Protecting Location Privacy through Semantics-aware Obfuscation Techniques

Maria Luisa Damiani, Elisa Bertino, Claudio Silvestri

Abstract The widespread adoption of location-based services (LBS) raises increasing concerns for the protection of personal location information. To protect location privacy the usual strategy is to obfuscate the actual position of the user with a coarse location and then forward the obfuscated location to the LBS provider. Existing techniques for location obfuscation are only based on geometric methods. We state that such techniques do not protect against privacy attacks rooted in the knowledge of the spatial context. We thus present a novel framework for the safeguard of sensitive locations comprehensive of a privacy model and an algorithm for the computation of obfuscated locations

1 Introduction

Location-based services (LBS) and in particular GPS-enabled location services are gaining increasing popularity. Market studies [7] forecast that the number of GPS-enabled mobile devices, including personal navigation devices, cellular handsets, mobile PCs, and a variety of portable consumer electronics devices, will grow from 180 million units in 2006 to 720 million units in 2011.

Mobile users equipped with location-aware devices typically request a LBS service by forwarding to the service provider a query along with the user's position. The service provider then answers the query based on the position. Unfortunately, the communication of the user's position to the service provider raises strong pri-

Maria Luisa Damiani
DICO, University of Milan, Via Comelico 39, 20135 Milan(I), e-mail: damiani@dico.unimi.it

Elisa Bertino
Purdue University, West Lafayette (US) e-mail: bertino@cs.purdue.edu

Claudio Silvestri
DICO, University of Milan, Via Comelico 39, 20135 Milan(I), e-mail: silvestri@dico.unimi.it

vacy concerns because it may result in the unauthorized dissemination of *personal location data*. Such data may in turn lead to the inference of sensitive information about individuals. For example the health status of a service user can be inferred from the nature of the clinics being visited.

Personal location data refers to the association (u, p) between user identifier u and position information p. Protecting *location privacy* means thus preventing u and p from being *both* disclosed without the consent of the user [2]. A well-known approach to the protection of location privacy is to deliberately degrade the quality of location information and forward to the LBS provider an imprecise position. Imprecision may however compromise the quality of service because the answer to the query may result too coarse. Therefore, the imprecise position must be defined at a resolution which is acceptable for the user. We refer to an imprecise user's position as *obfuscated location*.

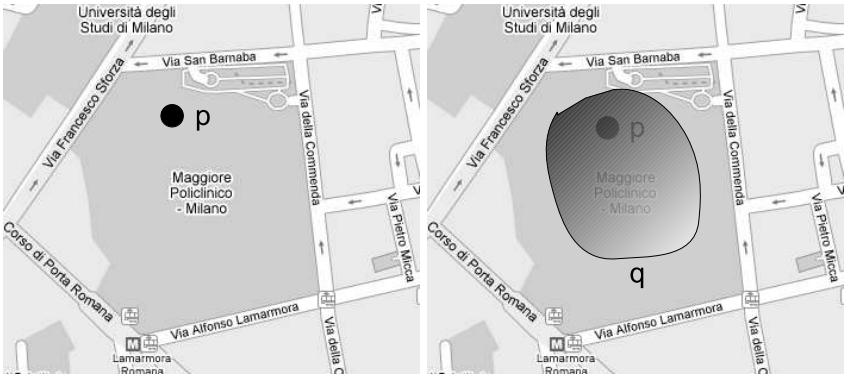
In general, obfuscated locations are computed using techniques, such as (location) *k-anonymity* [6, 10, 8], based on geometric methods. We refer to these techniques as *geometry-based*. We claim that geometry-based obfuscation techniques do not protect against the following simple privacy attack.

Location privacy attack

Assume that John issues a LBS request from position p inside *hospital Maggiore* in Figure 1 (a). John however does not want to disclose the fact of *being inside* the hospital because that might reveal he has health problems. Now assume that location p is obfuscated by region q using some geometry-based technique (Figure 1 (b)). We can observe that if an adversary knows that John is in the obfuscated location q and q is entirely contained in the spatial extent of the hospital (the location of the hospital is publicly known), then such adversary can immediately infer that John is in the hospital. As a result, sensitive information is disclosed against the user consent. Note however that if John would be a doctor, such a privacy concern would not arise because the location would be related to the user's professional activity. We refer to this privacy attack as *spatial knowledge attack*.

