Privacy Preserving and Continuous LBS using Grid System

C TEJOSAI RAM A.NARESH KUMAR

1. STUDENT – DEPARTMENT OF CSE,ANURAG ENGINEERING COLLEGE, KODAD, TELANGANA.
2. ASSISTANT PROFESSOR - DEPARTMENT OF CSE ,ANURAG ENGINEERING COLLEGE, KODAD, TELANGANA.

ABSTRACT

The widespread adoption of location-based services (LBS) raises increasing concerns for the protection of personal location information. A common strategy, referred to as obfuscation, to protect location privacy is based on forwarding the LSB provider a coarse user location instead of the actual user location. Conventional approaches, based on such technique, are however based only on geometric methods and therefore are unable to assure privacy when the adversary has semantic knowledge about the reference spatial context. This paper provides a comprehensive solution to this problem. Our solution presents a novel approach that obfuscates the user location by taking into account the semantic knowledge about the reference space. In the paper, we define several theoretical notions underlying our approach. We then propose two different strategies for generating obfuscated spaces. The paper includes several experimental results assessing performance, storage requirements and accuracy for the two approaches. The paper also discusses the system architecture and shows that the approach can be deployed also for clients running on small devices.

1. INTRODUCTION

The ever increasing collection of personal location data, pushed by the widespread use of location-sensing technologies, like satellite positioning systems, RFID and sensors, and the development of location-based services (LBS), motivates the great concern for the protection of personal location information (*location privacy*). The communication of a user's position to a LBS provider upon a service request may result in the unauthorized dissemination of personal location data. Such data, combined with other available information, may in turn lead to the inference of sensitive information about individuals. Various approaches have been thus proposed to assure location privacy. Most of those approaches are based on obfuscation techniques that aim at disguising the actual position of the user by forwarding to the LBS provider fake or less accurate (*generalized*) location information. Approaches based on k-anonymization refine obfuscation techniques by making sure that the generalized location of each user is undistinguishable from the generalized locations of other $k-1$ users.

A common problem to all the above approaches is that they do not take into account semantic knowledge that the adversary may have about the reference spatial context. We claim that by exploiting such knowledge, the adversary may be able to obtain more precise bounds about the actual user location (referred to as *location inference*), thus defeating the obfuscation mechanism. Another major drawback of such approaches is that they do not support location privacy preferences, that is, the specification of which locations are sensitive for which users. Not all locations may have the same sensitivity for all the users and therefore a suitable obfuscation mechanism should be able to generate obfuscation locations that are tailored to the privacy preference of each user. We believe that as we move toward more personalized LBS, privacy should be one of key personalization dimensions. Before moving to introduce the key contribution of the paper, we introduce a running example to illustrate the location inferences that are possible when spatial semantic knowledge is available to the adversary.

Example 1. *Assume that a user of a LBS is located within a hospital which for this user is a sensitive place, because the user is a patient of this hospital. Consider the following knowledge about the geographical context: the hospital is close to a lake and to a residential district; all these places, i.e. the lake, the district and the hospital, cover a polygonal region. Suppose that no boats are allowed on the lake. Assume also that the actual position of the user is obfuscated by a region containing the user's position (obfuscated location). Now suppose that an adversary has such a knowledge. From the observation of the spatial relationships existing between the obfuscated location and the spatial entities, like spatial containment, overlaps and disjointness, the adversary can easily infer whether the user is located in a sensitive place.*

In particular consider the following three cases: a) The obfuscated location is spatially contained in the extent of the hospital. In this case, the adversary may easily infer that the user is located in a sensitive place, that is, the hospital, although the actual position is blurred to a coarser region. b)The region corresponding to the user's obfuscated location includes the extent of both the hospital and the lake. Since the user cannot be physically located inside the lake, because no boats are allowed on the lake, the only realistic position is within the hospital and thus the obfuscated location is still sensitive. Notice that in this case information about the user's obfuscated location is combined with publicly available information, i.e., that no boat is allowed on the lake, in order to infer more precise information about the actual location of the user. c)The region corresponding to the user's obfuscated location overlaps part of the hospital and part of the residential district. Since the hospital is the only sensitive place, we can say that the obfuscated location is "sensitive to some extent". Suppose now that the user is not a patient of the hospital, but a physician. In such case the fact of being located in the hospital is likely not to be

sensitive for this user.

The example emphasizes the fact that a location, besides a geometric shape, has a qualitative meaning which depends on the entities spatially related to such location. The example clearly shows that privacy breaches occur because existing obfuscation techniques are unable to protect against the inferences made by linking the geometric information with the location meaning which, depending on the perceptions of users, may represent sensitive information. The protection of sensitive location information thus requires techniques able to take into account the qualitative context in which users are located as well as their privacy preferences. To our knowledge, the problem of location privacy inferences that exploit spatial semantic knowledge has not been yet addressed.

To address such gap, we have developed the PROBE (Privacy-preserving Obfuscation Environment) system. PROBE is able to obfuscate locations by taking into account spatial semantic knowledge and user preferences. The PROBE approach is based on a notion of *personal location privacy preference*, that is, a set of locations that are considered sensitive by an individual; for each such location, the individual may also specify the sensitivity level. A key component of PROBE is the privacy model, which we also present in this paper; such a model allows one to estimate the probability that an adversary may infer the presence of an individual inside a location which is sensitive for this individual. Such model takes also into account semantic knowledge about locations. Based on such model, PROBE generates *personalized obfuscated maps* that are then used to obfuscate the user location when the location has to be sent to the LBS. The generation of such maps is a key technical issue that we also address in this paper. We propose two different approaches for the generation of such maps. In particular we define two algorithms, called $SensFlow_{Pyr}$ and $SensFlow_{Hil}$, based respectively on the pyramid data structure and the Hilbert space-filling curves. Those data representation techniques have been also used in the framework of k-anonymity; their application to the framework of location obfuscation is a novel contribution of our work.

We have carried out an extensive experimental analysis to assess various aspects of the map generation techniques, such as performance and storage requirements. Our experiments show that our approach is efficient and requires low storage; as such, those maps can be stored in a large variety of mobile devices, such as cellular phones. In the paper we also describe an architecture showing how easily our technique can be deployed for very large scale user populations. The rest of the paper is organized as follows. Next sec- tion discusses related work. Then we introduce the location privacy model and in the following section the strategy for the generation of the obfuscated map; next the two algorithms, $SensFlow_{Pyr}$ and $SensFlow_{Hil}$ are presented; the experimental evaluation is reported in the subsequent section. Then we introduce the architecture of the PROBE system. Future plans are finally presented in the concluding section.

2. RELATED WORK

The most recent work on location privacy comprises two main categories of techniques, focusing respectively on the obfuscation of location information, and on the notion of location k-anonymity. The first category includes the approaches which assume that the user's identity must be known to the LBS provider, for example for accounting purposes [1, 5, 3]. The basic idea underlying these approaches is to hide the actual user's position by forwarding to the LBS provider a coarse geometric information. The semantics of space is not, however, taken into consideration. As such those approaches do not protect against location inference attacks, which is the focus of our approach.

The second category of related approaches focuses on the problem of computing k-anonymous locations. The concept of k-anonymity was initially proposed for relational databases [16], and then transposed to the LBS context as follows: the location attribute is treated as a *quasi-identifier*, that is as an attribute that though not containing an explicit reference to the individuals identity, can be easily linked with external data sources and thus reveals who the individual is. Hence, a request is *location k-anonymous* if the user's location is undistinguishable form the location of other k-1 individuals. Finally a *generalized location* is a region containing the position of k individuals. Various location generalization techniques have been proposed [6, 7, 11, 8]. We discuss in some detail the techniques proposed by Mokbel et al. in the context of the Casper system [11] and by Kalnis et al. [8] because they are related to the obfuscation methods we propose. In the approach adopted by the Casper system, space is discretized in a grid of cells which are organized in a pyramid data structure. In addition, a hash table allows one to directly locate the user. Such table contains the pointer to the lowest-level cell in the pyramid in which each user is located and his/her *privacy profile*. A privacy profile is defined as the pair (k, A_{Min}) where k means that the user wishes to be k-anonymous, and A_{Min} is the minimum acceptable area of the generalized location. The location generalization algorithm works bottom-up; if a cell, or combination of two adjacent cells, does not satisfy the privacy preferences, the algorithm is recursively executed with the parent cell until a valid cell is returned. Kalnis et al. [8] have developed the *Hilbert Cloak* algorithm which applies to a grid of cells represented as Hilbert space-filling curve. Users' positions are contiguous in the ordering and thus in space; they are grouped in intervals enclosing k users. Therefore each user belongs to a unique interval and all users in the same interval have the same generalized location.

