum bounding rectangle that contains the locations of the users in AS at time $r'.Tdata$. Finally, if A has area smaller than

S_{max} then the request with A as generalized region is returned. Otherwise, unlinking is required and the generalized request is computed through the $nnASR$ algorithm.

Algorithm 5 *Greedy-nnASR*

Input: an original request r, a value $h \in (0, 1]$, the set τ of generalized requests issued by $issuer(r)$, a value S_{max}.
Output: a generalized request that is safe against $UAtt_{C_{st+pid}}$ with re-identification threshold h.
Method:

1: **if** $(\tau = \emptyset)$ **then**
2: **return** $nnASR(r, h)$
3: **else**
4: $r'' =$ the last request in temporal order in τ
5: $p = r''.IDdata$ {the pid currently in use}
6: $r''' =$ the first request in temporal order in τ s.t. $IDdata = p$
7: $AS = AS_{C_{st}}(r''')$ {the anonymity set of r'''}
8: $A = MBR(AS, r.Tdata)$
9: **if** $(Area(A) \leq S_{max})$ **then**
10: **return** $finalizeGeneralization(r, A)$
11: **else**
12: perform unlinking
13: **return** $nnASR(r, h)$
14: **end if**
15: **end if**

Intuitively, the idea of *Greedy-nnASR* is that, if a new pid is used (at the first request or after unlinking), the generalized request is computed with the generalization algorithm $nnASR$ that was designed for the static case. Otherwise, the generalized region is computed as the area that contains the current locations of the users in the anonymity set of the first request in the trace. In practice, the algorithm computes an anonymity set for the first requests and then keep the same anonymity set as long as possible, until unlinking is required because the area that include the current locations of the users is larger than S_{max}.

Intuitively, the correctness of the *Greedy-nnASR* algorithm derives from the fact that the first request in a trace is safe, since it is computed by a defense function against $UAtt_{C_{st}}$ and the following requests in the trace

are also safe because their generalized areas cover the locations of the same users in the anonymity set of the first request in the trace.

Theorem 6 Greedy-nnASR *is a defense algorithm against* $UAtt_{C_{st+pid}}$.

For what concerns the complexity of the *Greedy-nnASR* algorithm, it is equal to the time required to compute the minimum bounding rectangle of the users in the anonymity set plus the time required to compute $nnASR$. In our implementation, we did not use any order to store the users in the anonymity set, hence the minimum bounding rectangle can be computed in linear time with respect to the cardinality of the anonymity set that is $\lceil 1/h \rceil$[1]. On the other hand, $nnASR$ can be computed in logarithmic expected time with respect to the number of users. Therefore, in theory, the algorithm can be computed in time linear in $1/h$. However, in practice $|I| >> 1/h$, and hence the time required to compute the algorithm is dominated by the time required to compute the $nnASR$.

4.2.3 *Square* Generalization algorithm

We say that *Greedy-nnASR* is a greedy algorithm because it provides good performace, in terms of the size of the generalized region, for the first request in a trace while, for the following requests in the trace, the size of the generalized region rapidly increases because the users in the anonymity set are likely to move in different directions. In Section 4.5 we show that, in our experimental setting, after few requests, the area of the generalized region is larger than S_{max} and hence unlinking is required.

In this section we present the *Square* generalization algorithm that tackles this problem by using a variable anonymity set. The intuition is that, for the first request in the trace, the *Square* algorithm computes an anonymity set that is larger than the one strictly required to generalize that request.

[1]MBR can be computed in constant time if the users are ordered according to both axis; However, to keep the set of users ordered implies an overhead in the management of the data structure each time a user moves.

When successive requests arrive, the anonymity set is computed as a subset of the anonymity set of the previous request in the trace. When the anonymity set contains less than $1/h$ users, unlinking is performed and a new anonymity set is computed.

Algorithm 6 first checks if the input request r is the first request issued by $issuer(r)$. If so, the set I' of users is set to I. Otherwise, I' is set to the anonymity set of the last generalized request issued by $issuer(r)$. Then, the algorithm selects the set AS of users of I', who are called *candidate pseudoissuers*, whose locations are in the square that is centered in the issuer's location and has area equal to S_{max}. (In fact, each square mentioned in this subsection is of size S_{max}.) A user, called *pseudoissuer*, is randomly chosen from this set of candidate pseudoissuers. The variable AS is reassigned to the set of users of I' whose location is in the square centered in the location of the *pseudoissuer*. If AS has cardinality not smaller than $1/h$, then it is safe to return a generalized request with a generalized region corresponding to the MBR of the locations of the users in AS at time t. Otherwise, unlinking is required. First, AS is reassigned to the set of users of I whose location is in the square centered in the location of the *pseudoissuer*. If now the cardinality of AS is not smaller than $1/h$, then the MBR of the locations of the users in AS at time t can be computed as generalized region and the generalized request can be returned after that unlinking is performed. Otherwise, if the cardinality of AS is still smaller than $1/h$, then the *Square* algorithm is able to compute a safe request and therefore r_{null} is returned.

The correctness of the *Square* algorithm relies on the fact that, when the algorithm is not capable of computing a safe request, then it suppresses the request. However, note that requests are suppressed only when a generalization is not possible even after the unlinking is performed. This is a very unlikely case, as we shall see in Section 4.5.

Theorem 7 Square *is a defense algorithm against* $UAtt_{C_{st+pid}}$.

For what concerns the computational complexity of the *Square* algorithm, the only non constant operation is the computation of the set of

Algorithm 6 *Square*

Input: an original request r, a value $h \in (0, 1]$, the set τ of generalized requests issued by $issuer(r)$, a value S_{max}.
Output: a generalized request that is safe against $UAtt_{C_{st+pid}}$ with re-identification threshold h.
Method:

1: **if** $(\tau = \emptyset)$ **then**
2: $I' = I$
3: **else**
4: $r'' =$ the last request in temporal order in τ
5: $I' = AS_{C_{st+pid}}(r'')$
6: **end if**
7: $t = r.Tdata$
8: $s =$ the square with edges parallel to the axis, with area equal to S_{max}, that is centered in the position of the issuer of r at time t
9: $AS =$ the set of users in I' whose location at time t is inside s
10: $pi =$ a random user in AS {pseudo issuer}
11: $s =$ the square with edges parallel to the axis, with area equal to S_{max}, centered in the position of pi at time t
12: $AS =$ the set of users in I' whose location is inside s
13: **if** $(|AS| \geq 1/h)$ **then**
14: **return** $finalizeGeneralization(r, MBR(AS, t))$
15: **else**
16: $AS =$ the set of users in I whose location is inside s
17: **if** $(|AS| \geq 1/h)$ **then**
18: perform unlinking
19: **return** $finalizeGeneralization(r, MBR(AS, t))$
20: **else**
21: **return** r_{null}
22: **end if**
23: **end if**

users of I' whose location is inside s. In the worst case, I' is equal to I and all the users are located in s, hence this operation has complexity $O(|I|)$. However, in practical cases, since $Area(s) << Area(A_{tot})$, the cardinality of the set of users located in s is much smaller than $|I|$. Consequently, if an R-tree is used to store users' location, the time required to compute this operation is logarithmic in $|I|$.

4.3 Context $C_{st+g+pid}$: a model for spatio-temporal anonymity with linking when the generalization function is known to the attacker

4.3.1 Context definition

In this section we consider context $C_{st+g+pid}$, the combination of context C_{st+g} and C_{pid}. In this context, the probability that a user i issues a trace of requests is given by the probability that there exists a set of potential original requests issued by i that is generalized to that trace. Formally, we can define the following attack

Definition 18 *Let $C_{st+g+pid}$ be the context in which the attacker has the knowledge and reasoning abilities of context C_{st+g} and of context C_{pid}.*

The attack based on context $C_{st+g+pid}$ is:

$$Att_{C_{st+g+pid}}(r',i) = \frac{\prod_{\forall r'' \in L_{C_{pid}}(r')} P[g(r''_i, \tau^{r'}_{r''}) = r'']}{\sum_{j \in I} \prod_{\forall r'' \in L_{C_{pid}}(r')} P[g(r''_j, \tau^{r'}_{r''}) = r'']} \tag{4.1}$$

where, for each $i \in I$ and each $r'' \in R'$, $r''_i = o(r'', i, loc_i(r''.Tdata))$ and $\tau^{r'}_{r''}$ is the set of requests of $L_{C_{pid}}(r')$ issued before r''.

Definition 18 may appear involved, but the intuition behind it is actually simple. The numerator of Equation (4.1) is the absolute probability that i issues the trace $L_{C_{pid}}(r')$. Indeed, this probability is given by the probability that all the original requests "derived" by the requests in $L_{C_{pid}}(r')$ through the $o()$ function are generalized to the requests $L_{C_{pid}}(r')$. Formally, this probability is: $P_{\forall r'' \in L_{C_{pid}}(r')}[g(r''_i, \tau^{r''}_i) = r'']$. Since the probabilities, for each $r'' \in L_{C_{pid}}(r')$ are all independent, they are all verified with the same probability of their product, i.e., the numerator of Equation (4.1). The denominator of Equation (4.1) simply normalizes the absolute probability to 1, among the users in I.

Now, let us consider the particular case of a deterministic generalization function. In this case, $\prod_{\forall r'' \in L_{C_{pid}}(r')} P[g(r''_i, \tau^{r'}_{r''}) = r'']$ is equal to 1 if the

generalization of each original request r''_i with previous requests $\tau^{r'}_{r''}$ is equal to r'', otherwise it is equal to 0. Hence, in this case, the attack is uniform. Indeed, a user i is not a possible issuer if there exists a request r'' in $L_{C_{pid}}(r')$ such that $g(r''_i, \tau^{r'}_{r''}) \neq r''$. Otherwise, i is a possible issuer and the number of possible issuers (each one with the same probability of being the real issuer) is given by the number of users j such that, for all $r'' \in L_{C_{pid}}(r')$, $g(r''_j, \tau^{r'}_{r''}) = r''$.