The spatial knowledge attack arises because geometry-based obfuscation techniques do not consider the actual semantics of space, namely the spatial entities populating the reference space and their spatial relationships, in other terms the *spatial knowledge*. Therefore those technique are unable to protect against the inferences made by linking the geometric information with the location meaning which, depending on the perceptions of user, may represent sensitive information. The protection of location privacy thus calls for techniques able to take into account the qualitative context in which users are located as well as their privacy preferences.

To address those requirements, we propose a novel location obfuscation framework, that we refer to as *semantic-aware obfuscation system*. The main contribution



(a)Position *p* (b)Obfuscated location *q*

Fig. 1 Example of obfuscated location

of this paper is the definition of the core components of the obfuscation system, that is:

- a privacy model supporting the obfuscation of sensitive locations based on user preferences;
- an algorithm, called *SensFlow* (i.e. Sensitivity Flow), implementing the obfuscation strategy.

The remainder of the paper is structured as follows. Next section overviews related work. Then we present the outline of the approach and the privacy model. The *SensFlow* algorithm and two alternative approaches to space subdivision are discussed in the subsequent section. The final section reporting open issues and research directions concludes the paper.

2 Related work

Recent work on privacy models in LBS comprises two sets of approaches, focused respectively on the protection of location information and on the concept of *k-anonymity*.

Privacy models for the protection of location information

The problem is to how to process the query without knowing the exact location of the user. Atallah at al. [1] have proposed three methods of varying complexity to process nearest-neighbor queries such as *Where is the nearest hospital?* The simplest method is as follows: the client applies a geometric translation to the user's position and forwards the approximated position to the LBS provider. The database

answers the query and returns an imprecise answer. The second method does not result in any accuracy loss but can potentially require more communication. The idea is to subdivide space in a grid of cells. The client queries the database with the tile that contains the client's location. The database answers the query with all spatial objects that are closest to at least one point in the query tile. Upon receiving these objects the client determines which of them is closest to the actual position. The third approach is more efficient and does not require any obfuscation of the user's position. The idea is to determine whether the user's position is contained in a cell of a space subdivision defined as Voronoi diagram without revealing to the database anything other than the Yes/No answer to the question. If the answer is Yes then the object associated with the cell is the one closest to the user. This mechanism, which uses a secure multi-part protocol [4], can be only applied whenever space is partitioned.

Another approach for processing nearest-neighbor queries is proposed by Duckham and Kulik [5]. In such approach the client obfuscates position p by supplying a set P of arbitrary positions including p. The database then answers the nearest-neighbor query by determining the objects that are closest to any point in P. Then, in the simplest case, the database returns the whole set of objects leaving the client to choose among them.

Protection of user identity through k-anonymity

A significant number of proposals are based on k-anonymity. The concept of k-anonymity has been originally defined for relational databases. A relational table T is *k-anonymous* when for each record there are at least (k-1) other records whose values, over a set of fields, referred to as *quasi-identifier*, are equal. A quasi-identifier consists of one or more attributes which, though not containing an explicit reference to the individuals identity, can be easily linked with external data sources and in this way reveals who the individual is. K-anonymity can be achieved by *generalization*, that is replacing a quasi-identifier attribute value with a less specific but semantically consistent value [13]. The concepts of k-anonymity are transposed in the LBS context as follows. The location attribute is treated as a quasi-identifier. 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. Location generalization techniques generate obfuscated locations independent of the query type. The first technique has been proposed by Gruteser. The idea behind this scheme is to recursively subdivide space in quadrants of a *quadtree* [6]. The quadtree is then traversed top down, thus from the largest quadrant covering the whole space, until the smallest quadrant is found which includes the requester and other $k-1$ users. Such a final quadrant constitutes the generalized location.

Another technique based on quadtrees has been proposed in the context of the Casper system [10]. A hash table allows one to directly locate the user. Such table contains the pointer to the lowest-level cell in the quadtree-based data structure in

which each user is located and his *privacy profile*. A privacy profile is defined by the pair (k, A_{Min}) where k means that the user wishes to be k-anonymous, and A_{Min} is the minimum acceptable resolution of the generalized location. The location generalization algorithm works bottom-up: if a cell or combination of two adjacent cells does not satisfy privacy preferences, then the algorithm is recursively executed with the parent cell until a valid cell is returned. Kelnis et al. in [8] observe that location k-anonymity algorithms may compromise location privacy if an attacker knows the generalization algorithm, the value of k and the position of all users. Specifically, this happens when a generalized location can univocally associated with a user. To address this problem, Kelnis et al. present a new algorithm based on the use of a linear ordering of locations.