We have used the pyramid and the Hilbert space-filling curve to define the two obfuscation algorithms. There is, however, a major difference between our approach and those above k-anonymity algorithms. In PROBE the goal is to generalize a region using a sensitivity metrics which takes into account the semantics of space, while k-anonymity techniques aggregate cells in larger regions based only on the position of individuals and geometric criteria.

Recent work on relational data privacy has pointed out that k-anonymity does not ensure a sufficient protection against other kinds of privacy attacks which originate because of the availability of background knowledge [10, 9]. Another criticism is that k-anonymity does not take into account personal anonymity requirements on the acceptable values of sensitive attributes [17]. In the framework of LBS, concerns about the limitations of k-anonymity are more recent [2, 15]. However none of those papers propose obfuscation techniques able to protect against location inference attacks. We, in a previous paper [4], have been the first to

raise the problem of protecting sensitive locations and propose a privacy model. Our previous approach has however many limitations. The notion of privacy preference is difficult to use in practice; the model assumes that users have equal probability of being located in any point; moreover the obfuscation technique is not scalable because based on an obfuscation algorithm which has a quadratic complexity. The PROBE system is a significant advance over previous work in that the privacy model is based on a sensitivity metrics defined in terms of probability. The system is scalable and based on an architecture which is simple to implement and does not require any third party. The computation of obfuscated maps in PROBE has a lower complexity that in our previous approach and therefore represents a significant advance.

3. THE PRIVACY MODEL

In this section we introduce the privacy model defining the key concepts of sensitivity, privacy preferences and obfuscated map.

3.1 Preliminaries

We first introduce the basic notions used in the rest of the paper. The *reference space* Ω is a possibly bounded and connected area in two-dimensional space. The *geometric objects* located in Ω have a spatial type compliant with standard type systems for GIS applications [13]. Without significant loss in generality we restrict ourselves to consider *regions*, that is, geometric objects of region type. A similar approach, however, can be adopted for spatial objects of linear type, such as road networks.

Spatial types are closed under (appropriately defined) geometric union $\cap^s$, intersection $\cap^s$, difference ($\setminus^s$). We denote with R the set of regions in Ω. The places of interest are described in terms of features. A *feature* describes a real world entity. It has a unique name, for example *Milano*, and a *type*, for example *City* [13]. A feature has a spatial extent of spatial type. In our model, spatial extents are thus regions. Further, we assume features to be spatially disjoint. Note that if two places are the one contained in the other, the corresponding features must be defined so that they do not overlap. For example if a restaurant is within a park, then the extent of the park feature should have a hole in correspondence of the restaurant feature extent. An advantage of our feature-based approach is that spatial features can be stored in commercial spatial DBMSs and easily displayed as maps.

The pair of sets (FT, F) representing respectively feature types and features is referred to as the *geographical database* of the application. We introduce some basic functions used in the rest of the paper. Let $ft \in FT$ be a feature type and $f \in F$ be a feature; the following functions are defined: (1) $I(ft)$ returns the set of features of type ft; (2) $Ext(f)$ returns the geometric extent of feature f; (3) $Cov(ft)$ returns the spatial union of the extents of all features of type ft, that is, $Cov(ft) = \bigcup^s_{f \in I(ft)} Ext(f)$.

3.2 Features classification

Users can specify the feature types that they consider *sensitive* and *unreachable*. A feature type is *sensitive* when it denotes a set of sensitive places. For example if *Religious Building* is a sensitive feature type, then *Duomo di Milano*, an instance of Religious Building, is a sensitive feature. Instead a features type is *unreachable* when it denotes a set of places which for various reasons, such as physical impediment, cannot be accessed by the user. For example, the feature type *MilitaryZone* may be unreachable if the user is a common citizen. We denote with FT_S, FT_{NR} respectively the set of sensitive and unreachable features types for a given user. A feature which is neither sensitive nor unreachable is non-sensitive. Such classification directly extends to regions.

Definition 3.1 (Region classification). *Let r be a region and FT_S, FT_{NR} be respectively the sensitive and unreachable feature types. We say that:*

- *r is sensitive if it contains some sensitive feature or a portion of it, that is, $(\bigcup^s_{ft \in FT_S} Cov(ft)) \cap^s r \neq \emptyset$;*
- *r is unreachable if it is completely covered by unreachable features, that is, $r \setminus^s \bigcup^s_{ft \in FT_{NR}} Cov(ft) = \emptyset$;*
- *r is non-sensitive if it is neither unreachable nor sensitive.*

Example 2. *Based on the running example, consider the features $f_{1...5}$ in Figure 1, where f_1 is a lake, f_2 a hospital, f_3 a shopping mall, f_4 a residential district, and f_5 a wood. Assume that hospitals are sensitive places, lakes are unreachable while the remaining features are non-sensitive. Now consider the regions r_1, r_2, r_3. Region r_2 falls inside the lake f_1, thus it is unreachable; region r_1 overlaps f_2, thus it is sensitive; region r_3 is neither unreachable nor sensitive.*

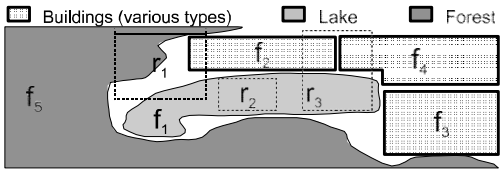

Figure 1: Example of sensitive (r_1), unreachable (r_2), and non-sensitive (r_3) regions.

Given a region r, we introduce function $RC(r) = r \setminus^s \bigcup^s_{ft \in FT_{NR}} Cov(ft)$ which computes the reachable portion of r.

3.3 Sensitivity of a region

We first introduce the probability measure P defined over the set of regions in Ω and ranging in [0,1]. $P(r)$ denotes the probability that a point in Ω falls inside region r. $P(\Omega) = 1$ and $P(\emptyset) = 0$. In case of uniform continuous distribution, the probability is defined as

$$P(r) = \int_r \frac{1}{Area(\Omega)} dxdy = \frac{Area(r)}{Area(\Omega)}$$

where $1/Area(\Omega)$ is the probability density function (pdf) and $Area$ the function returning the surface of the region.

We now estimate the probability that a user, known to be located in an arbitrary region r, is actually located in a sensitive place. Consider a sensitive feature type $ft \in FT_S$. We define *sensitivity of r wrt ft*, denoted as $P_{sens}(ft, r)$, the probability that a user, known to be in r, is actually

within the extent of any sensitive feature of type ft inside r or overlapping with r. Note that if a user is known to be in r, then it is also known that he/she is located in the reachable part of r. Formally:

Definition 3.2 (Sensitivity of a region). *Let $ft \in F T_S$ and $r \in R$. The sensitivity of r wrt ft is defined as:*

$$P_{sens}(ft, r) \equiv P(Cov(ft)|RC(r)) = \begin{cases} \frac{P(Cov(ft) \cap^s RC(r))}{P(RC(r))} & \text{if } RC(r) \neq \emptyset \\ 0 & \text{otherwise} \end{cases}$$

Example 3. *Consider again the features in figure 1, and suppose that the probability distribution is uniform, i.e., the probability associated with a region is proportional to its area. The lower part of region r_3 is covered by an unreachable feature and the sensitive feature f_2 covers approximatively one quarter of r_3. Thus, $P(RC(r_3)) = 0.5 \cdot P(r_3)$, $P(f_2 \cap^s r_3)) = 0.25 \cdot P(r_3)$, and*

$$P_{sens}(Hospital, r_3) = \frac{0.25 \cdot P(r_3)}{0.5 \cdot P(r_3)} = 0.5$$

Therefore, if a user is known to be in r_3, the probability that he/she is inside a sensitive feature is 0.5.

3.4 Privacy preferences

Users can specify "how much private" the obfuscated locations must be, by specifying privacy preferences. Privacy preferences (*preferences* for short) are defined by the *threshold function* $T : F T_S \rightarrow (0, 1)$. Let $T(ft) = v$; we say that v is the *threshold value* of ft. For example, $T(Clinic) = 0.5$ means that in any obfuscated location the probability that an attacker can guess that the user is in a clinic must be equal or less than 0.5.

