Semantics-aware Obfuscation for Location Privacy

Maria Luisa Damiani and Claudio Silvestri
Università di Milano
{damiani, silvestri}@dico.unimi.it

Elisa Bertino
Purdue University
bertino@cs.purdue.edu

The increasing availability of personal location data pushed by the widespread use of location-sensing technologies raises concerns with respect to the safeguard of location privacy. To address such concerns location privacy-preserving techniques are being investigated. An important area of application for such techniques is represented by Location Based Services (LBS). Many privacy-preserving techniques designed for LBS are based on the idea of forwarding to the LBS provider *obfuscated locations*, namely position information at low spatial resolution, in place of actual users' positions. Obfuscation techniques are generally based on the use of geometric methods. In this paper, we argue that such methods can lead to the disclosure of sensitive location information and thus to privacy leaks. We thus propose a novel method which takes into account the semantic context in which users are located. The original contribution of the paper is the introduction of a comprehensive framework consisting of a semantic-aware obfuscation model, a novel algorithm for the generation of obfuscated spaces for which we report results from an experimental evaluation and a reference architecture.

1. INTRODUCTION

Personal location data are increasingly being collected and processed by a large variety of applications, also pushed by the increased availability of location-sensing systems such as GPS/Galileo, RFID, and sensor networks. Unfortunately, personal location information can lead to the inference of sensitive information about individuals and thus to privacy breaches. For example, the health status of an individual can be inferred from the hospitals and medical labs he or she visits and disclosed to third parties without the consent of the individual. On the other hand, location information

Journal of Computing Science and Engineering, Vol. 2, No. 2, June 2008, Pages 137-160.

is important in a variety of applications, such as data analysis and information services, and therefore it cannot be simply removed from the data being collected. The problem is thus how to protect the privacy of individuals while retaining location information at an adequate level of accuracy so that it can be effectively used in applications.

Personal location data, in its most general form, are captured as tuples of the form (*userId*, *loc*), where *userId* is the value of an attribute identifying an individual, such as the social security number, and *loc* is the location occupied at a given time by this individual. Locations are described in terms of geometric objects, for example a point or a region in a coordinated space, or semantically meaningful spatial objects and descriptions, such as a house, a park, a shop's name or an address. Because a geometric object can be given one or more semantic descriptions and, vice versa a description can be given one or more geometric representations, it is reasonable to assume that the geometric and the semantic representations are interchangeable [Verykios et al. 2007]. Paraphrasing [Beresford and Stajano 2003], we say that a *location privacy threat* occurs when both the identity *userId* and the position *loc* are disclosed.

Location privacy is an important requirement for location-based services (LBS) applications. A LBS is typically requested through a query issued by a mobile client and answered by the LBS provider based on the current position of the client. An example of such a query is: *which is the clinic closest to me?* Position information, once transferred to the LBS provider, is no longer under the control of the user and thus can be arbitrarily used, for example for unsolicited advertisement purposes. To avoid abuses against privacy, the collection of personal location data by third parties is increasingly regulated by law [Bonchi et al. 2007]. Complementary, technological solutions are being investigated to protect personal information against data loss, data theft and abuses by dishonest LBS providers.

A naïve technique to protect location privacy is to remove the user's identifier from service requests. Unfortunately, the simple removal of the identifier may be not sufficient to anonymize requests because identity may be inferred from the association of the user's location with some external data source. For example, if the position of an individual is known to be inside a non-publicly accessible office, it is likely that such individual works at that office and thus can be easily identified. Moreover, applications may need to have available identity information for accounting and user authentication. Various techniques have been recently developed which attempt to protect either the identity or the actual location of the user.

The basic idea underlying most schemes is to hide the actual position by forwarding to the LBS provider a coarse location, that is, a location with low spatial resolution. The operation of deliberately degrading the quality of the information about an individual's location is also called *location obfuscation*. We can abstractly think of location obfuscation as a function which takes in input the actual position of an individual and returns a location which is either *imprecise* (i.e. the information lacks specificity) or *inaccurate* (i.e. it lacks correspondence between information and reality) [Worboys and Clementini 2001].

Location obfuscation techniques typically use geometric methods such as geometric

transformations and space subdivision to generate coarse locations. We refer to these techniques as *geometry-based*. Note that geometry-based techniques do not consider "what" is contained in space, that is, the spatial entities that populate the world, the places and their relationships. In other words those techniques do not account for *geographical knowledge*. We believe that the lack of concern for the semantics of space may lead to privacy breaches. In particular, we claim that an adversary with sufficient geographical knowledge may infer sensitive location information from obfuscated locations generated by geometric-based techniques. To support our claim we present the example below.