Theorem 8 *If g is a deterministic generalization function then $Att_{C_{st+g+pid}}$ is a uniform attack characterized by the anonymity set*

$$AS_{C_{st+g+pid}}(r') = \{i \in I | \forall r'' \in L_{C_{pid}}(r')\ g(r''_i, \tau^{r'}_{r''}) = r''\}$$

where, for each $i \in I$ and each $r'' \in R'$, $r''_i = o(r'', i, loc_i(r''.Tdata))$ and $\tau^{r'}_{r''}$ is the set of requests of $L_{C_{pid}}(r')$ issued before r''.

4.3.2 *Provident-hilb* generalization algorithm

In this section we present a defense algorithm against $Att_{c_{st+g+pid}}$ called *ProvidentHilb* (Algorithm 7). Similarly to the *Square* algorithm, $ProvidentHilb$ computes, for the first request in a trace, an anonymity set that is larger than the one strictly required to generalize that request. When successive requests arrive, the anonymity set is computed as a subset of the anonymity set of the previous request in the trace.

The *ProvidentHilb* algorithm differs from the *Square* algorithm in the computation of the anonymity set. Given r the original request given in input, the *ProvidentHilb* algorithm computes the anonymity set as a subset of the set I' that is initialized to the anonymity set of the previous request issued by $issuer(r)$. If r is the first request issued by $issuer(r)$, I' is initialized to I. The set I' is then partitioned into blocks using the *HilbPart* procedure. Unlinking is required if any of the blocks in the partition contains a set of users whose locations at time $r.Tdata$ is bounded by a minimal rectangle with area larger than S_{max}. In this case, *HilbPart* function is ex-

ecuted again to partition the set I of all users. Finally, the anonymity set computed as the block that contains the issuer of r.

Algorithm 7 *ProvidentHilb*

Input: an original request r, a value $h \in (0, 1]$, the set τ of generalized requests issued by $issuer(r)$, a value S_{max}.
Output: a generalized request that is safe against $UAtt_{C_{st+g+pid}}$ with re-identification threshold h.
Method:

1: **if** $(\tau = \emptyset)$ **then**
2: $I' = I$
3: **else**
4: $r'' =$ the last request in temporal order of τ
5: $I' = AS_{C_{st+g+pid}}(r'')$ {The anonymity set of r''}
6: **end if**
7: $t = r.Tdata$
8: $part = HilbPart(I', h, S_{max}, t)$
9: **if** $(\exists B \in part$ s.t. $Area(MBR(B, t)) > S_{max})$ **then**
10: perform unlink
11: $part = HilbPart(I, h, S_{max}, t)$
12: **end if**
13: B is the block of $part$ s.t. $issuer(r) \in B$
14: **return** $finalizeGeneralization(r, MBR(B, t))$

The computation of *HilbPart*, shown in Algorithm 8, is inspired by the *HilbASR* algorithm [29]. Users are totally ordered according to their locations (at time $r.Tdata$) along the Hilbert space filling curve. Then, the blocks are constructed considering the users one by one (Lines 4 to 11). 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 S_{max}. However, each block should also contain at least k users (with $k = \lceil 1/h \rceil$) and hence a user is always added to a block that contains less than k users (Line 6).

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 14 to 27). 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 23 to 26). 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, then the algorithm is trying to use too many blocks ($|part| \cdot k > |I'|$). In this case, the first block is deleted and its users are added to the second one.

Theorem 9 proves that *ProvidentHilb* is a defense algorithm against $Att_{C_{st+g+pid}}$.

Theorem 9 ProvidentHilb *is a defense algorithm against* $Att_{C_{st+g+pid}}$.

4.4 Context $C_{ast+g+pid}$: modeling approximate knowledge of users' locations with linking

4.4.1 Context definition

In this section we present the extension of context C_{ast+g} to the dynamic case $C_{ast+g+pid}$. Analogously to how the attacks with exact knowledge in the dynamic case extends the corresponding attacks in the static case, the attack $Att_{C_{ast+g+pid}}(r', i)$ can be simply defined as the product, for each request linked with r' of the attacks in the static case.

However, in the dynamic case, we want to model the case in which the possible user locations function can change during time. The intuition is that, if a user has a given probability of being located in a certain region during, for example, working hours, the same user may have different probability of being located in the same region during non-working hours.

To formally model this situation, we define a *temporal bottom granularity*. Temporal granularities and their relationships have been extensively studied, among others, in [11, 42, 9]. In the following we report the first definition of temporal granularity provided in [11]

Algorithm 8 *HilbPart*

Input: a set I' of users, a value $h \in (0, 1]$, a value S_{max}, a time instant t
Output: a partition *part* of I' such that each block of part contains at least $1/h$ users
Method:

1: $k = \lceil 1/h \rceil$
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** $(Area(MBR(B \cup \{u\}, t)) \leq S_{max}$ OR $|B| < k)$ **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**
17: $B_{second} =$ the second block in *part*
18: $B_{second} = B_{second} \cup B$
19: $part = part \setminus \{B\}$
20: **break**
21: **else**
22: B_{prev} is the block of *part* preceding B
23: $U =$ last, according to Hilbert order, $k - |B|$ users in B_{prev}
24: $B = B \cup U$
25: $B_{prev} = B_{prev} \setminus U$
26: **end if**
27: **end for**
28: **return** *part*

Definition 19 *A* temporal granularity *is a mapping G from the integers (the* index set*) to subsets of the time domain such that: (1) if $i < j$ and $G(i)$, $G(j)$ are nonempty, then each element of $G(i)$ is less than all elements of $G(j)$, and (2) if $i < k < j$ and $G(i)$, $G(j)$ are nonempty, then $G(k)$ is nonempty.*

Intuitively, a temporal granularity G associates to each integer t a (pos-

sibly empty) subset of the time domain such that (1) the indexes order is the same as the time domain order and (2) the subset of indexes that maps to nonempty subsets of the time domain is contiguous.

Several relationships between temporal granularities are defined in [11]. In the following, we use the *groups into* relationship formalized in Definition 20.

Definition 20 *A temporal granularity G* groups into *a temporal granularity H, denoted by $G \trianglelefteq H$, if for each index j, there exists a possibly infinite subset S of the index set such that $H(j) = \bigcup_{i \in S} G(i)$.*

Based on the definition of the groups into relationship, we define the *temporal bottom granularity* $\bot_T$ as the granularity that groups into all the other granularities in the system. Intuitively, this implies that each granule of each granularity in the system can be specified as the union of a set of granules of the bottom granularity.

Once the temporal bottom granularity is defined, we can extend Definition 11 to consider the temporal dimension.

Definition 21 *Given a set of users I, the bottom spatial granularity $\bot_S$ and the bottom temporal granularity $\bot_T$, the* possible user locations function *$pul : I \times \bot_S \times \mathbb{Z} \rightarrow [0, 1]$ is such that for each user $i \in I$ and each $t \in \mathbb{Z}$, $\sum_{s \in \bot_S} pul(i, s, t) = 1$.*

In Definition 21, the third paramether of the *pul* function represents the index of the granule of the temporal bottom granularity in which the distribution of user locations is considered.

Similarly to the static case, the definition of the possible user locations function can be extended to take an area A as paramether:

$$\forall i \in I, t \in \mathbb{Z}, pul(i, A, t) = \sum_{s \in A} pul(i, s, t)$$

The probability that a single user located in an area A has identity i, defined as $p(i, A)$ in Section 3.3.1, can also be easily extended to consider

the temporal domain. Formally, given $enu(A,t) = \sum_{i \in I} pul(i,A,t)$,

$$p(i,A,t) = \begin{cases} 0 & \text{if } enu(A,t) = 0 \\ \frac{pul(i,A,t)}{enu(A,t)} & \text{otherwise} \end{cases}$$

We can now define the attack based on context $C_{ast+g+pid}$:

Definition 22 *Let $C_{ast+g+pid}$ be the context in which the attacker has knowledge and reasoning abilities of contexts C_{ast+g} and C_{pid}. For each $r' \in R'$ and $i \in I$, the attack $Att_{C_{ast+g+pid}}(r', i)$ is:*

$$\frac{\prod_{r'' \in L_{pid}(r')} \sum_{s \in r''.Sdata} (p(i,s,r''.Tdata) \cdot P[g(o(r'',i,s)) = r''])}{\sum_{j \in I} \prod_{r'' \in L_{pid}(r')} \sum_{s \in r''.Sdata} (p(j,s,r''.Tdata) \cdot P[g(o(r'',j,s)) = r''])}$$

Intuitively, $Att_{C_{ast+g+pid}}(r', i)$ is the product, for each request linked with r', of the attacks in the static case, normalized to 1 for the users in I. However, the notation of $Att_{C_{ast+g}}$ cannot be used since, this static attack, defined in Section 3.3.1, uses a definition of the p function that is independent from the time instant in which the request is issued.

4.4.2 A defense function

Algorithm 9 shows a generalization algorithm that extends algorithm *PartitionArea* presented in Section 3.3.2. The idea is to iteratively partition a $2D$-region A that is initialized to A_{tot}. At each iteration, a partitioning of A is computed. For each block B, if any user could be identified, using the combination of contexts C_{ast} and C_{pid}, as the issuer of a request having B as the generalized location, with likelihood greater than h, then the algorithm terminates returning a generalized request that has A as generalized location. Otherwise, iteration continues and the value of A is set to the block that contains the location of the issuer.

The most involved part of the algorithm consists in checking if a user could be identified as the issuer of a request having B as generalized location. Intuitively, under context $C_{ast+pid}$ (i.e., the combination of C_{ast} and pid), the probability of a user being the issuer of a request from an area B is given by the probability, normalized to 1 among the users in I, of the user

being the issuer of the previous requests in the trace times the probability of the user being the issuer of a request from the area B. Lines 6 to 8 of Algorithm 9 computes the maximum probability among all the users and all the blocks of the partitions of A.