Recent work on relational data privacy has pointed out that k-anonymity does not ensure a sufficient protection against a number of privacy attacks. For example k-anonymity can generate groups of records that leak information due to the lack of diversity in the sensitive attribute. Such an information leak is called *homogeneity attack*. Against this attack, a possible counter-measure is *l-diversity*. The main idea behind l-diversity is the requirement that the values of the sensitive attributes must be *well represented* in each group [9]. In its simpler form, l-diversity means that each group should have at least l distinct values.

Another criticism against k-anonymity is that it does not take into account personal anonymity requirements on the acceptable values of sensitive attributes. To address this requirement, Xian and Tao [14] introduces the concept of *personalized anonymity*. The main idea is to organize the values of the sensitive attribute in a taxonomy and then let each user specify through a *guarding node* the most specific value of the attribute that the user wants to disclose. Interestingly, this approach attempts to protect the association between a user and the *meaning* of the sensitive attribute, which is close to what we propose. The approach of Xian and Tao, however, only works for categorical attributes.

3 Outline of the approach

The basic idea is to collect users' preferences about sensitive places and the desired degree of location privacy in *privacy profiles* and then carry out the process of location obfuscation in two steps. Such a process is described below. Consider a privacy profile v.

(1) The first step is to obfuscate the sensitive places specified in v based on the user's desired degree of privacy. This operation, that we call *obfuscated space generation*, results in the generation of a set of coarse locations hiding the actual extent of sensitive places in compliance with user preferences. We can abstractly think of obfuscated space generation as the function Obf:

$$Obf(v) = s$$

which maps profile v onto the set s of regions enclosing sensitive places.

(2) The second step is carried out upon a user's LBS request. Consider a user with privacy profile v in position p . The operation that we call *obfuscation enforcement* can be abstractly represented by the function Oe:

$$Oe(p,v) = q$$

mapping position p and profile v onto a location q where $q \in Obf(v)$ if p is contained in q and $q = p$ otherwise. As a result, when the location is obfuscated, an adversary cannot infer with certainty that the user is inside a sensitive (for the user) place. At most one can infer that the position *may be* in a sensitive place.

A naive implementation of the function Obf is to define, for each sensitive place, a region containing the place of interest. This solution has however important drawback: first if the sensitive place has a large extent, the obfuscated location may result too broad and thus compromise the quality of service. By contrast, if the obfuscated location is not large enough the probability of being located inside a sensitive place may be very high and thus obfuscation is ineffective. To overcome these drawbacks we subdivide sensitive regions in cells. Each cell has a sensitivity which depends on the user preferences in the privacy profile. Each cell is thus obfuscated separately through an *obfuscation algorithm*. To represent user preferences, we define a privacy model, called *obfuscation model*, centered on the following concepts.

- *Properties of places*. Places as classified into types. Users specify in their privacy profiles which types of places are *sensitive*, *non-sensitive* or *unreachable*. A place is sensitive when the user does not want to reveal to be in it; a place is unreachable when the user cannot be located in it; a place is non-sensitive otherwise.
- The *level of sensitivity* quantifies the degree of sensitivity of a region for a user. For example a region entirely occupied by a hospital has a high level of sensitivity, if hospital is sensitive for the user. We emphasize that the level of sensitivity depends on the extent and nature of the objects located in the region as well as the privacy concerns of the user.
- An *obfuscated space* is a set of obfuscated locations associated with a privacy profile. Specifically, the locations of an obfuscated space have a level of sensitivity less or equal than a sensitivity *threshold value*. The sensitivity threshold value is the maximum acceptable sensitivity of a location for the user. Since the threshold value is user-dependent, its value is specified in the privacy profile.

4 The obfuscation model

We first introduce the basic nomenclature used in the rest of the paper. A *position* is a point in a two-dimensional space S; *region* is a polygon; *location* broadly denotes a portion of space containing the user's position. Places are represented as *simple features*. A *feature* has an unique name, say *Milano*, and a unique *feature type*, say *City*. Furthermore, a feature has a spatial extent of geometric type [12] that, without

significant loss of generality, consists of a region. Features extents are spatially disjoint. Consider the case of two overlapping places, for example a restaurant within a park: the extent of the park feature must be defined in such a way that it does not contain the extent of the restaurant feature.

An advantage of our approach is that spatial features can be stored in commercial spatial DBMSs and easily displayed as maps. We denote with FT and F respectively the set of features types and the set of corresponding features. Hereinafter we refer to the pair (FT, F) as the *geographical database* of the application.