1.1 A Motivating Example

Assume that *John* is subscriber of an LBS application. John does not want to let anyone know that he has health problems. Therefore he does not want to reveal his position while being in a hospital. Conversely John is not concerned with location privacy when in other places, like a park.

Consider now the case of a hospital close to a lake and to a residential district. Suppose that each of these places, i.e. the lake, the district and the hospital, covers a polygonal region. Suppose moreover that no boats are allowed on the lake. Now assume that John requests a service from a certain position p and that an obfuscated location o is computed and transmitted to the LBS provider in place of p. Now suppose that an adversary has the same knowledge we have about the geographical context. From the observation of the spatial relationships existing between the obfuscated location and real world entities, like the topological relationship of spatial containment, overlapping and disjointness, the adversary may likely determine whether *John* is located in a hospital and thus in a sensitive place.

In particular consider the following three cases:

(a) The obfuscated location o is spatially contained in the extent of the hospital. In this case, *John* still results to be located in a sensitive place although the actual position is blurred to a coarser region.

(b) o includes the extent of both the hospital and the lake. Since *John* 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, that is, that no boat is allowed on the lake, in order to infer more precise information about the actual location of the user.

(c) o overlaps part of the hospital and part of the residential district. Since only the hospital is a sensitive place for *John*, we can say that the obfuscated location is "sensitive to some extent".

This 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 semantics of space is what ultimately determines the sensitivity of the location. Furthermore, it is important to note that different locations may have different degrees of sensitivity, depending also on the user's perception of what is sensitive.

From this observation it follows a key requirement, that is, the techniques for the generation of obfuscated locations must take into account the qualitative context in which users are located and move. To address such requirement, in this paper we propose a novel obfuscation framework based on the use of geographical knowledge, referred to as *obfuscation system*. In this paper we present the core components of the obfuscation system:

1. a semantics-aware *obfuscated space* model;
2. an algorithm, called *SensFlow*, for the generation of obfuscated spaces for which we report an experimental evaluation.

To our knowledge, this is the first approach addressing the problem of how to prevent the disclosure of sensitive locations in an LBS context.

1.2 Paper Organization

The remainder of the paper is structured as follows. Next section overviews related work. Then we introduce the baseline of the proposed approach and the architectural framework of the obfuscation system. The obfuscation model is presented in the subsequent section followed by the obfuscation method and the *SensFlow* algorithm along with the experimental evaluation. A final section, reporting future research directions, concludes the paper.

2. RELATED WORK

In this section we overview related work concerning location privacy and recent advances in data privacy for relational databases. Early proposals focus on the specification of *privacy policies*. Privacy policies state who and when a subject is allowed to access location information concerning a specific individual.

The drawback of privacy-policy based approaches is that they provide a strong protection only when LBS providers are trustworthy. Trustworthiness however is costly to ensure. Therefore it is more realistic to assume that personal location data are subject to privacy attacks. Note that we use the notion of *privacy attack* informally to mean that an adversary can detect with certainty the actual location of the user. A first attempt to formalize the concept of privacy attack is by Bettini et al. [Bettini et al. 2007]. Most recent work comprises two main categories of privacy models, focusing respectively on the protection of location information when the user cannot be anonymous, and the concept of *k-anonymity*.

2.1 Privacy Models for the Protection of Location Information

Following Atallah and Frikken [Atallah and Frikken 2004], the problem can be succintly formulated as follows: given a query such as "give me the address of the post-office that is nearest to my current position", how can the query be processed by the database without knowing the actual position of the requester? Atallah and Frikken propose three methods of varying complexity to process nearest-neighbor queries that we illustrate in detail. The first approach is straightforward; the client applies a geometric translation to the user's precise position and forwards the approximated position to the LBS provider. The database answers the query and

returns an approximated answer. The second method does not result in any accuracy loss but has potentially higher communication costs. The idea is to subdivide the space in a grid of cells. The client queries the database with the tile, thus with an obfuscated location, 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 basic idea is that the set of spatial objects in the database is processed so as to produce a Voronoi diagram, that is, a subdivision of the plan into cells such that each cell is associated with one of the objects of concern. Then a *secure-multiparty protocol* [Du and Atallah 2001] is used to assess whether the user's position is contained in a cell without revealing to the database anything other than the Yes/No answer to the question. If the answer is Yes, the object associated with the cell is the one closest to the user. Note that using a secure-multiparty protocol is much like using a trusted third-party getting the actual position of the user and the Voronoi diagram and then matching such position against the cells. Finally note that this mechanism can be only applied to spaces which are partitioned.