Algorithm 9 *PartitionAreaDyn*

Input: an original request r, a value $h \in (0, 1]$, the set τ of generalized requests issued by $issuer(r)$, a value S_{max}.
Output: a generalized request that is safe against $UAtt_{C_{ast+g+pid}}$ with re-identification threshold h.
Method:

1: $A = A_{tot}$
2: $continue =$ **true**
3: $i = issuer(r)$
4: $t = r.Tdata$
5: **while** (continue) **do**
6: $par = partition(A)$
7: $max = \max_{j \in I, B \in par} \left(\prod_{r'' \in \tau} p(j, r''.Sdata, r''.Tdata)\right) \cdot p(j, B, t)$
8: $sum = \sum_{j \in I} \left(\prod_{r'' \in \tau} p(j, r''.Sdata, r''.Tdata)\right) \cdot p(j, B, t)$
9: **if** $(max/sum > h)$ **then**
10: $continue =$**false**
11: **else**
12: $A =$ the block B of par s.t. $r.Sdata \in B$
13: **end if**
14: **end while**
15: **if** $Area(B) > S_{max}$ **then**
16: **return** $PartitionArea(r, h)$
17: **else**
18: **return** $finalizeGeneralization(r, A)$
19: **end if**

We conjecture that *PartitionAreaDyn* is a defense function. The idea is similar to the idea of *PartitionArea*: if r' is the result of $PartitionAreaDyn(r, h, \tau, S_{max})$, the generalization of any request issued in $r'.Sdata$ with the same h, τ and S_{max} parameters is r'.

Conjecture 2 PartitionAreaDyn *is a defense algorithm against* $Att_{C_{ast+g+pid}}$.

4.4.3 Specification of the *pul* function in the dynamic case

In Section 3.3.3 we show how to specify the possible user location function in the static case. In this section we extend the technique to the dynamic case. The idea is that, if the *ex* function were defined for each time granule of the bottom temporal granularity, then the corresponding *pul* can be defined with a straightforward extension of Definition 14.

However, it is not reasonable to assume that an explicit knowledge function is defined for each granule of the bottom granularity. Instead, we assume that the explicit knowledge function is defined for some temporal granularities. These granularities are such that none of their granules overlap and the union of their granules covers the entire time domain.

In the following, we indicate with ex_G the explicit knowledge function associated to the granularity G in $\mathcal{G}$. The *pul* function in a given time instant t can then be defined from the ex_G function such that there exists a granule of G that covers $\perp_T(t)$.

Definition 23 *Let $\mathcal{G}$ be a set of temporal granularities such that: 1) for each distinct pairs of granularities $G_1, G_2 \in \mathcal{G}$ and for each pair of integers $i, j \in \mathbb{Z}$, $G_1(i) \cap G_2(j) = \emptyset$; 2) $\bigcup_{G\in\mathcal{G}} \bigcup_{i\in\mathbb{Z}} G(i) = \bigcup_{i\in\mathbb{Z}} \perp_T(i)$.*

Moreover, let E be a set that contains an explicit knowledge functions ex_G for each $G \in \mathcal{G}$.

Then, for all $s \in \perp_S$, $i \in I$, $t \in \mathbb{Z}$, the possible user location function $pul(i, s, t)$ corresponding *to E is:*

$$pul(i,s,t) = \begin{cases} ex_G(i,A) \cdot \frac{Area(s)}{Area(A)} & \text{if } \exists A \in def_i \text{ s.t. } s \in A \\ (1 - \sum_{A\in def_i} ex_G(i,A)) \cdot \frac{Area(s)}{Area(undef_i)} & \text{otherwise} \end{cases}$$

where G is the granularity in $\mathcal{G}$ such that $\exists j \in \mathbb{Z}$ with $G(j) \supseteq \perp_T(t)$; $def_i = \{A \subseteq A_{tot} | ex_G(i, A)$ is defined$\}$; and $undef_i = A_{tot} \setminus \bigcup_{A\in def_i} A$.

4.5 Empirical evaluation of generalization algorithms

In this section we show the results of the experiments we performed for the defense algorithms in contexts C_{st+pid} and $C_{st+g+pid}$. The aim of the

experiments is to evaluate, for each of the proposed defense algorithms, the average length of the traces i.e., the average number of requests that can be issued before unlinking is required. We leave as a future work the empirical evaluation of the *PartitionAreaDyn* defense algorithm.

4.5.1 Experimental setting

Similarly to the experiments in the static case, the tests presented in this section were performed using artificial data. Users' locations were generated by the moving object generator developed by Brinkhoff [16] that was set to create $500,000$ user's traces in the city of San Francisco. The total area of the map is about 100 km^2. The resulting average density of users for km^2 is about $5,000$. All users start moving at time instant 1 from a randomly chosen location towards a randomly chosen destination. Users communicate their location every minute. In our tests, users' movement are monitored for a period of 40 minutes. When a user reaches her destination, she stops moving, but carries on providing her location.

Users are equally divided into two groups: pedestrians and cars. The speed limit for pedestrians is set to 4 km/h, while the speed limit for cars is 100 km/h. User's speed may also be limited by the street where the user is passing by; There are three kinds of streets, with average speed limits 100 km/h (highways), 40 km/h (main roads), 20 km/h (urban streets). Our tests are performed considering, as issuers, 100 pedestrians and 100 cars, randomly chosen among the users that do not reach their destination before the $30th$ minute. We assume that each of these users issues a request every minute.

4.5.2 Evaluation of the trace length

In the first set of experiments, we fix the value of k to 10 and evaluate the average length of the traces for different values of S_{max}. Three different values of S_{max} are chosen: 10^3 m^2, 10^4 m^2 and 10^5 m^2. Intuitively, the first threshold is for services that require very high precision, since it is close to

the average precision provided by common GPS devices; the second is close to the average precision achievable with triangulation techniques over GSM cells, and may still be acceptable for many services; the third threshold is still adequate for some services, like localized news or weather forecasts, but may also be used for common *nearest resource* services when filtering of the answer is provided either by the LTS or on the device itself.

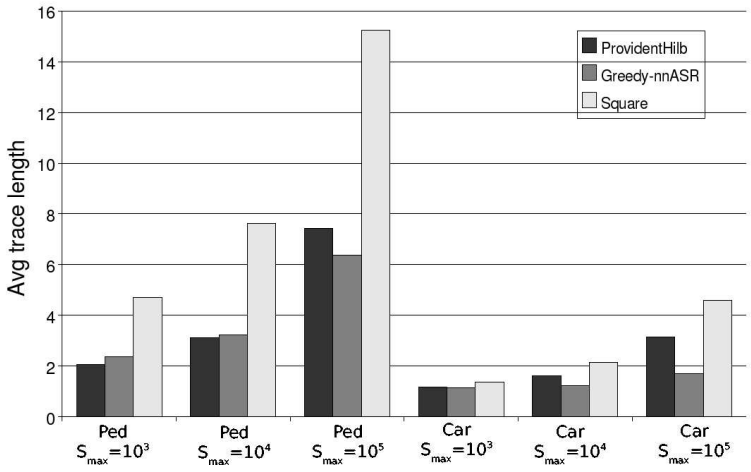


Figure 4.1: Average length of traces with $k = 10$

Figure 4.1 shows the experimental results. It is easily seen that *Square* performs better than the other two algorithms. In particular, *Square* outperforms, in terms of trace length, *Greedy-nnASR*. *ProvidentHilb* performs similarly to *Greedy-nnASR* which means that *Greedy-nnASR* has poor performances because it is a defense algorithm against $Att_{C_{st+pid}}$ while *ProvidentHilb* is a defense algorithm against $Att_{C_{st+g+pid}}$. Comparing the performance obtained considering cars and pedestrians, we can observe that requests issued by cars have much shorter traces. This indicates that the speed of the issuer strongly affects the ability of the generalization functions to generalize its trace.

In the second set of experiments we fix the value of S_{max} to 10^4 and we compare the average length of traces for different values of k. Analogously to the first experiment, we can observe that *Square* allows much longer traces

when pedestrians are considered while it provides only slightly better traces when cars are considered. Again, *ProvidentHilb* and *Greedy-nnASR* have similar performances. As expected, the larger is the value of k, the shorter are the traces.

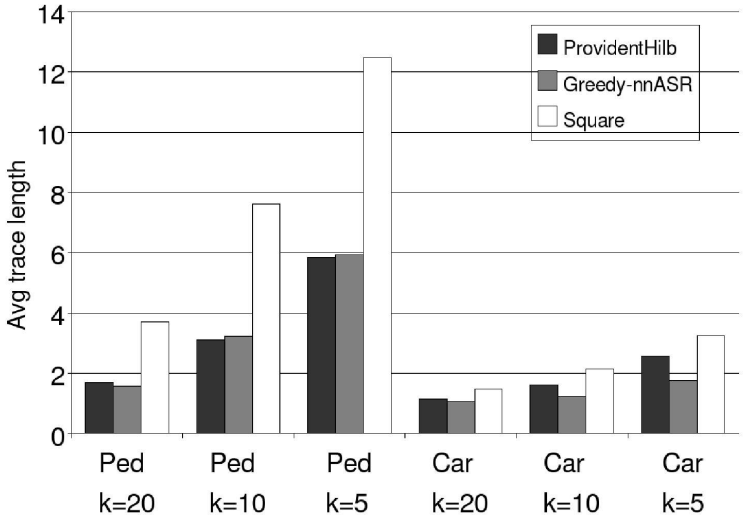


Figure 4.2: Average length of traces with $S_{max} = 10^4$

Chapter 5

Discussion

5.1 Personalization of the re-identification threshold

In this dissertation we did not consider issues related to the personalization of the re-identification threshold h. Some approaches (e.g. [22, 39]) explicitly allow different users to specify different values of the degree of anonymity that, as we showed in Section 3.1 is the inverse of the re-identification threshold h. A natural question is if the techniques proposed in this dissertation can be applied and can be considered safe even in this case. Once again, to answer this question it is essential to consider which knowledge an attacker may obtain. If the re-identification threshold h desired by each user at the time of a request is not assumed to be known by the attacker in contexts C_{st} and C_{st+g}, hence algorithms that are safe for these contexts remain safe even when the LTS admits different values of h.

However, it may be reasonable to consider contexts in which the attacker may obtain information about h. For example, if the attacker is able to derive information from different requests, he can exploit data mining techniques to derive, with a certain likelihood, the value of h required by each user. This knowledge can then be used to perform new attacks. Since these attacks are based on contexts different from C_{st+g}, the defense algo-

rithms against $Att_{C_{st+g}}$ presented in this dissertation are not able to provide protection against this new kind of attacks.