Sensitive and unreachable feature types

In a privacy profile a user specifies the feature types which are considered 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 *non-reachable* 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 non-reachable if the user is a common citizen. A feature type which is neither sensitive or unreachable is non-sensitive. In principle, a user can define multiple privacy profiles.

Quantifying the level of sensitivity of a region

We introduce first the concept of *sensitivity score* (simply score). The score of a feature type ft is a value which is assigned to ft to specify "how much sensitive" ft is for the user. For example the score of the *restaurant* feature type is typically lower than the score of *hospital* because an individual is usually more concerned with privacy of medical information than with information about his/her preferred restaurants. Formally, the score of feature type ft is defined by the function *Score(ft)* ranging between 0 and 1: value 0 means that the feature type is not sensitive or unreachable while a value 1 means that the feature type has the highest sensitivity. The concept of score captures the subjective perception of the degree of sensitivity. The score of each sensitive feature type is thus specified directly in the privacy profile. The score function is used for computing the sensitivity level of a region.

The *sensitivity level* (SL) of a region *r*, written as $SL_{Reg}(r)$, quantifies how much sensitive *r* is for the user. In particular, SL is defined as sum of the ratios of weighted sensitive area to the relevant area in the region. The *weighted sensitive area* is the surface in *r* occupied by sensitive features weighted with respect to the sensitivity score of each feature type. The *relevant area* of *r* is the portion of region not occupied by unreachable features.

To formally define SL, we introduce the following notation:

- E is the set of regions in the reference space

- (FT, F) is the geographical database, namely the set of features types and features
- $FT_{Sens} \subseteq FT$ is the set of sensitive feature types and $FT_{Nreach} \subseteq FT$ is the set of non-reachable features, with $FT_{Nreach} \bigcap FT_{Sens} = \emptyset$
- The functions: $Area_{Geo}(r)$ and $Area_{Fea}(r, ft)$ compute, respectively, the whole area of *r* and the area of *r* covered by features of type ft. In the latter case, only the portions of features which are contained in *r* are considered.

Definition 1 (Sensitivity level of a region). The sensitivity level of a region is defined by the function: $SL_{Reg} : E \rightarrow [0,1]$ such that, given a region *r*:

$$SL_{Reg}(r) = \sum_{ft \in FT_{Sens}} Score(ft) \frac{Area_{Fea}(r, ft)}{Area_{Rel}(r)}$$

where $Area_{Rel}(r) = Area_{Geo}(r) - \sum_{ft \in FT_{Nreach}} Area_{Fea}(r, ft)$. If *r* only contains non-reachable features, we define $SL_{reg}(r) = 0$. $\diamond$

Example 1. Consider a space consisting of four regions c_0, c_1, c_2, c_3; the set of sensitive feature types is $FT_{sens} = \{ft_0, ft_1, ft_3\}$ and the set of non-reachable feature types is $FT_{Nreach} = \{ft_2\}$. Table 1 reports, for each feature type ft_i and region c_j, the area occupied by ft_i in c_j, with i, j ranging over $\{0,1,2,3\}$. In addition, the row *NS* reports the non-sensitive area in each region. For example, region c_2 includes sensitive features (or portion) of type ft_0 and of type ft_3 both covering an area of 100 units; non-reachable features (or portion) of type ft_2 covering an area of 1000 units; and a non-sensitive area of 100 units. The row $Tot_{relevant}$ reports the *relevant* area in each region, that is the area not covered by unreachable features. For example the relevant area in region c_2 has an extent of 300 units. The last column on the right reports the sensitivity score assigned to each feature type.

$Area(c, ft)$	c_0	c_1	c_2	c_3	$Score(ft)$
ft_0	200	0	100	0	0.5
ft_1	100	0	0	100	0.7
ft_2	300	50	1000	400	0
ft_3	0	100	100	100	0.9
NS	0	0	100	0	-
$Tot_{relevant}$	300	100	300	200	-
SL_{reg}	0.57	0.9	0.47	0.8	-

Table 1 Area and sensitivity scores of feature types

The sensitivity level for regions c_0 and c_1 is:

- $SL_{reg}(c_0) = 0.5 \cdot \frac{200}{300} + 0.7 \cdot \frac{100}{300} = 0.57$
- $SL_{reg}(c_1) = 0.9 \cdot \frac{100}{100} = 0.9$.

It results that region c_1 is more sensitive than c_0. The motivation is that users located in region c_1 are certainly located in the extent of a feature of type ft_3, which has a high sensitivity score.