Another approach for the processing of nearest-neighbor queries is proposed by Duckham and Kulik [Duckham and Kulik 2005]. In this case the geographic space is represented in the database as a planar graph. The vertexes of the graph denote locations of objects and users while the edges represent how such locations are connected by roads. Edges have associated a weight representing the distance between two adjacent vertexes. 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 (vertexes) that are closest to any point in P. Then, in the simplest case, the database returns the whole set of vertexes leaving the client to choose among them.

Cheng et al. [2006] focus on a different class of spatial queries. The problem is formulated as follows: given a space populated by n users $\{u_1, ...,u_n\}$ located in uncertain (i.e. obfuscated) positions and represented as closed regions, determine which of them are located within distance C from a given user u_i. Such queries are called *imprecise location-based range queries*. Basically the idea is to process the query for each point of the location of u. Hence, unlike the approach by Atallah and Frikken, each possible answer is assigned a probability. We observe that the emphasis of such an approach is on query processing and not on obfuscation techniques.

2.2 Protection of User Identity through *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 is a set of one or more attributes which, though not containing an explicit reference to the individual identity, can be easily linked to external data

sources and in this way reveal who the individual is. k-anonymity can be achieved by generalizing the quasi-identifier values. Generalization is thus performed by replacing a value with a less specific but semantically consistent value [Sweeney 2002]. The value of k quantifies the degree of privacy in T.

In the LBS context, the k-anonymity concepts are translated as follows: like a table record, a *request* is a tuple of values including the user's location and the location based query. The location attribute is treated as a quasi-identifier. A request is thus k-anonymous if the user's location is indistinguishable form the location of other k-1 individuals. Finally a *generalized location* is a region containing the position of k individuals. Note that a generalized location is nothing but an obfuscated location.

Typically k-anonymity techniques are applied using a three-tier architecture consisting, beyond a client and a server, of a trusted intermediary, referred to as *anonymizer*. The anonymizer receives the user's service request, removes the user's identity, acquires and generalizes the location of the user and then forwards it to the LBS. The LBS in turn processes the request and returns the anonymizer a set of candidates containing the actual results.

Location generalization techniques generate obfuscated locations independent of the query type. The first such technique has been proposed by Gruteser and Grunwald [Gruteser and Grunwald 2003]. Their approach is based on a recursive subdivision of the space in quadrants. The set of quadrants is represented by a *quadtree* data structure. 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 enough users to satisfy the anonymity constraint k. Such a final quadrant constitutes the generalized location. Another technique based on quadtrees has been proposed in the Casper system [Mokbel et al. 2006]. The anonymizer uses a hash table 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 the user *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.

Kalnis et al. [2007] observe that 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 be univocally associated with a user. To prevent such privacy leaks, Kalnis et al. propose the Hilbert Cloak algorithm. The main idea is to sort users' locations by utilizing a classical spatial ordering technique, the Hilbert space-filling curve technique, mapping two-dimensional coordinates of each user onto a one-dimensional value. Users' position, which are contiguous in the ordering and thus, for how the Hilbert space-filling curve is defined, in space, are grouped in *buckets* of k users. Therefore each user belongs to a unique k-bucket and all users in the same bucket have the same generalized location.

2.3 Beyond *k*-anonymity

Finally we discuss related work concerning advances in data privacy for relational databases. Recently several papers have pointed out that *k*-anonymity has a number of drawbacks and thus it 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*. As an example consider a table T consisting of the three following attributes: *Zip Code* and *Age* represent a quasi-identifier, *Disease* is a sensitive attribute. Assume a publicly-available, *k*-anonymous dataset T composed of groups of at least k records and assume that all the tuples in a certain group have an identical value for the sensitive attribute, say ([13000–13099], [40–50], *cancer*). Then, if John is known to be present in the table, live at zip code 13057 and be 44 years old, one can immediately infer that John has the cancer, albeit the table is *k*-anonymous. *l-diversity* is a possible counter-measure against such an attack. The main idea of *l*-diversity is the requirement that the values of the sensitive attributes must be *well represented* in each group [Machanavajjhala et al. 2006]. 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. For example, consider the previous table T, and assume it to be 2-anonymous. Consider a group consisting of the following tuples: t_1 = ([13000–13099], [40–50], *bronchitis*), t_2 = ([13000–13099], [40–50], *pneumonia*). If John is known to belong to such a group, then it is easy to infer that John must have suffered from bronchitis or pneumonia, which is acceptable according to *k*-anonymity and also *l*-diversity. However, John may want not to disclose that he has had respiratory problems. On the other hand, an individual suffering from *flu* may be unconcerned with the disclosure of the sensitive attribute value. To address this requirement, Xian and Tao [2006] introduce the concept of *personalized anonymity*. The main idea is to organize the values of sensitive attributes 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. Unfortunately this technique can be only applied to categorical attributes.