A straightforward solution to extend the defense algorithm against $Att_{C_{st+g}}$ to these cases is the following: when a request r needs to be generalized with degree of anonymity h, the anonymity set is computed considering only the users that can possibly issue a request requiring that degree of anonymity. Clearly, the solution is viable only if a limited set of h values is available and a large number of users using each value exists. If this is not the case, more sophisticated strategies need to be devised to obtain defense algorithms, and, to our knowledge, this is still an open research issue.

5.2 Shape of the generalized area

In this dissertation we assumed the generalization area to be a rectangle with edges parallel to the axis. The choice of using this shape as generalized area is driven by the fact that the SP must be able to efficiently compute the set of target objects (e.g., points of interests) that are closer to any point of the generalized area. This kind of query is called *range K nearest neighbor*[1] ($RKNN$) [28].

To the best of our knowledge, the problem of efficiently computing $RKNN$ queries was addressed only when the range is an axis parallel rectangle [28, 39] or a circonference [29]. In the general case of an arbitrary area, no efficient algorithm was proposed and therefore no other shape should be used when performing the generalization. Nevertheless, in this dissertation we disregarded the circonference as a possible shape for the generalized area. Actually, most of the defense algorithms we propose can be easily extended to generate a circonference instead of a rectangle. For example, all the algorithms that first obtain the anonymity set and then compute the minimum bounding rectangle of the locations of the users, can be modified to compute the *smallest enclosing circle* (SEC). We leave as a future work the compar-

[1]Note that the symbol K used here for range K nearest neighbor queries is different from the degree of anonymity k.

ison of the performance, in terms of the size of the generalized area, of the algorithm that uses MBR versus the algorithms that uses SEC.

A completely different approach consists in having a generalization function that, instead of returning a single generalized area, returns a set of points, each one corresponding to the (potential) location of one or more users. Then, this set of points can be used in two different ways: the LTS could send to the SP a request for each point, or the LTS could send a generalized request in which the location information is the set of points. The first possibility is analogous to the solution of issuing "fake" requests, proposed in [30]. In both cases the SP returns the union, for each point p that generalizes user's location, of the K nearest neighbor points of interests closer to p. In this case, the quality of service should be evaluated in terms of the total number of points of interest returned by the SP. To the best of our knowledge, this problem has never been specifically addressed and we leave a detailed study as a future work. However, as a preliminary consideration, we can notice that, in order to limit the number of points of interest returned by the SP, the points that generalize user's location should be close to each other so that most of them share the same K nearest points of interests. Consequently, some of the techniques adopted in this dissertation can be adapted for this different kind of generalization.

5.3 User friendly specification of temporal granularities

In Section 4.4.3 we showed how temporal granularities can be exploited to define the explicit knowledge function. Several different granularities can be used, as, for example: *early-morning*, *lunchtime*, *working-hours*, *night-before-a-working-day*.

The problem of representing time granularities has been extensively studied in the literature. Three main objectives can be identified: a) a sufficiently expressive formal model for time granularity, b) a convenient way to define

new time granularities, and c) efficient reasoning tools over time granularities. In the applicative context we present in this dissertation, the three objectives above correspond to the definition of a framework for time granularities that allows the privacy analyst (who is in charge to specify the possible attacks) to specify all the time granularities that are required to define the *ex* function. These granularities should be defined through a user friendly application and efficient reasoning should be possible, for example to identify if a granularity covers a certain time instant.

Consider a). In the last decade significant efforts have been made to provide formal models for the notion of time granularity and to devise algorithms to manage temporal data based on those models. In addition to *logical* approaches [40, 19], a framework based on periodic-set representations has been extensively studied [11], and more recently an approach based on strings and automata was introduced [49, 15]. In the following of this section, we consider the last two approaches because they support the effective computation of basic operations on time granularities. In both cases the representation of granularities can be considered as a *low-level* one, with a rather involved specification in terms of the instants of the time domain.

Consider requirement b) above. Users may have a hard time in defining granularities in formalisms based on low-level representations, and to interpret the output of operations. It is unreasonable to ask users (privacy analysts, in our applicative context) to specify granularities by linear equations or other mathematical formalisms that operate directly in terms of instants or of granules of a fixed time granularity. Hence, a second stream of research investigated more *high-level* symbolic formalisms providing a set of algebraic operators to define granularities in a compact and compositional way. The efforts on this task started even before the research on formal models for granularity [32, 41] and continued as a parallel stream of research [10, 42, 47, 48].

Finally, let us consider requirement c) above. Several inferencing operations have been defined on low-level representations, including equivalence,

inclusion between granules in different granularities, and even complex inferencing services like constraint propagation [12]. Even for simple operations no general method is available operating directly on the high level representation. Indeed, in some cases, the proposed methods cannot exploit the structure of the expression and require the enumeration of granules, which may be very inefficient. This is the case, for example, of the granule conversion methods presented by Ning e at. [42]. Moreover, we are not aware of any method to perform other operations, such as equivalence or intersection of sets of granules, directly in terms of the high level representation.

In [9] we proposed a unique framework to satisfy the requirements a), b), and c) identified above, by adding to the existing results a smart and efficient technique to convert granularity specifications from the high-level algebraic formalism to the low-level one, for which many more reasoning tools are available. In the same paper it is also shown how the proposed framework can be used as a theoretical base to design a user-friendly application for the specification of time granularities.

Chapter 6

Related work

6.1 Anonymity in databases

Guaranteeing users' anonymity is a well-known problem for the release of data in database tables [44]. In this case, the problem is to protect the association between the identity of an individual and a tuple containing her sensitive data; the attributes whose values could possibly be used to restrict the candidate identities for a given tuple are called *quasi-identifiers* [21, 14].

The idea first proposed in [44] to formally measure the intuitive notion of anonymity of a table relies on the concept of *k-anonymity*. Intuitively, a table is k anonymous if each tuple is not distinguishable, considering the values in the quasi-identifiers attributes, from at least other k tuples.

This research area has been very active in the last years. Two main problems were addressed. First, the specification of efficient algorithms that render an input table into a k anonymous table and that avoid to generalize or suppress too many values from the input table [44, 46, 45, 2, 33, 38, 1, 34] A second research direction aims to extend the notion of k-anonymity in order to provide privacy protection under different assumptions with respect to the ones considered in [44] (among others, [50, 35, 14, 51, 17, 52]). In particular, Xiao et al. [52], consider the case in which the database table can be *re-published* several times, each time after the insertion or deletion

of some tuples. That paper is particularly interesting with respect to this dissertation because the problem of the re-publication of data is intrinsic in the research area of privacy in LBSs. However, the techniques proposed in that paper cannot be easily applied in the field of LBSs. The main difference, is that in [52] each user can be identified, in different publications, by a single combination of values of quasi-identifier attributes. In practice, for a given user only sensitive data can change in different publications. On the contrary, in LBSs the location of the user can change at each request and therefore the values of the quasi identifying attributes can change each time a request is issued.

6.2 Spatio temporal anonymity

To the best of our knowledge, the problem of privacy in LBSs was first presented by Gruteser et al. [24]. In that paper, the *interval cloaking* algorithm, (see Section 3.1.2) is proposed as a spatio-temporal generalization technique to guarantee the anonymity of a LBS request.

A different algorithm, called *CliqueCloak* was proposed by Gedik et al. [22]. The main difference with respect to the *interval cloaking* algorithm is that *CliqueCloak* computes the generalization among the users that actually issue a request and not among the users that are potential issuers. Indeed, *CliqueCloak* collects original requests without forwarding them to the SP until it is not possible to find a spatio-temporal generalization that includes at least k pending requests. Then the requests are generalized and forwarded to the SP. The advantage of the proposed technique is that it allows the users to personalize the degree of anonymity (see Section 5.1). The problem of this approach is that, since in general the number of users that issue a request is much smaller than the number of users that could potentially issue a request, the spatio-temporal region requires a broader generalization in order to include other $k-1$ requests. Moreover, since the anonymity set of each request is composed by a set of users, each one being the issuer of a request, the technique is subject to the homogeneity attack discussed in

Section 2.1.1.

Bettini et al. first addressed the problem of anonymity in the dynamic case [13]. In that paper the authors show that anonymity is not guaranteed if each request in a trace is generalized using a generalization algorithm for the static case. Then, the notion of *Historical k-anonymity* is proposed to define when a trace of request is anonymous.

In [39] Mokbel et al. propose the *Casper* generalization algorithm for the static case that makes it possible to achieve the personalization of the degree of anonymity. Indeed the algorithm, described in Section 3.1.2, allows the user to specify a privacy profile composed by two parameters: k and A_{min}. The parameter k is the degree of anonymity while A_{min} is the minimum size of the generalized region that should be forwarded to the SP. In the paper it is not clear how a large value of the parameter A_{min} can enhance better privacy. Indeed, as we formalized in this dissertation, the probability that an attacker can re-identify the issuer of a request is not affected by the size of the generalized region. A different interpretation of the role of A_{min} is that a large generalized region can help to prevent the release of private information. However, in [39] this intuition is not supported by any formal result.

Beresford in [3] showed a counterexample to the *interval cloaking* algorithm. The problem, shown in Example 5, was called the "outlier problem". The first algorithm that does not suffer the "outlier problem" was proposed by Kalnis et al [29] and was called *HilbASR* (see Section 3.2.3). Though the algorithm solves the "outlier problem", no proof of its correctness was provided in [29]. Indeed, the common problem of papers [24, 22, 39, 3, 29] is the lack of a formal framework to prove the correctness of the generalization algorithm. This problem was first addressed in [8] and [37] where a preliminary version of the framework presented in this dissertation was proposed. An extended version of these papers is to be published in [6].

An alternative formal result was independently published by Ghinita et al. to prove the correctness of the *HilbASR* algorithm [23] . The paper

also presents a framework that makes it possible to compute the *HilbASR* algorithm in a distributed manner so that no centralized LTS is required. The paper presents two issues. First, the knowledge of the attacker is not modeled and therefore it is not clear how to define the probability that an attacker can re-identify the issuer (what we called "attack" in this dissertation). Second, we are not convinced about the correctness of the formal result that is used to prove that *HilbASR* is correct. Actually, as we proved in this dissertation, *HilbASR* is a defense algorithm against $UAtt_{C_{st+g}}$; However, the proof provided in [23] seems to be incorrect. Intuitively, the formal result states that a generalization function guarantees anonymity if every generalized request satisfies a property called *reciprocity*[1]. A request r' satisfies reciprocity if there exists a set AS of users located in $r'.Sdata$ such that i)$|AS| > k$ (k represents the degree of anonymity), ii) $issuer(r') \in AS$ and iii) every user $u \in AS$ is located in the generalized region computed considering all the other users in AS as issuers.