Obfuscated space

Finally we introduce the concept of obfuscated space. An obfuscated space is a space partition consisting of regions which are *privacy-preserving*. We say that a region *r* is privacy-preserving when the level of sensitivity of *r*, $SL_{Reg}(r)$ is equal or below a threshold value. The *threshold value* is the maximum acceptable sensitivity of locations for the user. Its value ranges in the interval (0,1]. A value equal to 1 means that the user does not care of location privacy in any point of space. We rule out the value 0 because it would be satisfied only if there were no sensitive locations (against the initial assumption). The threshold value is another parameter specified in the privacy profile. We formally define the notion of obfuscated space and of privacy profile in the definition below.

Definition 2 (Obfuscated space). Let (FT, F) be the geographical database. Moreover let:

- $FT_{Sens} \subseteq FT$ be a set of sensitive feature types.
- $FT_{Nreach} \subseteq FT$ be a set of non-reachable feature types.
- *Score* be the score function.
- $\theta_{sens} \in (0,1)$ be the sensitivity threshold value.

Then:

(1)An *obfuscated space OS* is a space partition such that:

$$\max_{c \in OS} SL_{Reg}(c) \leq \theta_{sens}$$

(2)The *privacy profile* associated with *OS* is the tuple

$$< FT_{Sens}, FT_{Nreach}, Score, \theta_{sens} >$$

Example 2. With reference to example 1, consider the profile:

- $FT_{Sens} = \{ft_0, ft_1, ft_3\}$ where ft_0 represents night clubs, ft_1 religious buildings and ft_3 clinics.
- $FT_{Nreach} = \{ft_2\}$ where ft_2 represents a military zone
- $Score(ft_0) = 0.5$, $Score(ft_1) = 0.7$, $Score(ft_2) = 0$, $Score(ft_3) = 0.9$
- $\theta_{sens} = 0.9$

Consider the four regions $\{c_0, c_1, c_2, c_3\}$ and the sensitivity level of each of them (reported in Table 1). It can be noticed that such value, in all cases, is less or equal than θ_{sens}. Thus, the set of regions is an obfuscated space.

5 Computing the obfuscated spaces

After presenting the privacy model, the next step is to define how to compute an obfuscated space. Our strategy consists of two main steps:

1. Specification of the initial partition. The reference space is subdivided in a set of small regions, referred to as cells, which constitute the initial partition denoted as C_{in}. The granularity of the initial partition, that is, how small the cells are, is application-dependent.
2. Iteration method. The current partition is checked to verify whether the set of cells is an obfuscated space. If not, it means that at least one cell is not privacy preserving. A cell c is thus selected among those cells which are not privacy-preserving and merged with an adjacent cell to obtain a coarser cell. The result is a new partition. This step is iterated until the solution is found, and thus all privacy preferences are satisfied or the partition degenerates into the whole space.

In the following we describe these two steps, starting from the latter.

5.1 The iteration method

Consider a partition $\mathscr{C}$ of the reference space. Given two adjacent cells $c_1, c_2 \in \mathscr{C}$, the *merge* of the two cells generates a new partition $\mathscr{C}'$ in which cells c_1 and c_2 are replaced by cell $c = c_1 \cup^S c_2$ with $\cup^S$ denoting the operation of spatial union. We say that partition $\mathscr{C}'$ *is derived* from partition $\mathscr{C}$, written as $\mathscr{C}' \succcurlyeq \mathscr{C}$. Consider the set $P_{\mathscr{C}_{in}}$ of partitions derived directly or indirectly from the initial partition $\mathscr{C}_{in}$ through subsequent merge operations. The poset $H = (P_{\mathscr{C}_{in}}, \succcurlyeq)$ 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*).

It can be shown that an obfuscated space, if it exists, can be generated by progressively aggregating cells in coarser locations and thus by deriving subsequent partitions. The demonstration, that we omit, is articulated in two steps. First it is shown that the *SL* (i.e. sensitivity level) of the cell resulting from a merge operation is less or equal the sensitivity level of the starting cells. Then it is shown that the sensitivity level of the partition (i.e. the maximum sensitivity value of cells) resulting from subsequent merge operations is less or equal than the sensitivity level of the starting partition.