We say that a region r *satisfies* preference $T(ft) = v$ if the sensitivity $P_{sens}(ft, r)$ is equal or less than v. Note that we do not consider the preference $T(ft) = 1$ because it would mean that ft is not sensitive, against the initial assumption. We also rule out the preference $T(ft) = 0$ because it can be only satisfied if ft has no instances which is not an interesting case. As we will show later on in the paper, users specify the threshold function T in their privacy profile.

3.5 Weakly and strongly privacy-preserving region

A region is privacy-preserving wrt a privacy profile when it is "sufficiently obfuscated" for the user. We distinguish two conditions, referred to as *weakly* and *strongly* privacy-preserving. We say that a region r is *weakly privacy-preserving*, if r satisfies all the preferences specified by the user. Formally:

Definition 3.3 (Weakly privacy-preserving region). *Let $r \in R$ be a region and T a threshold function. r is weakly privacy-preserving wrt T if and only if the following condition holds:*

$$\forall ft \in F T_S, P_{sens}(ft, r) \leq T(ft).$$

Example 4. *With reference to Example 1, consider region 3. It contains part of: a hospital (f_2, type hs), a residential district (f_4, type res) and a lake (f_1, type lake). Assume the following areas: $Area(r_3) = 100$, $Area(f_2 \cap r_3) = 25$, $Area(f_4 \cap r_3) = 20$, $Area(f_1 \cap r_3) = 50$. If $T(hs) = 0.5$, res is non-sensitive, and lake is unreachable then $P_{sens}(hs, r_3) = 0.5$, therefore the privacy preference is satisfied.*

Depending on the privacy requirements, this condition may not however be sufficiently restrictive. Consider the following example:

Example 5. *Consider the feature types ReligiousSite (rs) and Hospital (hs). Let $T(rs) = T(hs) = 0.5$ be the related preferences. Now consider the region r in in Figure 2.b and assume that r is split in two equal parts, one occupied by a religious site (f_2) and the other by a hospital (f_1). The sensitivities of the region wrt rs and hs are thus:*

$$P_{sens}(rs, r) = P_{sens}(hs, r) = \frac{Area(hs)}{Area(r)} = 0.5.$$

According to Definition 3.3, r is weakly privacy-preserving. However, one can immediately observe that, although the feature in which the user is located when in r is not known, the user is certainly located in a sensitive place, i.e. either in the hospital or in a religious building. The only uncertainty that the adversary has regards which specific sensitive feature the user is located in. Depending on the privacy requirements of the user, the disclosure of this information may result into a privacy breach.

For a stronger privacy protection of privacy, we introduce the notion of *strongly privacy-preserving* region. We define the utility function $Ft(r)$ which determines the types of the sensitive features overlapping region r, that is: $Ft(r) = \{ft_i \in F T_S | Cov(ft_i) \cap^s r f = \emptyset\}$.

Definition 3.4 (Strongly privacy-preserving region). *Let $r \in R$ be a region. Let function $\Pi_{sens}(r)$ be defined as follows:*

$$\Pi_{sens}(r) = \begin{cases} P(\bigcup^s_{ft_i \in F T_S} Cov(ft_i)|RC(r))) & \text{if } RC(r) \neq \emptyset \\ 0 & \text{otherwise} \end{cases}$$

r is strongly privacy-preserving if and only if the following condition holds:

$$\Pi_{sens}(r) \leq \min_{ft \in Ft(r)} T(ft).$$

Example 6. *Consider example 5. r is not a strongly privacy-preserving region because r is fully covered by sensitive features and thus in any case the user is located in a sensitive place. Specifically:*

$$\Pi_{sens}(r) = 1 \not\leq 0.5$$

The following theorem shows that a strongly privacy-preserving region is also weakly privacy-preserving wrt each feature type while the vice-versa is not true.

Theorem 3.5. *Let $r \in R$ be a region and T a threshold function. Then:*

$$\Pi_{sens}(r) \leq \min_{ft \in Ft(r)} T(ft) \Longrightarrow \forall ft \in F T_S, P_{sens}(ft, r) \leq T(ft).$$

The reverse implication does not hold.

Proof: Reported in the appendix.

The choice of the privacy-preservation model to adopt, i.e. weak or strong, depends on the privacy requirements of the user.

3.6 Obfuscated map

A set of privacy-preserving regions obfuscating all of the sensitive features in the reference space forms an *obfuscated map*. If the regions in such a set are weakly privacy-preserving then the obfuscated map is *weak* otherwise it is *strong*. Formally:

Definition 3.6 (obfuscated map). *Let Ω be the reference space, and (FT, F) a geographical database. Moreover let:*

- *$FT_S \subseteq FT$ be a set of sensitive feature types;*
- *$FT_{NR} \subseteq FT$ be a set of non-reachable feature types;*
- *T a threshold function.*

Then :

(1) An obfuscation of Ω is a set $S = \{r_1, ..r_n\}$ of regions verifying the following conditions:

- *$\forall i \neq j \in \{1..n\}, r_i \cap^s r_j = \emptyset$, that is, the regions are disjoint*
- *$\forall i \in \{1..n\}, \exists ft \in FT_S, r_i \cap^s Cov(ft) \neq \emptyset$, that is each region overlaps at least one sensitive feature extent*
- *$\bigcup^s_{i\in\{1..n\}} r_i \supseteq \bigcup^s_{ft\in FT_S} Cov(ft)$, that is, the spatial union set of the regions covers the spatial union set of the sensitive features*

(2) A weakly obfuscated map of Ω is an obfuscation S of Ω that verifies the following condition:

$$\forall i \in \{1..n\}, \forall ft \in FT_S, P_{sens}(ft, r_i) \leq \mathrm{T}(ft)$$

(3) A strongly obfuscated map of Ω is an obfuscation S of Ω that verifies the following condition

$$\forall i \in \{1..n\}, \forall ft \in FT_S, \Pi_{sens}(ft, r_i) \leq \min_{ft\in FT(r_i)} \mathrm{T}(ft)$$

We refer to the tuple $< FT_S, FT_{NR}, \mathrm{T} >$ as the privacy profile *associated with the obfuscated map S.*

Example 7. *Consider the features $= \{f_1, f_2, f_3\}$ of type ft_1, ft_2, ft_3 respectively and the set of regions $S = \{r, r_1, r_2\}$ illustrated in Figure 2. Assume the following preferences:* $\mathrm{T}(ft_1) = 0.5$, $\mathrm{T}(ft_2) = 0.4$ *and* $\mathrm{T}(ft_3) = 0.3$.

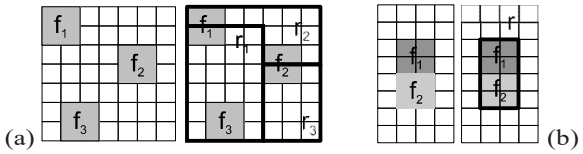

Figure 2: Examples of obfuscation spaces

The set S is a weakly obfuscated map because the following condition is satisfied:

$$\forall n, m \in \{1, 2, 3\}, P_{sens}(ft_n, r_m) = \frac{Area(o_n \cap^s r_m)}{Area(r_m)} \leq \mathrm{T}(ft_n)$$

Moreover S is also a strongly obfuscated map:

- $\Pi_{sens}(r_1) = \frac{Area(f_1 \cap^s r_1) + Area(f_3 \cap^s r_1)}{Area(r_1)} < \mathrm{T}(ft_3) = 0.3$
- $\Pi_{sens}(r_2) = \frac{Area(f_1 \cap^s r_2) + Area(f_2 \cap^s r_2)}{Area(r_2)} < \mathrm{T}(ft_2) = 0.4$
- $\Pi_{sens}(r_3) = \frac{Area(f_2 \cap^s r_3)}{Area(r_3)} < \mathrm{T}(ft_2) = 0.4$

4. COMPUTING THE OBFUSCATED MAPS

We now address the problem of generating the obfuscated maps. The generation of obfuscated maps poses two main issues. The first issue is how to compute a privacy-preserving region. The problem is not trivial because the shape of a privacy-preserving region depends on the distribution of probability P. The higher is the probability that one guesses the location of an individual in a sensitive region, the larger is the area obfuscating such a region. Moreover, an obfuscated area may have an irregular shape; therefore standard spatial operations, such as buffer zoning, which computes an area at a specified distance around a region, are of little usefulness. The second problem is how to limit the loss of geometric precision because an obfuscated position can be arbitrarily large and that may compromise the quality of location information.