In Example 10 we show that even if all the requests returned by a generalization function g have the reciprocity property, the attacker could uniquely identify the issuer of some of the requests.

Example 10 *Consider* 4 *original requests* r_1, r_2, r_3 *and* r_4 *issued by users* i_1, i_2, i_3, i_4, *respectively. Let* g *be a generalization function such that:*

- $g(r_1).Sdata$ *and in* $g(r_2).Sdata$ *include the locations of* i_1, i_2, i_3 *and* i_4*;*
- $g(r_3).Sdata$ *includes the location of* i_1, i_2 *and* i_3*;*
- $g(r_4).Sdata$ *includes the location of* i_1, i_2, i_4.

Now, considering the following sets $AS_1 = AS_2 = AS_4 = \{i_1, i_2, i_4\}$ *and* $AS_3 = \{i_1, i_2, i_3\}$ *it can be easily seen that* $g(r_1)$, $g(r_2)$, $g(r_3)$ *and* $g(r_4)$ *have the reciprocity property with a value of* k *equals to* 3.

[1]In the following, we use the notation presented in this dissertation to discuss the results presented in [23] where a different notation was used.

However, according to Definitions 4 and 10, both $g(r_3)$ and $g(r_4)$ are not safe with respect to $UAtt_{C_{st+g}}$ with threshold $1/3$. Indeed, r_3 is the only request that is generalized by $g()$ to a requests whose spatial region includes the location of i_1, i_2 and i_3 but not the location of i_4. Therefore, $g(r_3)$ can be uniquely associated to i_3 by an attacker that knows the locations of the 4 users and the generalization function used by the LTS. Similarly, $g(r_4)$ can be uniquely associated to i_4.

6.3 Enforcing location privacy

As explained in Section 2, in the single-issuer case a user's privacy can be protected by preventing the attacker from identifying the identity or the private information of that user. In this dissertation we focus on enforcing anonymity i.e., in preventing the attacker from discovering the identity of the user.

In [26] a technique is proposed to prevent the attacker from inferring user's private information. In this case, the private information is represented by the *sensitive areas* i.e., the locations that the user does not want to be associated with her identity. In the proposed solution, the user specifies the sensitive areas so that the privacy preserving system can prevent the attacker from understanding that the user visited a sensitive area. The straightfarward solution of suppressing all the requests from these areas is not effective since an attacker could infer that a user visited a sensitive area only from her request trace outside the sensitive area.

The proposed solution is based on a partition of all the areas (sensitive or not) in zones, each one including at least k sensitive areas. Then, each request is suspended until the user crosses a zone boundary. If the user has not visited a sensitive area, all the pending requests are submitted, otherwise they are suppressed.

To the best of our knowledge, the proposed solution is the first one that addresses this kind of problem. However, it is not clear if it is an effective solution. First, it is debatable if it is appropriate to extend to sensitive areas

the concept of k-anonymity. Indeed, if a user specifies some sensitive areas, she does not want her identity to be associated with any of them; On the contrary the proposed solution allows an attacker to infer that a user visited a sensitive area even if it cannot say which one in a set of k. Secondly, it is not clear if it is acceptable to always postpone the submission of a request until a user changes a zone. Finally, the way in which the zones are constructed is critical. Indeed, it seems possible, in some specific cases, that despite the proposed defense technique, an attacker could infer the exact sensitive area a user visits.

6.4 Identification and prevention of critical request traces

An important aspect in the dynamic scenario is how an attacker can identify a request trace, and how a privacy preserving system can avoid it. Two cases have been considered.

In [4, 5] LBSs that require a pseudo-id are considered. The proposed privacy preserving technique is based on the notion of *mix-zone* introduced by the authors, and aims to avoid that an attacker traces for a long time the requests from the same user. The central idea is to change a user's pseudo-id each time the user enters in a mix-zone. A mix-zone is analogous to a mix-node in communication systems [18], and can be intuitively described as a spatial area such that, if an individual crosses it, then it won't be possible to link his future positions (outside the area) with known positions (before entering the area).

The results can be applied in the dynamic case, but cannot be used to provide a complete solution to any of the threats described in Section 2.1. Indeed, the technique aims at reducing the length of the request traces but does not evaluate if he attacker can obtain the sensitive association. Nevertheless, reducing the request trace length is an important task to facilitate privacy protection in a dynamic context. Hence, this technique could be

very useful as a part of a privacy preserving system.

A different approach to the issue of request traces is to consider LBS that do not require pseudo-ids. This case is considered in [25] where the authors experiment to see if it is possible for an attacker to trace a user. A known algorithm for tracking multiple objects is applied to trace a small number of users whose locations are frequently collected. The authors conclude that it is practically possible for an attacker to obtain request traces even if pseudo-ids are not submitted to the SP. This paper does not propose a solution to preserve privacy but is a preliminary step in the definition of a technique that could be used by a privacy preserving system to evaluate if a user is possibly being traced by an attacker. In the absence of such a solution, a privacy preserving system should adopt a conservative approach assuming that a user can always be traced.

6.5 Techniques based on access control

Advanced access control models can be used in the context of LBS services to specify and enforce privacy policy rules. The rules can define, for example, the type of data that each service provider can access, the resolution of that data, and possibly other constraints. With respect to our reference model, policies can be defined by users as well as by service providers, and can be enforced by the LTS. Among the efforts in this direction, in [53] a push-based LBS scenario is considered; users can define authorizations that not only select which service providers can access location/profile information, but can also constrain the area and time in which they can send their offers to the users. The LTS is in charge of enforcing the authorizations. Among other efforts, the IETF Geopriv working group is proposing a format for expressing privacy preferences for location information [20].

Access control is an important component of a privacy preserving solution. However, the best results in addressing the privacy threats would probably be obtained by coupling access control with anonymization techniques discussed in this dissertation.

Chapter 7

Conclusions and future work

7.1 Summary of the contributions

The main message of this dissertation is that it is not possible to guarantee the correctness of a privacy-preserving technique if there are no assumptions about the external knowledge available to an attacker and about his reasoning abilities.

To support this idea, we presented a formal framework that formalizes, among others, the concepts of *attack* and *defense function*. The former makes it possible to specify, for any context C, the best effort to identify the issuer of a request that can be performed by the attackers having at most external knowledge and reasoning abilities specified by C. A defense function is a generalization function that is proved to provide privacy protection against a given attack. We showed that a generalization function that is a defense function against an attack based on a certain context may not be a defense function against an attack based on a different context.

Based on this formalism, we specified the attacks in the two contexts that were implicitly considered in most of the previous work and we catalogued the generalization algorithms proposed in the literature according to the contexts in which they compute a defense function. We also presented a new generalization algorithm that outperforms the existing generalization

algorithms in terms of size of the generalized area and that has similar performances in terms of computation time.

The formalism we propose can also be used to model the dynamic case i.e., when the attacker can reason with a trace of requests issued by the same (anonymous) user. No detailed algorithm was proposed in the literature to provide a protection in this context. In this dissertation we proposed new defense algorithms and we empirically compared them showing that, in this case, the protection requires a much broader generalization than in the static case. Our empirical results give evidence that, if the attacker knows the identity of each user in each location (as in the contexts assumed in many previous works) and is able to link requests issued by the same user, only few requests can be issued by a user before the generalized area becomes so large as to make the service useless.

To address this problem, we presented a less conservative context in which it is possible to model an approximate knowledge of users locations. Our preliminary results in this direction consist of two generalization algorithms that we conjecture are defense functions. We leave the proof of correctness of these algorithms and their experimental evaluation as a future work.

7.2 Future work

Our current research effort is dedicated to the improvement of the results in the context in which the attacker has approximate knowledge of users locations. First of all, we need to formally prove the correctness of the algorithms that we conjecture are defense functions. Moreover, we need to empirically evaluate these algorithms. In order to obtain experimental results it is necessary to have traces of users' movements for a long period, in the order of several days, or weeks. We are tackling the problem of obtaining this kind of trace with two different solutions. First, we are recording the movements of real users using GPS receivers. This solution has the advantage of providing real data and not just simulated data. However, we will

obtain a limited number of traces (in the order of few tens). A different solution consists in generating synthetic data through a simulator. We are currently using a simulator called SIAFU [36], that makes it possible to generate users' movements considering users' activities, like working, staying at home or similar. Our short terms objective is to simulate the movements of the inhabitants of a small town (about 500 users) for a time period of one week. Our long term objective is to simulate the movements of thousands of users in a medium sized city were about one million people live. When the input data will be available, we will be able to evaluate the efficiency, in terms of size of the generalized area and computation time, of the algorithm proposed in this dissertation for this context.

In future, we are also planning to investigate how to protect users against the general privacy threat depicted in Figure 2.1. Preliminary results presented in [43] suggest that the framework proposed in this dissertation requires an extension in order to model the multiple-issuer cases in which, as we pointed out in Section 2.1.1, *homogeneity attacks* can be performed.

Another open problem, discussed in Section 5.2, is how different shapes of the generalized area can be used. To address this problem, it could be necessary to reconsider the metric used to evaluate the quality of the generalization. In this dissertation, we used the size of the generalized area as the only metric but, in general, different definitions can be provided. For example, if a specific service is considered (e.g., to find the closest points of interests) then a specific measure of the quality of service can be provided (e.g., the average number of points of interest returned by the service provider).

Finally, we believe that most of the results presented in this dissertation can be extended to the more general case of privacy in context based services, in which the reply to a user's request is computed considering different context information about that user. Clearly, the location of the user is a relevant context information that could require generalization.