The algorithm

The algorithm computes the obfuscated space by progressively merging adjacent cells. In general, for the same privacy profile, multiple obfuscated spaces can be generated. We consider *optimal* the obfuscated space with the maximum cardinality, thus possibly consisting of the finest-grained regions. The problem of finding the optimal obfuscated space can be formulated as follows:

Given an initial partition $\mathscr{C}_{in}$, determine, if it exists, the sequence of merge operations such that the resulting partition $\mathscr{C}$ is the obfuscated space with the maximum number of cells

In this paper, we present an algorithm which computes an approximated solution to the problem. The idea is to progressively expand each cell which is not privacy preserving until a terminating condition is met. This approach raises a number of issues. The first issue is how to choose the cells to be merged. We adopt the following heuristic: we select the adjacent cell which determines the most sensible reduction of sensitivity of the aggregated cell. A second issue concerns the criteria for the expansion of cells. To address such issues, we have identified two basic strategies: the first strategy is to expand one over-sensitive cell (i.e. a non-privacy-preserving cell) at a time, until the level of sensitivity is below the threshold; the second strategy is to expand "in parallel" all cells which are over-sensitive. The second strategy is the one which has been adopted because it allows one to better control the size of the aggregated cells.

Insights on the *SensFlow* algorithm

We represent a space partition through a *Region Adjacency Graph* (RAG)[11]. In general a RAG is defined from a partition by associating one vertex with each cell and by creating an edge between two vertices if the associated cells share a common boundary. Within this framework, the edge information is interpreted as possibility of merging the two cells identified by the vertices incident to the edge. Such a merge operation implies to collapse the two vertices incident to the edge into one vertex and to remove this edge together with any double edge between the newly created vertex and the remaining vertices [3].

The input parameters of the algorithm are: 1) the initial RAG built on the initial partition; 2) the privacy profile. The algorithm returns an obfuscated space if it exists, an error otherwise. Starting from the RAG corresponding to the initial partition, the algorithm shrinks the graph by merging adjacent cells until all privacy constraints are satisfied or a solution cannot be found. At each iteration, the algorithm looks for non-privacy preserving cells; then each of such cells is merged with at most one adjacent cell. Among the cells in the neighborhood, merging is executed with the cell which determines the most significant reduction in the sensitivity of the resulting aggregated region. After the merge, the algorithm proceeds to scan the remaining cells, and the whole loop is repeated until no cell is modified. The complexity of the algorithm, evaluated with respect to the two key operations, that is (a) merge operations, (b) number of edges analyzed, is $O(n^2)$.

5.2 The specification of the initial partition

The above algorithm is applied to an initial space partition which is then mapped onto a graph. Now a key design issue is to define how to build the initial partition and how to specify sensitive and unreachable cells in such a partition. In other words, given a map of space, what kind of partition can be generated? And how

can the sensitivity level of the initial partition be computed? We have investigated two approaches: a) To subdivide space into a regular grid of cells. Cells have thus equal shapes and sizes. b) To subdivide space into a set of irregular tiles based on a natural subdivision of territory. Each tile represents a real world entity, for example a census block.

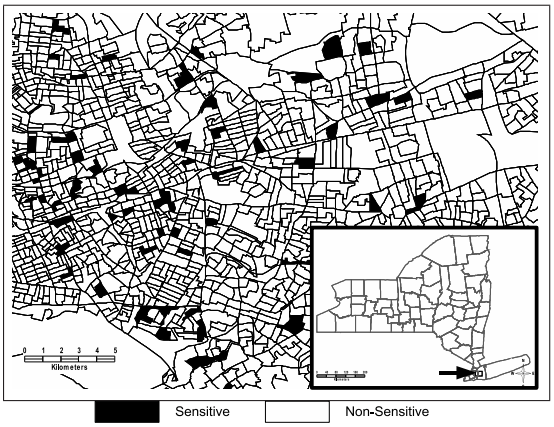


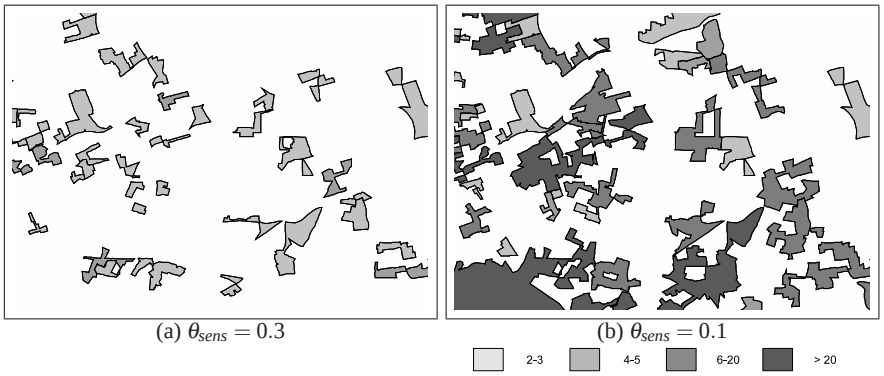
Fig. 2 Sensitive cells in the initial partition

We now discuss the experiments carried out using these two approaches. The adopted software platform consists of a Java implementation of the *SensFlow* algorithm, the system Intergraph Geomedia for the visualization of spatial data and Oracle Spatial for the construction of the RAG.