To address those requirements the key idea is to use a discrete representation of the reference space. In particular the reference space is subdivided in a grid of regular cells of application-dependent size. Given a grid $ttR = \{c1....c_m\}$, the cells are pairwise spatially disjoint, that is, $\forall i \neq j \in \{1, ..m\}, c_i \cap^s c_j = \emptyset$. The spatial union of cells is the whole reference space Ω. The grid ttR is also referred to as the *initial partition*. The sensitivity is measured with respect to cells while the obfuscated locations are computed by aggregating cells around a sensitive cell. A cell which does not satisfy the privacy requirements is said to be *over-sensitive*. Conventionally, we use the term *cell* to identify an element of any partition of the reference space and *base cell* to identify a cell of the initial partition.

We now reformulate the basic notion of obfuscated map, previously introduced in the abstract model, based on the notion of discrete space.

Definition 4.1 (Discrete obfuscated map). *Let $ttR = \{c1....c_m\}$ be the initial partition of Ω and $S = \{r_1...r_n\}$ be an obfuscated map of Ω. We say that S is a* discrete obfuscated map *wrt GR if and only if each obfuscated region is equal to the spatial union of a set of base cells, that is:*

$$\forall i \in \{1..n\}, \exists c_j, \ldots c_{j+k} \in ttR, r_i = \bigcup^s_{t\in\{0..k\}} c_t.$$

Example 8. *The obfuscated map $S = \{r_1, r_2, r_3\}$ in Example 7 is a discrete obfuscated map wrt the grid shown in Figure 2.*

4.1 Outline of the strategy

Finally we reformulate the initial problem of how generating obfuscated maps in the following terms: *given an initial partition and a privacy profile, compute a set S of regions such that S is a* discrete *obfuscated map.* The notions of weakly and strongly obfuscated maps can be extended to discrete obfuscated maps, thus resulting in the notions of weakly and strongly discrete obfuscated maps. For the generation of discrete obfuscated maps, we propose the following strategy. Consider a geographical database (FT, F) and a privacy profile PP. The strategy is articulated in two main phases:

1. Grid initialization. The reference space is discretized in a grid ttR. For each base cell the coverage of each feature type is first computed by using standard spatial operators; then the probabilities $p_i = P(Cov(ft_i))$, $ft_i \in FT$, are computed. This operation needs only to be performed once for each geographical database.

2. Iteration method. Based on privacy profile PP , each base cell c is logically assigned the tuple $< v_1...v_n >$ where v_i, $i \in \{1,..n\}$, is the sensitivity of the cell c wrt ft_i, that is, $P_{sens}(c, ft_i)$. At the first step of the iteration, the initial partition represents the *current partition*. The current partition is then checked to verify whether it represents an obfuscated map. If this is not the case, it means that at least one cell is not privacy-preserving. A cell c is thus selected from the set of cells that are not privacy-preserving and merged with an adjacent cell to obtain a coarser cell. The result is a new current partition. This step is iterated until the solution is found or the partition degenerates into the whole space.

Next subsection introduces some general properties of the iteration method.

4.2 Properties of the iteration method

Let ttR be a grid and C be a partition (not necessarily the initial one) of ttR. Two cells $c_1, c_2 \in$ C are adjacent if they have a common border. Given two adjacent cells c_1, c_2, the operation which merges the two cells generates a new partition C^j in which cells c_1 and c_2 are replaced by cell $c = c_1 \cup c_2$. We say that partition C^j *is derived* from partition C, written as $C^j \preceq C$. Consider the set $P_{C_{in}}$ of partitions derived directly or indirectly from the initial partition C_{in} through subsequent operations of merge. The poset $H = (P_{C_{in}}, \preceq)$ is a bounded lattice in which the least element is the initial partition while the greatest element is the partition consisting of a unique element, that is, the whole space (called *maximal partition*).

We now show that when two cells are merged, the sensitivity of the resulting cell is lower than the maximum between the sensitivity values of the two starting cells. Then it can be shown that the maximum sensitivity value in a partition C_A (wrt each feature type and each region) is weakly anti-monotonic with respect to the "is derived" relation, that is, that the maximum sensitivity of a partition is equal or greater than the maximum sensitivity of a derived partition. Finally we show that depending on the privacy-preserving model (weakly or strongly), the information about the sensitivity of the reference space Ω can be useful to determine whether an obfuscated map exists. Proof of theorems are reported in appendix.

Theorem 4.2. *Let c_1 and c_2 be two cells of a partition* C, *and $c = c_1 \cup^s c_2$. Then:*

a) $P_{sens}(ft, c) \leq \max\{P_{sens}(ft, c_1), P_{sens}(ft, c_2)\}$.

b) $\Pi_{sens}(c) \leq \max\{\Pi_{sens}(c_1), \Pi_{sens}(c_2)\}$.

Theorem 4.3. *Let C_A and C_B be two distinct partitions of $P_{C_{in}}$ and let $ft \in FT_S$ be a sensitive feature type. Then:*

a) $C_A \preceq C_B \Longrightarrow \max_{r \in C_A} P_{sens}(ft, r) \leq \max_{r \in C_B} P_{sens}(ft, r)$.

b) $C_A \preceq C_B \Longrightarrow \max_{r \in C_A} \Pi_{sens}(r) \leq \max_{r \in C_B} \Pi_{sens}(r)$.

Theorem 4.4. *(A) A weakly obfuscated map exists if and only if the region representing the whole reference space Ω is weakly privacy-preserving. (B) A sufficient, but not necessary, condition for the existence of a strongly obfuscated map is that the reference space is strongly privacy-preserving.*

Given a privacy profile, multiple obfuscated maps can be generated. We consider optimal the obfuscated map S^j with the maximum cardinality, thus possibly consisting of the finest-grained regions. Next section presents two heuristic algorithms for the generation of obfuscated maps.

5. OBFUSCATION ALGORITHMS

The first algorithm is referred to as $Sens_{Pyr}$. The basic idea of such algorithm is to represent the grid of cells using a pyramid data structure similar to the structure used for k-anonymization in the Casper system [11]. The goal of Casper, however, is different, in that its goal is k-anonymize a position of the mobile user requesting a LBS service, whereas the goal of our approach is to compute the set of regions obfuscating all sensitive cells.

5.1 Algorithm $Sens_{Pyr}$

The pyramid data structure takes the form of a tree in which the nodes represent the regions obtained by recursively subdividing space in quadrants until the base cells are reached. The root at level 0 corresponds to the entire reference space; the leaves are the base cells; a node d which is not a leaf node has four children, each representing a quadrant of the region denoted by d.

Given a user privacy profile, the value of sensitivity wrt each feature type can be computed for each cell. The example in Figure 3 shows a space populated by a unique feature of type 'Hospital'. The pyramid has three levels; the sensitivity value of the cells overlapping the hospital extent is reported in the tables associate with the cells. Notice that the sensitivity of any cell c overlapping the hospital, that is, $P_{sens}(c, Hospital)$, is less than 1 because each such cell is not entirely occupied by the sensitive feature.

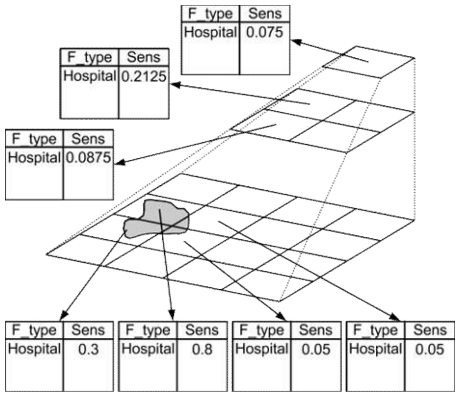

Figure 3: Example of pyramid and associated sensitivity values.

The function computing the obfuscated map is detailed in Algorithm 1. The input parameters of the function are the privacy profile pp and the pyramid pyr. Initially the set tt of privacy-preserving regions is empty. The function then analyzes each leaf node of pyr (line 3). If the node $cell$ is not included in any region in tt, and is not privacy-preserving, the function attempts to generalize the cell with the quadrant g the cell belongs to at the parent level. Note

Algorithm 1 $Sens_{Pyr}$ Algorithm

```
        d Obfuscate the pyramid pyr using privacy profile pp
 1: function pyrObfuscate(pyr, pp)
 2:     tt ← ∅                                   d Obfuscated regions
 3:  for all cell ∈ pyr.leaves do
 4:         if ¬obfuscated(cell, tt)∧overSensitive(cell, pp) then
 5:             g ← generalize(cell, pp)
 6:             if g = ⊥ then
 7:                 return ⊥                     d Cell obfuscation failed
 8:             else
 9:                 add(tt, g)                   d Also remove redundancies
10:             end if
11:         end if
12: end for
13: return tt
14: end function

15: function generalize(cell, pp)
16: if ¬overSensitive(cell, pp) then
17:         return cell                          d cell satisfies pp
18: else if cell = root then
19:         return ⊥                             d cell obfuscation failed
20: else
21:         return generalize(parent(cell), pp)
22: end if
23: end function
```

that we use the term cell generalization and cell obfuscation as synonym. If g is not privacy-preserving, the process is iterated, until a coarser quadrant is found or the top level is reached. If the generalization is successful, g is added to the set tt. The *add* operation, at line 9, removes any previously any previously privacy-preserving region included in g and marks the leaf nodes contained in g as "visited". Therefore the cells in tt are disjoint. The function returns either a failure message or a set of grid cells at different resolution.