Appendix A

Proofs

A.1 Proof of Theorem 1

Proof. By Definition 5, a generalization function $g()$ is a defense function against $UAtt_{C_{st}}$ with threshold $1/k$ if for each original request r, $g(r)$ is safe against $UAtt_{C_{st}}$ with threshold $1/k$. By Definition 4, $g(r)$ is safe against $UAtt_{C_{st}}$ with threshold $1/k$ if $UAtt_{C_{st}}(g(r), issuer(r)) \leq 1/k$. Therefore, we only need to show that, for each original request r, $UAtt_{C_{st}}(g(r), issuer(r)) \leq 1/k$.

Since $UAtt_{C_{st}}$ is a uniform attack, by Proposition 1 it follows that, for each original request r, $UAtt_{C_{st}}(g(r), issuer(r)) = (1/|AS_{C_{st}}(g(r))|)$. By hypothesis, $g()$ is such that, for each original request r, $g(r)$ is k-anonymous in context C_{st}. By Definition 9 this implies that $|AS_{C_{st}}(g(r))| \geq k$, and therefore $UAtt_{C_{st}}(g(r), issuer(r)) \leq 1/k$. □

A.2 Proof of Theorem 2

Proof. To prove the theorem, we only need to show that, for each generalized request r', if $i \in AS_{C_{st+g}}(r')$ then $i \in AS_{C_{st}}(r')$. By definition of anonymity set and by Definition 10, for each generalized request r', given $r = o(r', i, loc_i(r'.Tdata))$, if $i \in AS_{C_{st+g}}(r')$ then $P[g(r) = r'] > 0$. By Definition 1, if $P[g(r) = r'] > 0$, then $r.Sdata \in r'.Sdata$. By definition

of the $o()$ function, $issuer(r) = i$ and, from Property 1, it follows that $loc_i(r.Tdata) = r.Sdata$. Consequently, $loc_i(r.Tdata) \in r'.Sdata$ and hence, by Definition 8, $i \in AS_{C_{st}}(r')$. □

A.3 Proof of Theorem 3

In order to prove Theorem 3, we first formulate and prove Lemma 1.

Lemma 1 *If the generalization algorithm used by the LTS computes a deterministic generalization function g, the attack $Att_{C_{st+g}}$ is uniform and, for each $r' \in R'$ it is characterized by*

$$AS_{C_{st+g}}(r') = \{i \in I | g(o(r', i, loc_i(r'.Tdata)) = r'\}$$

Proof of Lemma 1.

Proof. By Definition 10, the attack based on context C_{st+g} is given by

$$Att_{C_{st+g}}(r', i) = \frac{P[g(o(r', i, loc_i(r'.Tdata)) = r']}{\sum_{j \in I} P[g(o(r', j, loc_j(r'.Tdata)) = r']}$$

If the generalization algorithm used by the LTS computes a deterministic generalization function g, then, for each $r' \in R'$ and $i \in I$,

$$P[g(o(r', i, loc_i(r'.Tdata)) = r'] \in \{0, 1\}$$

Indeed, $Att_{C_{st+g}}(r', i)$ equals 0 if $g(o(r', i, loc_i(r'.Tdata)) \neq r'$; Otherwise, it equals $1/|AS|$ where AS is the set of users j such that $g(o(r', j, loc_j(r'.Tdata)) = r'$. Therefore, according to Definition 3, $Att_{C_{st+g}}$ is a uniform attack. Moreover, for each $r' \in R'$, $Att_{C_{st+g}}$ is characterized by the anonymity set

$$AS_{C_{st+g}}(r') = \{i \in I | g(o(r', i, loc_i(r'.Tdata)) = r'\}$$

□

Proof of Theorem 3.

Proof. By Definition 6, Gen_{st+g} is a defense algorithm if the execution of Gen_{st+g} with h as re-identification threshold computes a defense function against Att_{Cst+g} with re-identification threshold h. By Definition 5,

Gen_{st+g} computes a defense function against $Att_{C_{st+g}}$ with re-identification threshold h if, for each original request $r \in R$, $r' = Gen_{st+g}(r, h)$ is a safe request against $Att_{C_{st+g}}$ with re-identification threshold h. By Definition 4, r' is a safe request against $Att_{C_{st+g}}$ with re-identification threshold h if $Att_{C_{st+g}}(r', issuer(r')) \leq h$. Since Algorithm 1 computes a deterministic generalization function, by Lemma 1, $Att_{C_{st+g}}(r', issuer(r')) \leq h$ if

$$|AS_{C_{st+g}}(r')| = |\{i \in I | g(o(r', i, loc_i(r'.Tdata)) = r'\}| \geq 1/h$$

We prove that this inequality holds by proving that, in each execution of Algorithm 1, the last value A assigned to variable AS is such that (1) the set A has cardinality at least $1/h$ and (2) for each user i in A, there exists a potential request $\bar{r}$ in R issued by i such that $Gen_{st+g}(\bar{r}, h) = r'$. The thesis follows from the fact that the request returned by Algorithm 1 has $MBR(A)$ as generalized area.

(1) Since at Line 1 of Algorithm 1 the variable AS is set to I that has cardinality at least $1/h$ and AS is reassigned to a block only if each block has cardinality at least $1/h$, varibale AS is never assigned to a set with cardinality smaller than $1/h$.

(2) Let i be a user in A, and $\bar{r} = o(r', i, loc_i(r'.Tdata)$ be a potential original request issued by i. Moreover, let $AS(j)$ and $\overline{AS}(j)$ denote the values of the variables AS in the j-th iteration of $Gen_{st+g}(r, h)$ and $Gen_{st+g}(\bar{r}, h)$, respectively. We show by induction on the number of iterations that $AS(j) = \overline{AS}(j)$ for each iteration j.

Induction basis: In the first iteration, $AS(1) = \overline{AS}(1) = I$.

Induction: Assume at iteration j, $AS(j) = \overline{AS}(j)$. We need to show that either both executions terminate at that iteration, or $AS(j+1) = \overline{AS}(j+1)$. Indeed, since $AS(j) = \overline{AS}(j)$, the partition procedure is deterministic and has the value of the variable AS as the only input, it follows that the same partition $\{AS_1, \ldots, AS_n\}$ is computed for $AS(j)$ and for $\overline{AS}(j)$. Consequently, if there exists a block with cardinality less than $1/h$, both executions terminate, returning the same value for AS. On the contrary, if all the blocks have cardinality larger than or equal to $1/h$, then $AS(j+1)$

is the block in $\{AS_1 \ldots, AS_n\}$ that contains $issuer(r)$ and $\overline{AS}(j+1)$ is the block in the same set that contains i. We claim that $AS(j+1)$ must contain i. Indeed, we know that the resulting AS of the algorithm must be a subset of $AS(j+1)$ due to successive partitioning method of the algorithm and we assumed that i is in the resulting AS. Now, since $\overline{AS}(j+1)$ also contains i, we know $AS(j+1) = \overline{AS}(j+1)$ due to the fact that $\{AS_1 \ldots, AS_n\}$ is a partition of $AS(j) = \overline{AS}(j)$. □

A.4 Proof of Theorem 4

Proof. Analogously to the proof of Theorem 3, we need to prove that

$$|AS_{C_{st+g}}(r')| = |\{i \in I | g(o(r', i, loc_i(r'.Tdata)) = r'\}| \geq 1/h$$

where the generalization algorithm used by the LTS to compute g is $dichotomicPoints()$. To prove that this inequality holds, we only need to show that (1) the last set A assigned to variable AS in Algorithm 2 has cardinality at least $1/h$ and (2) for each user i in A, there exists a request $\bar{r}$ in R issued by i such that $dichotomicPoints(\bar{r}, h) = r'$.

(1) At each iteration of the main loop, if $|AS| \geq 2/h$, AS is partitioned in two blocks, otherwise the algorithm terminates. The two blocks in which AS is partitioned have cardinality $\lfloor |AS|/2 \rfloor$ and $\lceil |AS|/2 \rceil$, respectively. Since AS is partitioned only if $|AS| \geq 2/h$, the two blocks have at least cardinality $1/h$. Therefore, the final AS set has at least cardinality $1/h$.

(2) Since AS is the only parameter used in the computation of the partition, the proof is analogous to point (2) in the proof of Theorem 3. □

A.5 Proof of Theorem 5

Proof. Analogously to the proof of Theorem 3, we need to prove that

$$|AS_{C_{st+g}}(r')| = |\{i \in I | g(o(r', i, loc_i(r'.Tdata)) = r'\}| \geq 1/h$$

where the generalization algorithm used by the LTS to compute g is $grid()$. To prove that this inequality holds, we only need to show that (1) the last set A assigned to variable AS in Algorithm 3 has cardinality at least $1/h$ and (2) for each user i in A, there exists a request $\bar{r}$ in R issued by i such that $grid(\bar{r}, h) = r'$.

(1) In each of the two iterations of the Algorithm 3, variable AS is reassigned to a set with cardinality at least equal to the current value of the variable upb. Indeed, the cardinality of AS is given by $end - start + 1$; When the condition of the **if** statement of line 13 is verified, then $end - start + 1 = upb$. Otherwise,

$$end - start + 1 = |AS| - 1 - (nob - 1)upb + 1 =$$

$$= |AS| - nob \cdot upb + upb \geq |AS| - |AS| + upb = upb$$

During the first iteration $upb = \lfloor |I|/nob \rfloor$ where $nob = \left\lfloor \sqrt{|I| \cdot h} \right\rfloor$ (note that the value of nob is fixed before the main cycle and is never changed). During the second iteration, upb is set to $\lfloor |AS|/nob \rfloor$. Since $|AS|$ is at least the value that upb has during the first iteration, then $upb \geq \left\lfloor \frac{\lfloor |I|/nob \rfloor}{nob} \right\rfloor$ and hence, since $k \geq 1/h$, we need to prove that $\left\lfloor \frac{\lfloor |I|/nob \rfloor}{nob} \right\rfloor \geq k$ This inequality is equivalent to $\frac{\lfloor |I|/nob \rfloor}{nob} \geq k$ and hence to $\lfloor |I|/nob \rfloor \geq k \cdot nob$.

This inequality follows from the fact that, by definition, $nob = \left\lfloor \sqrt{|I|/k} \right\rfloor$, hence $nob \leq \sqrt{|I|/k}$ and therefore, since $nob > 0$, $k \cdot nob^2 \leq |I|$. Consquently, $|I|/nob \geq k \cdot nob$ and finally $\lfloor |I|/nob \rfloor \geq \lfloor k \cdot nob \rfloor = k \cdot nob$.