We present first the experiment with the irregular tessellation of space. Seemingly the advantage of the irregular tessellation against grid is that tiles may represent physical entities. Therefore, since sensitive places, such clinics or religious places, have well-known boundaries, they likely correspond to tiles and thus can be more easily identified. Creating a space tessellation at very high resolution is, however, extremely costly. A more practical solution is to use publicly available datasets, albeit at lower resolution. A typical dataset representing a space partition is the US Census data.

We have thus run the algorithm on an initial partition obtained from US Census Block dataset. The data set consists of 15000 polygons representing Census Block Groups, that is, aggregation of census blocks. Each polygon is a cell of the partition. We assume:

- A unique feature type ft with $score = 1$ thus at the highest sensitivity.
- The density s of sensitive cells is a parameter of the experiment. For example *s=0.05* means that 5% of cells contain sensitive features.
- The percentage of area which is sensitive in a cell is assigned randomly. Figure 2 shows a portion of the initial partition with $s = 0.05$: the black cells are sensitive, whereas the white cells are non-sensitive.



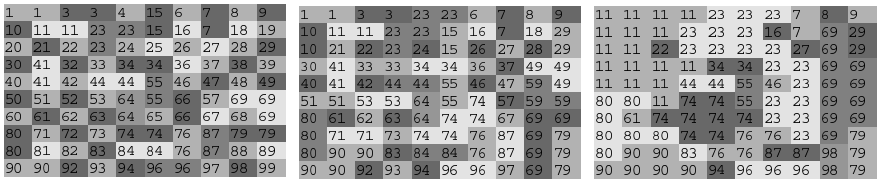
(a) $\theta_{sens} = 0.3$ (b) $\theta_{sens} = 0.1$

Fig. 3 Visual representation of two obfuscated spaces relative to area in Figure 2 ($s = 0.05$)

The *SensFlow* algorithm has been run using different values of the sensitivity threshold. The experimental results are shown in the maps in Figure 3. The generalized regions are represented by polygons of different color, based on the number of aggregations: the color is darker for the more aggregated regions; white space denotes the original space. We can observe that the granularity of the obfuscated space is coarser for lower values of the sensitivity threshold.

The main limitation of this approach is that the publicly available data set is not sufficiently precise. Cells are generally too broad, especially in rural areas and that compromises the quality of service.

We have thus evaluated the grid-based approach to space subdivision. Space is subdivided into a grid of regular cells. Features do not have any physical correspondence with cells. Features are thus contained in a cell or overlap multiple cells. The sensitive area in the cell results from the spatial intersection of the feature extent with the cell. We have run the algorithm over a grid of 100 squared cells, assuming again a unique feature type with maximum score.



$\theta_{sens} = 0.5$ $\theta_{sens} = 0.4$ $\theta_{sens} = 0.3$

Fig. 4 Visual representation of the cell aggregation for different values of the sensitivity threshold θ_{sens}. Merged cells are indicated using both the same number and the same color

Figure 4 shows the obfuscated spaces generated for different values of the sensitivity threshold. The result is visualized as follows: adjacent cells which have not been merged are assigned different gray tones; merged cells have an identical gray

tone and are labeled by the same number. We can observe how the granularity of obfuscated locations (i.e. a set of cells with identical label) changes for different values of θ_{sens}. From the experiments it turns out that the grid-based approach is more flexible because the granularity of partition can be defined based on application needs. On the other hand, the whole process of discretization of features in cells is much more complex.