The sensitivity of each cell (either leaf or internal) is evaluated at most once. Therefore, as the total number of cells is $O(n)$, the complexity of the algorithm, measured as number of cells is $O(n)$.

An advantage of the algorithm is that the obfuscated regions can be represented in a simple way, for example by the pair (l, n) where l is the pyramid level and n the position inside the grid at that level. This algorithm however suffers from a major drawback, in that the obfuscated regions are very coarse and thus the quality of the obfuscated location information is low. We have thus investigated a different algorithm, referred to as $Sens_{Hil}$, which aggregates cells at the finest granularity and provides more precise obfuscated locations.

5.2 The algorithm $Sens_{Hil}$

$Sens_{Hil}$ maps the initial grid onto a linear space by using a Hilbert space-filling curve [14], which have been previously used in the framework of k-anonymization [11]. Such a curve is a one-dimensional curve which visits every point within a two-dimensional space. Figure 4 illustrates the curves representing grids with 2×2, 4×4, 8×8, 16×16, and 32×32 cells, respectively. Cells are identified by an integer value corresponding to the order of the cells in the traversal; we refer to such an order as *cell sequence*. The interval $[a, b]$ in the linearized space includes the set of cells which are visited starting from the a^{th} cell and ending with the b^{th}. For example in the first grid in Figure 4, the interval $[2, 4]$ denotes the set of cells $\{(0, 1), (1, 1), (1, 0)\}$, corresponding

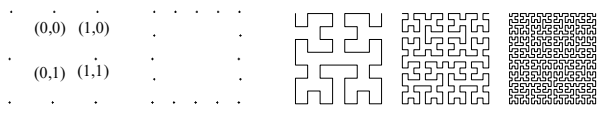

Figure 4: Hilbert curves for different grid sizes

to the 2^{nd}, 3^{rd}, and 4^{th} cells touched by the curve.

The function HilObfuscate, generating an obfuscated map, is detailed in Algorithm 2. The algorithm consists of two phases. The first phase is called *forward generalization*. The algorithm starts scanning the cell sequence from the first cell. As an over-sensitive cell *cell* is found, the algorithm attempts to generate a privacy-preserving interval g starting from *cell*. If such interval is found, g is inserted into the result set tt and the scan proceeds until all cells have been examined. Upon completion of the scan, it may happen that the last sensitive cell cannot be generalized, because, for example, represents the last cell in the cell sequence. If this is the case, the algorithm expands the current interval backwards until a convenient interval is found or the entire sequence of cells is scanned again from the last cell to the first one. This phase is called *backward generalization*. Note that in order to ensure that intervals are disjoint, the addition of g to tt in the backward phase through the *add (G,g)* operation, at line 31, may entail a change of the set tt. For example, the operation $add(\{[1, 2], [4, 7], [8, 9]\}, [5, 12])$ results into the set $tt = \{[1, 2], [4, 12]\}$. Each cell is examined at most twice, once per phase. The overall complexity of the $Sens_{Hil}$ algorithm is, thus, $O(n)$.

5.3 Property of the algorithms

It can be shown that if $Sens_{P\ yr}$ and $Sens_{Hil}$ do not fail, the set of regions they return is an obfuscated map. Formally:

Theorem 5.1. *Let ttR be the initial partition and pp a privacy profile. The non-empty set PyrObfuscate(ttR, pp) and the non-empty set HilObfuscated(ttR, pp) are obfuscated maps.*

Proof. (Sketch) We need to show that the non-empty set of discrete regions returned by each function satisfies the definition of discrete obfuscated map. Consider first the former algorithm. The regions of the obfuscated space are quadrants at different granularities.

6. EXPERIMENTAL EVALUATION

In this section we report results from the experimental evaluation of the two obfuscation algorithms. The algorithms have been developed in Java. The implementation of $Sens_{Hil}$ relies on the library of C-functions available from [12]. The experiments were run on a laptop PC equipped with an AMD Turion Mobile MT30 1.6GHz CPU, 1,37GB of RAM and Windows XP. The current version assumes an uniform continuous probability distribution.

Publicly available data sets about places are not sufficiently accurate for our purposes, because places are spatially represented in terms of points and not of regions. We have thus developed a grid initialization tool, referred to as *Spatially-aware Generalization* tool (SAG, for short). SAG enables the generation of grids randomly populated by features of user-defined type. Features have a rectangular

Algorithm 2 $Sens_{Hil}$ Algorithm

```
                                  d Obfuscate grid using privacy profile pp
 1: function hilObfuscate(grid, pp)
 2:     tt ← ∅                                          d Obfuscated regions
 3:     for idx ← 0 . . . maxHilbertIdx(grid) do        d Hilbert scan
 4:         cell ← getHilbertCell(idx)
 5:         if overSensitive(cell) then
 6:             g ← generalizeForward(idx, grid, pp)
 7:             add(tt, g)
 8:             idx ← g.last
 9:         end if
10: end for
11: fixBackward(G,pp) d Fix the last interval if needed
12: return tt
13: end function

14: function generalizeForward(startIdx, grid, pp)
15:     g ← [startIdx, startIdx]                        d Result
16: for idx ← startIdx . . . maxHilbertIdx(grid) do
17:         g.end ← idx                                 d Expand current interval
18:         if ¬overSensitive(g, pp) then
19:             return g
20:         end if
21: end for
22: return g
23: end function

                d Backward expansion if the last interval g violates pp
24: procedure fixBackward(tt, pp)
25: g ← last(tt)
26: if overSensitive(g, pp) then
27:         g ← generalizeBackward(g, grid, pp)
28:         if overSensitive(g, pp) then
29:             tt ← ⊥                                  d Obfuscation failed
30:         else
31:             add(tt, g)                              d Also remove redundancies
32:         end if
33: end if
34: end procedure

35: function generalizeBackward(g, grid, pp)
36: for idx ← g.first − 1 . . . 0 reverse do
37:         g.first ← idx                               d Expand backward
38:         if ¬overSensitive(g, pp) then
39:             return g
40:         end if
41: end for
42: return g
43: end function
```

shape, of varying size, and are represented as group of cells. Each cell c is either empty or completely covered by a feature of feature type ft. Thus, $P_{sens}(ft, c) \in \{0, 1\}$. The operation of grid initialization requires the following parameters: a) the number of rows and columns of the grid; b) the set FT of feature types - a feature type is defined by simply specifying its name; c) for each type $ft \in FT$, the percentage of cells covered by ft, that is, sensitive cells, over the total number of grid cells.

The system populates the space with rectangles of varying size. The side of each rectangle is generated on a random basis according to a binomial distribution with parameters n=6 (number of independent experiments) and p=0.5 (probability of each experiment). The average size of the rectangle side is 3 cells while the values range in the interval [0,6].

Another functionality of SAG is to enable the specification of privacy profiles. The user flags the features types in the set FT that are unreachable and sensitive and then specifies for each sensitive feature the threshold value. SAG enables

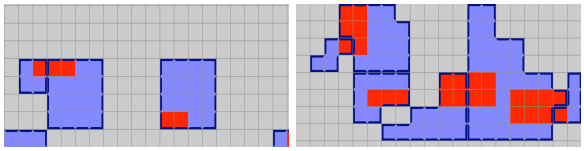

Figure 5: Obfuscated regions created by: (left) $Sens_{P\ yr}$, (right) $Sens_{Hil}$

the selection of the obfuscation algorithm for the generation of the obfuscated map. The parameters are specified through an end-user interface.