(2) Since AS is the only parameter used in the computation of the partition, the proof is analogous to point (2) in the proof of Theorem 3. □

A.6 Proof of Theorem 6

Proof. By Definition 6, *Greedy-nnASR* is a defense algorithm if the execution of *Greedy-nnASR* with h as re-identification threshold computes a defense function against $Att_{Cst+pid}$ with re-identification threshold h. By Definition 5, *Greedy-nnASR* computes a defense function against $UAtt_{C_{st+pid}}$

with re-identification threshold h if, for each original request $r \in R$, $r' = Greedy\text{-}nnASR(r, h, \tau, S_{max})$ is a safe request against $UAtt_{C_{st+pid}}$ with re-identification threshold h. By Definition 4, r' is a safe request against $UAtt_{C_{st+g}}$ with re-identification threshold h if $UAtt_{C_{st+pid}}(r', issuer(r')) \leq h$. Since $UAtt_{C_{st}}$ is a uniform attack, by Proposition 1 it follows that, $UAtt_{C_{st+pid}}(r', issuer(r)) = (1/|AS_{C_{st+pid}}(r')|)$. Therefore we need to prove that $|AS_{C_{st+pid}}(r')| \geq 1/h$. where, by Definition 17,

$$AS_{C_{st+pid}}(r') = \bigcap_{r'' \in L_{C_{pid}}(r')} AS_{C_{st}}(r'')$$

If $\tau = \emptyset$, then $AS_{C_{st+pid}}(r') = AS_{C_{st}}(r')$ and the thesis follows the fact that $nnASR$ is a defense algorithm against $UAtt_{C_{st}}$. The same holds if unlinking is performed (Line 12 and 13).

In the other cases, a request r' is returned that has, as generalized area, the MBR of the current locations of the users in the anonymity set of r'''. Therefore, $AS_{C_{st}}(r') \supseteq AS_{C_{st}}(r''')$. Analogously, for each request $\bar{r} \in L_{C_{pid}}(r')$, $AS_{C_{st}}(\bar{r}) \supseteq AS_{C_{st}}(r''')$. Consequently, $\bigcap_{r'' \in L_{C_{pid}}(r')} AS_{C_{st}}(r') = AS_{C_{st}}(r''')$. The thesis follows from the fact that r''' is a safe request against $UAtt_{C_{st+pid}}$ with re-identification threshold h. □

A.7 Proof of Theorem 7

Proof. We prove that, for each original request $r \in R$, $h \in (0, 1]$, trace τ and value S_{max}, the execution of $Square(r, h, \tau, S_{max})$ returns a request r' that is either r_{null} or a safe request against $UAtt_{C_{st+pid}}$ with re-identification threshold h. Since $UAtt_{C_{st+pid}}$ is a uniform attack, $UAtt_{C_{st+pid}} = (1/|AS_{C_{st+pid}}(r')|)$ where, by Definition 17, $AS_{C_{st+pid}}(r') = \bigcap_{r'' \in L_{C_{pid}}(r')} AS_{C_{st}}(r'')$. Therefore we prove that $\bigcap_{r'' \in L_{C_{pid}}(r')} AS_{C_{st}}(r'') \geq 1/h$.

If $\tau = \emptyset$, then variable AS is reassigned, at Line 12, to the set of users of I whose location is inside the square s defined al Line 11. If $|AS| < 1/h$ then the **if** conditions of Lines 13 and 17 are not verified and r_{null} is returned. Otherwise, if $|AS| \geq 1/h$, the request r' having $MBR(AS, r.Tdata)$

as generalized area is returned. By Definition 8, $AS = AS_{C_{st}}(r')$. Since $\tau = \emptyset$, $L_{C_{pid}}(r') = \{r'\}$ and hence $\bigcap_{r'' \in L_{C_{pid}}(r')} AS_{C_{st}}(r'') = AS_{C_{st}}(r')$ that has cardinality at least $1/h$. Hence r' is a safe request against $UAtt_{C_{st+pid}}$ with re-identification threshold h.

If $\tau \neq \emptyset$, then let r'' be the last request in temporal order in τ. In this case, at Line 12 the variable AS is reassigned to the set of users in $AS_{C_{st+pid}}(r'')$ whose location is inside s. Therefore, $AS \subseteq AS_{C_{st+pid}}(r'')$. If $|AS| \geq 1/h$, the request r' having $MBR(AS, r.Tdata)$ as generalized area is returned. By Definition 8, $AS = AS_{C_{st}}(r')$. By Definition 17, $AS_{C_{st+pid}}(r') = \bigcap_{r''' \in L_{C_{pid}}(r')} AS_{C_{st}}(r''')$. Since r'' is the last request in temporal order in τ, $L_{C_{pid}}(r') = L_{C_{pid}}(r'') \cup \{r'\}$. Therefore,

$$AS_{C_{st+pid}}(r') = \bigcap_{r''' \in L_{C_{pid}}(r'')} AS_{C_{st}}(r''') \cap AS_{C_{st}}(r')$$

By Definition 17,

$$\bigcap_{r''' \in L_{C_{pid}}(r'')} AS_{C_{st}}(r''') = AS_{C_{st+pid}}(r'')$$

Therefore,

$$AS_{C_{st+pid}}(r') = AS_{C_{st+pid}}(r'') \cap AS_{C_{st}}(r')$$

Since $AS_{C_{st}}(r') \subseteq AS_{C_{st+pid}}(r'')$, then $AS_{C_{st+pid}}(r') = AS_{C_{st}}(r')$ that, by hypotesis, has cardinality at least $1/h$. Hence r' is a safe request against $UAtt_{C_{st+pid}}$ with re-identification threshold h.

If $|AS| < 1/h$ then AS is reassigned to the set of users of I whose location is inside s. The proof of the theorem in this case is analogous to the proof in the case in which $\tau = \emptyset$. □

A.8 Proof of Theorem 8

The proof of Theorem 8 is analogous to the proof of Lemma 1.

A.9 Proof of Theorem 9

Proof. Since *ProvidentHilb* is a deterministic algorithm, by Theorem8 it follows that $Att_{C_{st+g+pid}}$ is a uniform attack and, for each generalized request r' computed by *ProvidentHilb*, the attack is characterised by the anonymity set

$$AS_{C_{st+g+pid}}(r') = \{i \in I | \forall r'' \in L_{C_{pid}}(r')\ g(r''_i, \tau^{r'}_{r''}) = r''\}$$

where, for each $i \in I$ and each $r'' \in R'$, $r''_i = o(r'', i, loc_i(r''.Tdata))$ and $\tau^{r'}_{r''}$ is the set of requests of $L_{C_{pid}}(r')$ issued before r''.

We prove that, for each original request $r \in R$, $h \in (0, 1]$, trace τ and value S_{max}, the execution of $ProvidentHilb(r, h, \tau, S_{max})$ returns a request r' that is safe request against $UAtt_{C_{st+g+pid}}$ with re-identification threshold h.

If $\tau = \emptyset$, *ProvidentHilb* returns the request r' having $MBR(B, r.Tdata)$ as generalized area, where B is the block of the parition computed with $HilbPart$ that contains the issuer. Since $\tau = \emptyset$, $L_{C_{pid}}(r') = \{r'\}$ and hence

$$\{i \in I | \forall r'' \in L_{C_{pid}}(r')\ g(r''_i, \tau^{r'}_{r''}) = r''\} = \{i \in I | g(r'_i, \emptyset) = r'\}$$

Since, the *HilbPart* function is computed independently from the issuer, for each user j in B, $g(r'_j, \emptyset) = r'$. Since each block B returned by the $HilbPart$ has at least cardinality $1/h$, $AS_{C_{st+g+pid}}(r') \geq 1/h$ and hence r' is a safe request against $UAtt_{C_{st+pid}}$ with re-identification threshold h.

If $\tau \neq \emptyset$, then *ProvidentHilb* computes the partition *part* among $AS_{C_{st+g+pid}}(r'')$ i.e., the set of users in the anonymity set of the last request r'' of τ. Then, the algorithm returns the request r' that has $MBR(B, r.Tdata)$ as generalized area, where B is the block computed with $HilbPart$ that contains the issuer. By Theorem 8,

$$AS_{C_{st+g+pid}}(r') = \{i \in I | \forall r''' \in L_{C_{pid}}(r')\ g(r'''_i, \tau^{r'}_{r'''}) = r''\}$$

Since r'' is the last request in temporal order in τ,

$$AS_{C_{st+g+pid}}(r') = \{i \in I | \forall r''' \in L_{C_{pid}}(r'')\ g(r'''_i, \tau^{r'}_{r'''}) = r''' \text{ and } g(r'_i, \tau^{r'}_{r'}) = r'\}$$

The last equation can be rewritten as

$$AS_{C_{st+g+pid}}(r') = \{i \in I | \forall r''' \in L_{C_{pid}}(r'')\ g(r_i''', \tau_{r'''}^{r'}) = r'''\} \cap \{i \in I | g(r_i', \tau_{r'}^{r'}) = r'\}$$

By Theorem 8,

$$\{i \in I | \forall r''' \in L_{C_{pid}}(r'')\ g(r_i''', \tau_{r'''}^{r'}) = r'''\} = AS_{C_{st+g+pid}}(r'')$$

therefore

$$AS_{C_{st+g+pid}}(r') = AS_{C_{st+g+pid}}(r'') \cap \{i \in I | g(r_i', \tau_{r'}^{r'}) = r'\}$$

Since the block B is computed among the users in $AS_{C_{st+g+pid}}(r'')$ and has at least cardinality $1/h$, then $AS_{C_{st+g+pid}}(r') \geq 1/h$ and hence r' is a safe request against $UAtt_{C_{st+pid}}$ with re-identification threshold h.