6 Open issues and conclusions

In this paper we have presented a comprehensive framework for the protection of privacy of sensitive locations. Because of the novelty of the approach a number of important issues are still open, pertaining various aspects concerning: the privacy model, the computational complexity and the system architecture respectively. As concerns the privacy model, one could observe that the sensitivity of a place may vary depending on the context, such as time. Indeed in our approach the user is allowed to specify multiple profiles and thus, ideally, one could select the privacy profile based on the contextual conditions. Unfortunately this solution may result into an excessive burden for the user. Some mechanism for a context-driven selection of privacy profiles would thus be desirable. Another observation is that our privacy model requires detailed knowledge of the extents of sensitive places, while such a knowledge is difficult and costly to acquire. We believe that in the next few years high quality spatial data will become increasingly available under the push of the growing LBS market and thus the development of obfuscation services by LBS providers or third parties will become affordable. Our privacy model can be improved in several ways. First, we observe that the notion of threshold value may be not so intuitive for the user. As a consequence, the specification of the privacy profile may be complex. Second, in our model we assume that mobile users have equal probability of being located in any point outside an unreachable area, while that contrasts with the evidence that some areas are more frequented than others and thus an individual is more likely in those places than in others. The investigation of a probabilistic model is a major effort of the future activity.

A distinct class of issues are about the computational cost of obfuscated map generation. The present algorithm has a quadratic complexity. For an effective deployment of the system, a more efficient algorithm is needed. A related aspect is the development of a suitable platform for the experimental evaluation of the algorithms including a generator of initial partitions. Another major class of issues concerns the specification of a distributed system architecture. We envisage two main architectural solutions. The straightforward approach is to use a *trusted Obfuscation Server* as an intermediary between the client and the LBS provider. The TOS creates the obfuscated spaces and stores them along with the associated privacy profile in a local repository. At run time, the user's request is forwarded to the Obfuscation Server which applies the obfuscation enforcement. This scheme has a main drawback in that it requires a dedicated and trusted server. This may result into a bottleneck;

further the trustworthiness of the server is costly to ensure. To overcome this limitation, an alternative approach is to base the architecture on the following idea. The Obfuscator Server is still used but exclusively to generate obfuscated maps upon user's requests. Once generated, the map is then transferred back to the requesting client which stores it locally. Finally, the obfuscation enforcement is then carried out on the client. Because of the storage limitations of mobile devices, the generated map should be not only generated in an acceptable time for the user but also have a reasonable size.

Acknowledgements This work has been partially funded by the European Commission project IST-6FP-014915 "GeoPKDD: Geographic Privacy-aware Knowledge Discovery and Delivery (GeoPKDD)" (web site: http://www.geopkdd.eu), and by the US National Science Foundation grant 0712846 "IPS: Security Services for Healthcare Applications".

References

1. M. Atallah and K. Frikken. Privacy-preserving location-dependent query processing. In *ACS/IEEE Intl. Conf. on Pervasive Services (ICPS)*, 2004.
2. A. R. Beresford and F. Stajano. Location privacy in pervasive computing. *IEEE Pervasive Computing*, 2(1):46–55, 2003.
3. L. Brun and W. Kropatsch. Contains and inside relationships within combinatorial pyramids. *Pattern Recognition*, 39(4), 2006.
4. W. Du and M. J. Atallah. Secure multi-party computation problems and their applications: a review and open problems. In *NSPW '01: Proceedings of the 2001 workshop on New security paradigms*, pages 13–22, New York, NY, USA, 2001. ACM.
5. M. Duckham and L. Kulik. A formal model of obfuscation and negotiation for location privacy. In *Pervasive Computing*, volume 3468 of *Lecture Notes in Computer Science LNCS*, pages 152–170. Springer Berlin / Heidelberg, 2005.
6. M. Gruteser and D. Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *MobiSys '03: Proceedings of the 1st international conference on Mobile systems, applications and services*, pages 31–42, New York, NY, USA, 2003. ACM Press.
7. In-Stat. http://www.instat.com/press.asp?id=2140&sku=in0703846wt. Publication date: 5 November 2007.
8. P. Kalnis, G. Ghinita, K. Mouratidis, and D. Papadias. Preventing location-based identity inference in anonymous spatial queries. *IEEE Transactions on Knowledge and Data Engineering*, 19(12):1719–1733, 2007.
9. A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. In *22nd IEEE International Conference on Data Engineering*, 2006.
10. M. F. Mokbel, C.-Y. Chow, and W. G. Aref. The new casper: query processing for location services without compromising privacy. In *VLDB'2006: Proceedings of the 32nd international conference on Very large data bases*, pages 763–774. VLDB Endowment, 2006.
11. M. Molenaar. *An Introduction to the Theory of Spatial Object Modelling for GIS*. CRC Press, 1998.
12. Open GIS Consortium. Open GIS simple features specification for SQL, 1999. Revision 1.1.
13. L. Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(5):571–588, 2002.
14. 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, New York, NY, USA, 2006. ACM.