SAG also supports the visualization of the obfuscated maps returned by the obfuscation algorithm. From Figure 5 one can appreciate the different shapes of the obfuscated regions generated by $Sens_{P\ yr}$ and $Sens_{Hil}$; the dark grey cells (red in the color version of the paper) are the oversensitive features, while the lighter grey cells (blue in the color version of the paper) fill the obfuscated regions. The cells in the background are non-sensitive cells. Moreover the features, that we recall are represented as group of cells, may be obfuscated by more than one region. In other words, an obfuscated region may obfuscate a portion of a sensitive feature and not necessarily the whole feature. This aspect is important because it allows one to reduce the width of obfuscated regions.

6.1 The experimental setting

We have carried out two sets of experiments. The first set of experiments measures the following parameters for varying percentages of coverage over a grid of fixed size: the rate of the successful generation of maps, the average number of obfuscated locations, and the average number of cells per obfuscated location. The second set of experiments evaluates the *generalization time*, that is, the time spent for generating an obfuscated map (when such a generation is successful) for grids of varying-size and varying percentage of coverage, and the comparison between strongly and weakly obfuscated maps.

6.1.1 Experiments with varying coverage percentage

We use a grid of size 1024×1024 cells for almost all the experiments. At a resolution of 10 metres, the reference space is thus about $10km \times 10km$ which is the size of an average city. We consider a unique sensitive feature type. The independent variable of the experiments is the percentage of space covered by sensitive cells, that is, the coverage. Such a variable ranges in the interval $[1, 45]$, which seems a reasonable choice; a value of x means that the percentage of cells representing portions of sensitive features is $x\%$. We recall that the feature extent varies between 0 and 36 cells, with an average of 9 cells. Further, we consider three possible values for the threshold function, that is, $T(ft) \in \{0.1, 0.2, 0.4\}$. Each algorithm is run 100 times, for different values of the coverage and the threshold value. We have carried out three experiments.

Experiment 1: Success rate. We evaluate the rate of successful generation of obfuscated maps (*success rate*). Generating an obfuscated map means to satisfy the privacy constraints specified in the user profile, based on the spatial distribution of sensitive places. As the density of sensitive locations and the strength of privacy requirements increase,

the probability of failure in the map generation increases. The graphs in Figure 6 show that the generation is successful until the percentage of coverage is below a *breaking* value. When the coverage value is higher that such value, a solution cannot be found. For example, when the threshold has value 0.4, the breaking value is nearly 40. It can be noticed that the breaking values are nearly the same for the two algorithms.

Experiment 2: Average number of obfuscated regions. We evaluate the average number of obfuscated regions returned by the two algorithms when the map generation process does not fail. The two graphs in Figure 7 show that the number of obfuscated regions generated by $Sens_{Hil}$ is significantly higher than the number of obfuscated regions generated by $Sens_{Pyr}$. It can be noticed that the cardinality increases up to a maximum value and then decreases. The reason of such behavior is that for low percentages of coverage, the number of cells to obfuscate is relatively low. The number of obfuscated regions however increases up to a maximum value. When the density of sensitive cells is too high the algorithms generate large obfuscated areas and thus the number of obfuscated regions globally decreases.

Experiment 3: Average size of the obfuscated location. The goal of this experiment is complementary to the goal of the previous experiment, in that this experiment adds the information about the average number of cells in an obfuscated region and thus about the precision of the obfuscated regions. Not surprisingly, the graphs in Figure 8 show that the $Sens_{Hil}$ generates more precise obfuscated maps than $Sens_{Pyr}$. Quantitative values are reported later on. It can be noticed that the size of the obfuscated maps grows very rapidly, especially for $Sens_{Pyr}$ as the percentage of coverage becomes closer to the breaking point. Figure 11 visualizes the different precision of the obfuscated maps generated by the two algorithms using a grid 64 × 64 with a percentage of coverage ranging between 5 and 35.

6.1.2 *Experiments on grids of varying size*

Table 1 and table 2 report the measures resulting from the experiments carried out using $Sens_{Pyr}$ and $Sens_{Hil}$, respectively, over grids of increasing size ranging between 64 × 64 and 4096 × 4096 with a 10% coverage. Each table row specifies: the grid size (GridSize), the average number of obfuscated regions (NReg), the average number of cells per regions (NCell), and the generalization time (GenTime). If we look at the experiments over a grid of 1024 × 1024 we observe that:

- The number of obfuscated regions generated by $Sens_{Hil}$ is about 40% higher than the number of regions generated by $Sens_{Pyr}$.
- The average number of cells per obfuscated regions in $Sens_{Hil}$ is 46 against 118 of $Sens_{Pyr}$. At the given resolution ($10m \times 10m$) the average area of the obfuscated region generated by $Sens_{Hil}$ is thus $4600m^2$.
- The generalization time for $Sens_{Hil}$ is 177 ms against 185 ms of $Sens_{Pyr}$. The performance is thus not significantly different.

Further the two graphs in Figure 9 show that the generalization time increases linearly with the size of the grid for both algorithms. Moreover, such a time is not significantly affected by the coverage for coverages that are not close to the breaking point. Note that the generalization time is high (few seconds) when the grid is 4096 × 4096. Consider, however, that these experiments have been run on a laptop.

GridSize	NReg	NCell	GenTime(ms)
64 × 64	28	112	0.49
128 × 128	106	120	2.08
256 × 256	429	118	11.5
512 × 512	1726	118	72
1024 × 1024	6943	117	185
2048 × 2048	27735	117	758
4096 × 4096	111026	117	3255

Table 1: Measures for $Sens_{Pyr}$

GridSide	NReg	NCell	GenTime(ms)
64 × 64	43	46	0.49
128 × 128	175	47	2.17
256 × 256	700	47	8.8
512 × 512	2835	46	31
1024 × 1024	11372	46	177
2048 × 2048	45483	46	543
4096 × 4096	181920	46	2104

Table 2: Measures for $Sens_{Hil}$

The experiment reported in Figure 10 compares, for both algorithms, the number of regions in weakly and strongly obfuscated maps generated from the same profile for varying grid sizes. We consider two feature classes, ft_1 and ft_2, with coverage $Cov(ft_1) = 5\%$ and $Cov(ft_2) = 10\%$ respectively. The graphs in Figure 10 show that strongly obfuscated maps contain larger regions. However, the strongly obfuscated maps generated by $SensFlow_{Hilb}$ are even more precise than weakly obfuscated maps generated by $SensFlow_{Pyr}$.

7. A DISTRIBUTED ARCHITECTURE FOR THE PROBE SYSTEM

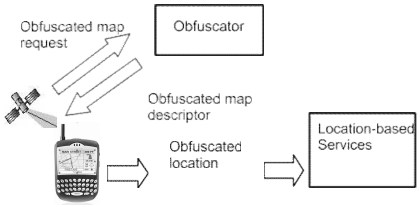

Figure 12: Architecture of the obfuscation system

In this section we discuss how the PROBE system can be deployed in the framework of a LBS architecture. Assume a conventional LBS networked infrastructure consisting of a set of GPS-aware clients which can connect to a LBS provider. The clients are assumed to have computational resources. Moreover we assume to use the $Sens_{Hil}$ algorithm.

PROBE has two main components (see Figure 12): the Obfuscation Server ((*Obfuscator*), in charge of generating obfuscated maps, and the *Privacy Enforcer*, a client application which forwards a possibly obfuscated position to the LBS.