□

Appendix B

Notation

A summary of the notation used in this dissertation follows:

- R the set of all possible original requests.
- R' the set of all possible generalized requests.
- r an original request (if not differently stated).
- r' a generalized request (if not differently stated).
- $r.IDdata$ the identity of the issuer of r.
- $r'.IDdata$ a pseudo-id or **null**.
- $r.STdata$ the exact spatio-temporal information of r.
- $r'.STdata$ the possibly generalized spatio-temporal information of r'.
- $r.Sdata$ the exact location from where r is issued.
- $r'.Sdata$ the generalized area from where r' is issued.
- $r.Tdata$ ($r'.Tdata$) the exact time from where r (r', respectively) is issued.
- $r.SSdata$ ($r'.SSdata$) the service specific information of r (r', respectively).

- $o(r', i, s)$ the potential original request r equal to r' except from the fact that it is issued by i from the location s.
- I the set of users.
- i a user in I (if not differently stated).
- $issuer(r)$ ($issuer(r')$) the issuer of r (r', respectively).
- A_{tot} the total area where the LTS provides privacy protection.
- $loc_i(t)$ the location of i at time t.
- g a deterministic or randomized generalization function.
- $Att_C(r', i)$ an attack based on context C.
- $AS_C(r')$ the anonymity set of r' in context C, i.e., the set of users that have non zero probability, considering Att_C, of being the issuer of r'.
- $UAtt_C(r', i)$ the uniform attack based on context C.
- h the re-identification threshold.

Bibliography

[1] G. Aggarwal, T. Feder, K. Kenthapadi, S. Khuller, R. Panigrahy, D. Thomas, and A. Zhu. Achieving anonymity via clustering. In *Proc. of the 25th ACM symposium on Principles of database systems*, pages 153–162. ACM Press, 2006.

[2] R. J. Bayardo and R. Agrawal. Data privacy through optimal k-anonymization. In *Proc. of the 21st International Conference on Data Engineering*, pages 217–228. IEEE Computer Society, 2005.

[3] A. R. Beresford. *Location privacy in ubiquitous computing*. PhD thesis, University of Cambridge, 2005.

[4] A. R. Beresford and F. Stajano. Location privacy in pervasive computing. *IEEE Pervasive Computing*, 2(1):46–55, January–March 2003.

[5] A. R. Beresford and F. Stajano. Mix zones: User privacy in location-aware services. In *Proc. of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops*, page 127. IEEE Computer Society, 2004.

[6] C. Bettini, S. Mascetti, and X. S. Wang. Privacy protection through anonymity in location-based services. *Handbook of Database Security: Applications and Trends,* Springer, To appear.

[7] C. Bettini, S. Mascetti, and X. S. Wang. Privacy threats in location-based services. In S. Shekhar and H. Xiong, editors, *Encyclopedia of GIS*. Springer, To appear.

[8] C. Bettini, S. Mascetti, X. S. Wang, and S. Jajodia. Anonymity in location-based services: towards a general framework. In *Proc. of the 8th International Conference on Mobile Data Management*. IEEE Computer Society, 2007.

[9] C. Bettini, S. Mascetti, X. S. Wang, and S. Jajodia. Supporting temporal reasoning by mapping calendar expressions to minimal periodic sets. *Journal of Artificial Intelligence Research*, 28:299–348, 03 2007.

[10] C. Bettini and R. D. Sibi. Symbolic representation of user-defined time granularities. *Annals of Mathematics and Artificial Intelligence*, 30(1-4):53–92, 2000.

[11] C. Bettini, X. S. Wang, and S. Jajodia. *Time Granularities in Databases, Data Mining, and Temporal Reasoning*. Springer, 2000.

[12] C. Bettini, X. S. Wang, and S. Jajodia. Solving multi-granularity temporal constraint networks. *Artif. Intell.*, 140(1/2):107–152, 2002.

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

[14] C. Bettini, X. S. Wang, and S. Jajodia. The role of quasi-identifiers in k-anonymity revisited. Technical Report RT-11-06, DICo, University of Milan, 2006.

[15] D. Bresolin, A. Montanari, and G. Puppis. Time granularities and ultimately periodic automata. In Proc. of the 9th European Conference on Logics in Artificial Intelligence *volume 3229 of* LNCS, pages 513–525. Springer, 2004.

[16] T. Brinkhoff. A framework for generating network-based moving objects. *GeoInformatica*, 6(2):153–180, 2002.

[17] J.-W. Byun, Y. Sohn, E. Bertino, and N. Li. Secure anonymization for incremental datasets. In *Proc. of Third VLDB Workshop on Secure Data Management*, Lecture Notes in Computer Science. Springer, 2006.

[18] D. L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981.

[19] C. Combi, M. Franceschet, and A. Peron. Representing and reasoning about temporal granularities. *Journal of Logic and Computation*, 14(1):51–77, 2004.

[20] J. Cuellar, J. Morris, D. Mulligan, J. Peterson, and J. Polk. Geopriv requirements, 2006. http://www.ietf.org/html.charters/geopriv-charter.html.

[21] T. Dalenius. Finding a needle in a haystack - or identifying anonymous census record. *Journal of Official Statistics*, 2(3):329–336, 1986.

[22] B. Gedik and L. Liu. Location privacy in mobile systems: A personalized anonymization model. In *Proc. of the 25th International Conference on Distributed Computing Systems*, pages 620–629. IEEE Computer Society, 2005.

[23] G. Ghinita, P. Kalnis, and S. Skiadopoulos. Prive: anonymous location-based queries in distributed mobile systems. In *Proc. of the 16th international conference on World Wide Web*, pages 371–380. ACM Press, 2007.

[24] M. Gruteser and D. Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *Proc. of the 1st International Conference on Mobile Systems, Applications and Services (MobiSys)*. The USENIX Association, 2003.

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

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

[27] B. Hoh and M. Gruteser. Protecting location privacy through path confusion. In *Proc. of the First International Conference on Security and Privacy for Emerging Areas in Communications Networks (SecureComm)*, pages 194–205. IEEE Computer Society, 2005.

[28] H. Hu and D. L. Lee. Range nearest-neighbor query. *IEEE Transactions on Knowledge and Data Engineering*, 18(1):78–91, 2006.

[29] P. Kalnis, G. Ghinta, K. Mouratidis, and D. Papadias. Preserving anonymity in location based services. Technical Report B6/06, National University of Singapore, 2006.

[30] H. Kido, Y. Yanagisawa, and T. Satoh. An anonymous communication technique using dummies for location-based services. In *Proc. of the International Conference on Pervasive Services*, pages 88–97. IEEE Computer Society, 2005.

[31] A. Kobsa. Privacy-enhanced personalization. *Communications of the ACM*, 50(8):24–33, 2007.

[32] B. Leban, D. McDonald, and D. Forster. A representation for collections of temporal intervals. In Proc. of the 5th National Conference on Artificial Intelligence, pages 367–371. Morgan Kaufmann, 1986.

[33] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Mondrian multidimensional k-anonymity. In *Proc of the 22nd International Conference on Data Engineering*, 2006.

[34] A. Machanavajjhala and J. Gehrke. On the efficiency of checking perfect privacy. In *Proc. of the 25th ACM symposium on Principles of database systems*, pages 163–172. ACM Press, 2006.

[35] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. *Proc. of the 22nd International Conference on Data Engineering*, 0:24, 2006.

[36] M. Martin and P. 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.

[37] S. Mascetti and C. Bettini. A comparison of spatial generalization algorithms for lbs privacy preservation. In *Proc. of the 1st International Workshop on Privacy-Aware Location-based Mobile Services*. IEEE Computer Society, 2007.

[38] S. Mascetti, C. Bettini, X. S. Wang, and S. Jajodia. k-anonymity in databases with timestamped data. In *Proc. of 13th International Symposium on Temporal Representation and Reasoning*. IEEE Computer Society, 2006.

[39] M. F. Mokbel, C.-Y. Chow, and W. 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.

[40] A. Montanari. *Metric and Layered Temporal Logic for Time Granularity*. PhD thesis, ILLC Dissertation Series 1996-02, University of Amsterdam, 1996.

[41] M. Niezette and J. M. Stevenne. An efficient symbolic representation of periodic time. In Proc. of the first International Conference on Information and Knowledge Management *volume 725 of* Lecture Notes in Computer Science, pages 161–168. Springer, 1992.

[42] P. Ning, X. S. Wang, and S. Jajodia. An algebraic representation of calendars. *Annals of Mathematics and Artificial Intelligence*, 36(1-2):5–38, 2002.

[43] L. Pareschi and C. Bettini. Beyond anonymity in location based services. In *Proc. of the 15th Italian Symposium on Advanced Database Systems*, pages 447–454, 2007.

[44] P. Samarati. Protecting respondents' identities in microdata release. *IEEE Transactions on Knowledge and Data Engineering*, 13(6):1010–1027, 2001.

[45] L. Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5):571–588, 2002.

[46] L. Sweeney. k-anonymity: a model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5):557–570, 2002.

[47] P. Terenziani. Symbolic user-defined periodicity in temporal relational databases. *IEEE Transactions of Knowledge and Data Engineering*, 15(2):489–509, 2003.

[48] B. Urgun, C. E. Dyreson, R. T. Snodgrass, J. K. Miller, M. D. Soo, N. Kline, and C. S. Jensen. Integrating multiple calendars using TauZaman. *Software-Practice Experience*, to appear, 2007.

[49] J. Wijsen. A string-based model for infinite granularities. In Spatial and Temporal Granularity: Papers from the AAAI Workshop. *Technical Report WS-00-08*, pages 9–16. AAAI Press, 2000.

[50] R. C.-W. Wong, J. Li, A. W.-C. Fu, and K. Wang. k-anonymity: an enhanced k-anonymity model for privacy preserving data publishing. In *Proc. of the the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 754–759. ACM Press, 2006.

[51] X. Xiao and Y. Tao. Personalized privacy preservation. In *SIGMOD '06: Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 229–240. ACM Press, 2006.

[52] X. Xiao and Y. Tao. M-invariance: towards privacy preserving re-publication of dynamic datasets. In *Proc. of the 2007 ACM SIGMOD international conference on Management of data*, pages 689–700. ACM Press, 2007.

[53] M. Youssef, V. Atluri, and N. R. Adam. Preserving mobile customer privacy: an access control system for moving objects and customer profiles. In *Proc. of the 6th international conference on Mobile data management*, pages 67–76. ACM Press, 2005.