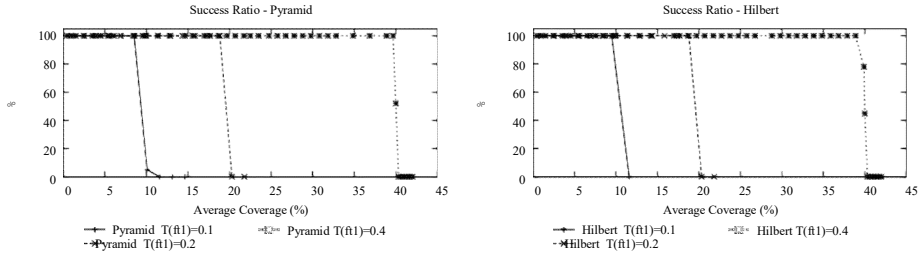

Figure 6: Success rate

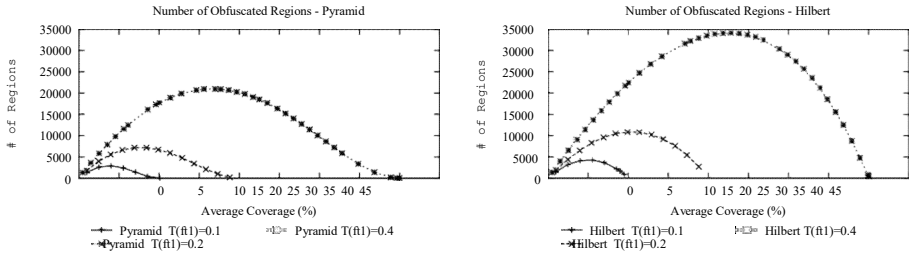

Figure 7: Avg. number of obfuscated regions

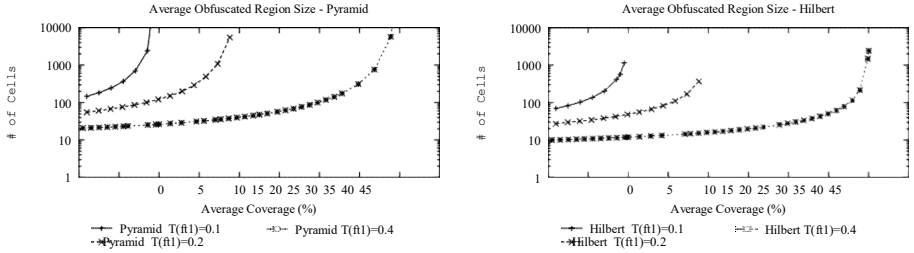

Figure 8: Avg. number of cells per obfuscated region

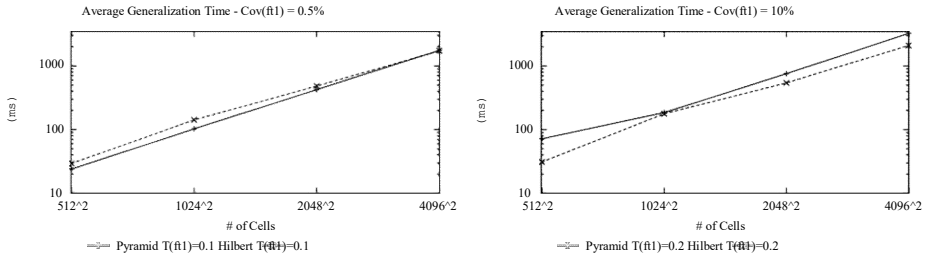

Figure 9: Avg. time for varying-size grids and different coverages

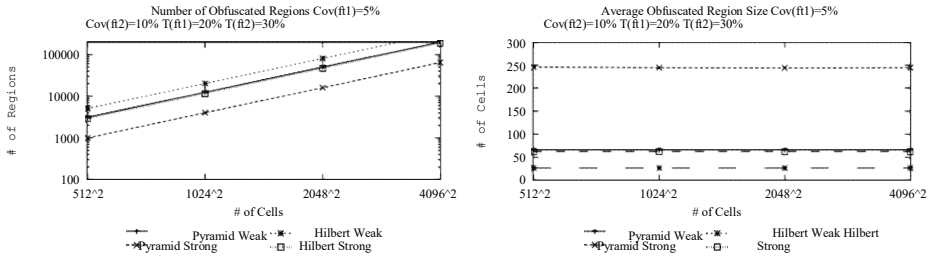

Figure 10: Weakly vs strongly obfuscated maps

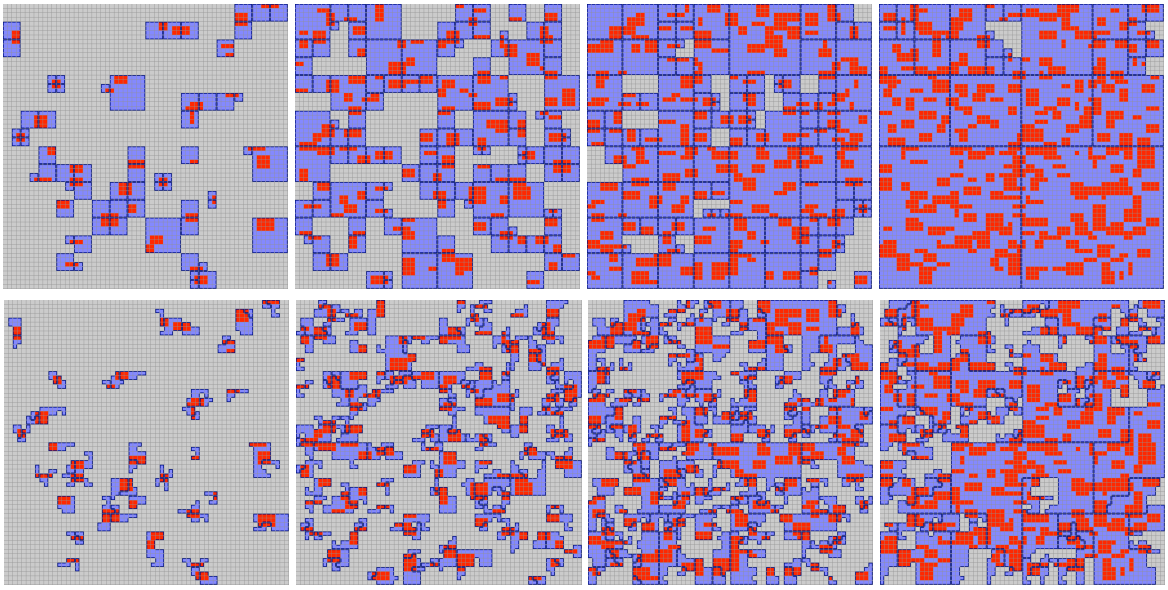

Figure 11: Obfuscated maps generated by $Sens_{pyr}$ (top row) and $Sens_{hil}$ (bottom row) algorithms for coverages ranging from 5% (on the left) to 35% (on the right) with a grid 64 × 64

7.1 The Obfuscator

The Obfuscator generates obfuscated maps upon explicit requests of users. Whenever needed, but likely not frequently, the client forwards to the Obfuscator the request of an obfuscated map. Such a request is accompanied by the privacy profile. We assume that the user, on behalf on whom the client is running, has composed a privacy profile through some Web interface presenting, for example, the list of feature types, even at different granularity, from which the user can select those unreachable and sensitive and possibly specify the threshold value. The Obfuscator has access to a geographical database. If the obfuscated map can be generated then the Obfuscator returns the resulting map to the client which keeps it locally. The Obfuscator must be trusted to generate the obfuscated maps in compliance with clients' requests. Note that the code of such service is very small and therefore it is feasible that the code be verified. The Obfuscator service can be provided by the LBS provider or by a third party. To keep the focus on major aspects, we refer to the former case.

7.1.1 Protocol

Obfuscated Map request. A *Obfuscated Map Request* specifies: a) the privacy profile; b)the bounding box of the area of concern. The privacy profile is a set $PP = \{t_1 ... t_n\}$ of triples representing the privacy preferences. $t_i \equiv < ft, flag, v >$, $i \in \{1..n\}$. If $flag = 0$ then the feature type ft is a sensitive type and $v = \mathrm{T}(ft)$ is the threshold value; conversely ft is an unreachable feature type. In addition, the user specifies the bounding box of the region of interest, for example selecting from a predefined list of varying size regions, in such a way that the the Obfuscator can limit the size of the grid and thus the size of the obfuscated map to generate and transfer.

Obfuscated Map Descriptor The Obfuscator sends back the *Obfuscated Map Descriptor*. Such a descriptor consists of a failure message if the system has not been able to generate a map. Conversely, it specifies: the bounding box of the grid used for the generation of the obfuscated map (*MBB*), the number of rows and columns of the grid (*Dim*) and a non-empty set of obfuscated regions (S). Following the $Sens_{Hil}$ algorithm, an obfuscated region $r \in S$ is represented by an interval $[a, b]$ where a and b are the indexes corresponding to two grid cells in the linear Hilbert space.

Grid size	# Obfuscated locations	≈ *Size(KB)*
512 × 512	2835	22
1024 × 1024	11372	90
2048 × 2048	45483	363
4096 × 4096	181920	1455

Table 3: Avg. size of the obfuscated maps

7.1.2 Transmission of an obfuscated map

We now estimate the size of the obfuscated map transferred back to the client. If the encoding of the interval representing an obfuscated region requires 8 bytes, the size of an obfuscated map is $n \times 8$ bytes, where n is the number of regions in the obfuscated map. Based on the experiments reported the previous section, Table 3 reports the average size of the obfuscated maps generated for grids of varying size, assuming a coverage of %10.

7.2 Privacy Enforcer

The Privacy Enforcer is the client application which computes the location information to be transferred to the LBS provider upon a user's query. The Privacy Enforcement function is reported in Algorithm 3. The function requires in input: the user's position p, and an obfuscated map descriptor, i.e. *MBB*, *Dim* and *S*. The algorithm maps p onto a Hilbert index and then checks if such a position is included in any interval of the set S (line 4). If it is not,

then the function returns the original position p otherwise the interval. The position is then transferred to the LBS provider possibly together with the grid description (MBB, Dim).

Algorithm 3 Privacy Enforcer Algorithm

```
1: function PrivacyEnforcment(p, MBB, Dim, S)
2:     CellCoord ← ttetCellCoord(p, MBB, Dim)      d Returns
   the row and column of the cell containing p
3:     HilbertIndex ← ttetHilbertIndex(CellCoord, Dim)      d
   Returns the Hilbert index of the previous cell
4:     Int = ttetInterval(S, HilbertIndex)          d Returns the
   obfuscated region containing the Hilbert index or null
5:     if Int = ⊥ then
6:         return p                    d Returns the original position
7:     else
8:         return Int                  d Returns the obfuscated location
9:     end if
10: end function
```

8. CONCLUSIONS

PROBE is a comprehensive system for the protection of location privacy against location inference attacks in LBS. A key feature of the system is that it allows the subscribers of a LBS to specify location privacy preferences about the places that they consider sensitive as well as the desired degree of privacy protection. As part of PROBE we have also developed a technique for efficiently computing obfuscated maps that are personalized based on the user privacy preferences. The technique has very small storage requirements and thus it is suited for use on small devices, such as cellular phones.

Our work has many open directions for future research activities. A first research direction is the integration of PROBE obfuscation techniques with k-anonymity. K-anonymity techniques do not protect against location inference. Consider a k-anonymized location. If all k individuals are located inside a hospital and John is known to be one of those individuals, then one can easily infer that John is inside a sensitive location. The research goal is thus how to preserve anonymity while protecting the sensitive location information. A different direction is related to ontological questions about location privacy which has not been investigated yet. For example, the user may wish to hide being *outside* a certain location, rather than being *inside*. Moreover, privacy preferences may also depend on time. For example, one could want to hide his presence at the shop near the workplace during working hours. The third direction is to extend PROBE to the support of trajectory anonymity.

9. REFERENCES

APPENDIX

A. PROOFS OF THEOREMS

A.1 Theorem 3.5

Proof. We show that a region which is strongly privacy-preserving is also weakly privacy-preserving. Case 1) Assume r be unreachable. The following holds: $\Pi_{sens}(r) = 0 \leq \min_{ft \in Ft(r)}\{\mathrm{T}(ft)\}$. On the other hand $P_{sens}(ft, r) = 0$ for any feature type ft and thus the thesis is demonstrated. Case 2) Assume r be reachable and assume that r is strongly privacy-preserving. Since the features of the set $FT_S = \{ft_1, ...ft_j\}$ are disjoint, $\Pi_{sens}(r)$ can be rewritten as:

$$\Pi_{sens}(r) = P_{sens}(ft_1, r) + \ldots + P_{sens}(ft_j, r)$$

Therefore, if $\Pi_{sens}(r) \leq \min_{ft \in Ft(r)}\{\mathrm{T}(ft)\}$ then for each $ft_i \in FT_S$ with $i \in \{1..j\}$, $P_{sens}(ft_i, r) \leq \min_{ft \in Ft(r)}\{\mathrm{T}(ft)\} \leq \mathrm{T}(ft_i)$ and thus the thesis is demonstrated.

The inverse implication is not true as shown in the counter-example 5.

A.2 Theorem 4.2

Proof. We demonstrate only the proposition a), since the latter proposition can be shown in a similar manner. For the sake of readability, we denote with $\bar{x}$ the reachable part of cell x. Note that $\bar{c}_1$ and $\bar{c}_2$ are disjoint regions. Therefore $P(\bar{c}) = P(\bar{c}_1 \cup^s \bar{c}_2) = P(\bar{c}_1) + P(\bar{c}_2)$.

The inequality can be rewritten as:

$$P_{sens}(ft, c) \leq P_{sens}(ft, c_1) \vee P_{sens}(c) \leq P_{sens}(ft, c_2)$$

$$\frac{P(Cov(ft) \cap^s \bar{c})}{P(\bar{c}_1) + P(\bar{c}_2)} \leq \frac{P(Cov(ft) \cap^s \bar{c}_1)}{P(\bar{c}_1)}$$

$$\vee \frac{P(Cov(ft) \cap^s \bar{c})}{P(\bar{c}_1) + P(\bar{c}_2)} \leq \frac{P(Cov(ft) \cap^s \bar{c}_2)}{P(\bar{c}_2)}$$

Consider the first inequality:

$$\frac{P(Cov(ft) \cap^s \bar{c})}{P(\bar{c}_1) + P(\bar{c}_2)} \leq \frac{P(Cov(ft) \cap^s \bar{c}_1)}{P(\bar{c}_1)}$$

. By applying simple algebraic operations we obtain the expression:

$$P(\bar{c}_1)P(Cov(ft) \cap^s \bar{c}_2) - P(\bar{c}_2)P(Cov(ft) \cap^s \bar{c}_1) \leq 0$$ Similarly consider the second term of the inequality,

$$\frac{P(Cov(ft) \cap^s \bar{c})}{P(\bar{c}_1) + P(\bar{c}_2)} \leq \frac{P(Cov(ft) \cap^s \bar{c}_2)}{P(\bar{c}_2)}$$

. We obtain:

$$P(\bar{c}_1)P(Cov(ft) \cap^s \bar{c}_2) - P(\bar{c}_2)P(Cov(ft) \cap^s \bar{c}_1) \geq 0$$

Since one of two expressions must be necessarily true, it follows that the inequality is verified and thus the thesis is demonstrated.

A.3 Theorem 4.3

Proof. Point a). Assume to derive C_B from C_A by applying a single merge operation between two cells. Following Theorem 4.2 the resulting cell c has a sensitivity wrt ft which is equal or less than the highest sensitivity of the replaced cells wrt the same feature type. Therefore, the sensitivity of the derived partition remains the same of the starting partition (if the cell which presents the highest level of sensitivity wrt ft in C_A has not been merged) or it decreases. Consider now a sequence of merging operations each generating an intermediate partition C_A `` $C_1$ `` C_2 `` ...$C_k$ `` C_B. It follows that:

$$\max_{r \in C_A} P_{sens}(ft, r) \leq \max_{r \in C_1} P_{sens}(ft, r) \leq .... \leq \max_{r \in C_B} P_{sens}(ft, r)$$

that is what we wanted to demonstrate. Point b) can be shown with an analogous argumentation. □

A.4 Theorem 4.4

Proof. Point a). If the reference space is weakly privacy-preserving, it constitutes a weakly obfuscated map. Further that map can be always generated because the maximal partition is the greatest element of the lattice $H = (P_{C_{in}}, \text{“}\text{)}$, and therefore can be obtained through successive derivations. Now we show that the condition not only is sufficient but also necessary. Assume per-absurdo that the maximal partition is not weakly privacy-preserving while another partition C_A exists including the regions forming an obfuscated map. Thus, there is at least one feature type ft, such that: $P_{sens}(ft, \Omega) > T(ft)$ while for every cell $r \in C_A$: $P_{sens}(ft, r) \leq T(ft)$. For Theorem 4.2, that is a contradiction and thus the thesis is true.

Point b) We now show through a counter-example that a strongly obfuscated map can exist although the reference space is not strongly privacy-preserving. Assume a uniform continuous distribution of probability. Consider a partition consisting of only two cells, say c_1 and c_2, both with area 10. Assume two feature types ft_1 and ft_2 with $T(ft_1) = 0.1$ and $T(ft_2) = 0.9$. And assume that c_1 contains uniquely instances of ft_1 such that the ft_1 coverage is 10% the cell area while c_2 contains only instances of ft_2 such that the ft_1 coverage is 90% the cell area. The sensitivity of each cells is thus:

- $\Pi_{sens}(c_1) = P_{sens}(ft_1, c_1) = 0.1$
- $\Pi_{sens}(c_2) = P_{sens}(ft_1, c_2) = 0.9$

The partition is thus an obfuscated map. However if we merge the two cells we obtain a cell c which is not privacy preserving: $\Pi_{sens}(c) = P_{sens}(ft_1, c_1) + P_{sens}(ft_2, c_2) = 0.5 \not\geq \min_{ft \in \{ft_1, ft_2\}} T(ft) = T(ft_1) = 0.1$