3. SEMANTICS-AWARE OBFUSCATION

We now outline the general approach we propose for semantic-aware obfuscation. The main idea is as follows: users specify which places they consider sensitive and the desired degree of privacy in a *privacy profile*. Based on the privacy profile, the privacy-preserving system generates a set of obfuscated locations and associates it with such a profile. At run time, a user issues an LBS request specifying also the profile and thus implicitly the set of obfuscated locations. If the user's position falls inside any location o from such a set, then o is forwarded to the LBS provider in place of the actual position. An adversary cannot thus infer with certainty that the

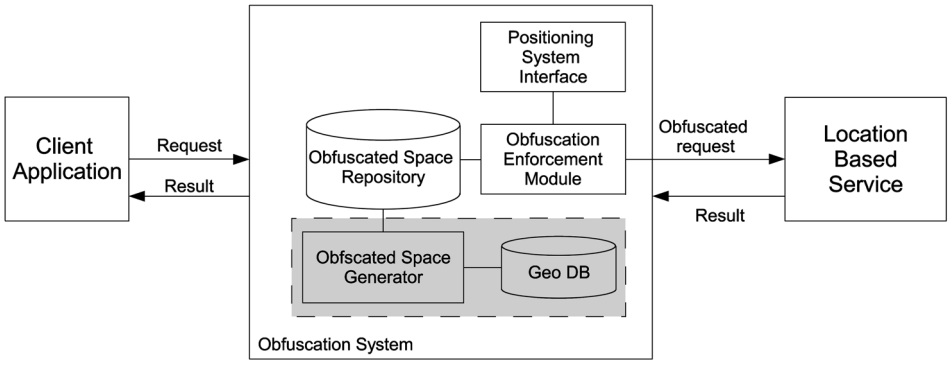


Figure 1. Architecture of the system.

user is inside a sensitive (for the user) place. At most the adversary can infer that the position *may be* in a sensitive place. To implement this strategy, we propose a privacy-reserving framework based on a number of key concepts.

- *Sensitive places.* We assume that relevant places are classified in categories or types of spatial objects. Each user can specify 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 that area; a place is unreachable when nobody can be located in that place; conversely a place is non-sensitive.
- *Level of sensitivity.* The level of sensitivity quantifies the level of location privacy leak the user is exposed to when located in a region. For example a region entirely occupied by a hospital has a high level of sensitivity. 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.
- *Obfuscated space.* It is a partition of the reference space consisting of *obfuscated locations*. Specifically, obfuscated locations are regions which have a level of sensitivity less than or equal to than a *sensitivity threshold value*. The sensitivity threshold value quantifies how much sensitive the resulting obfuscated location can be for the user. The sensitivity threshold value and the set of sensitive and unreachable types of places is specified in the *privacy profile* of the obfuscated space. The obfuscated space is generated by an *obfuscation algorithm*.
- *Obfuscation enforcement.* It refers to the mapping of the user's position onto a region of an obfuscation space. Given an obfuscated space *OS*, the location which obfuscates user's position p is the region of *OS* which contains p.

Figure 1 illustrates the functional components of the obfuscation system. The first component is the *obfuscated spaces repository*. Such repository stores the obfuscated spaces which are generated based on the privacy preferences of users. Because obfuscated spaces only need to be generated once and then used every time a position is obfuscated, it seems reasonable to assume that the obfuscated space is created *off-line* and not while processing the request. Each obfuscated space in the repository has a *privacy profile*. Such profile specifies user preferences, in particular which values are assigned as input to the parameters of the obfuscation algorithm.

The obfuscation algorithm is run by the obfuscated space *Generator*. The *GeoDB* is the spatial database, which contains the spatial entities of concern to the application. Both the Generator and the GeoDB must be trusted.

The user interaction is as follows. The user starts a session by specifying the obfuscation space to consider in the session. When the user requests an LBS, the query is sent to the obfuscation system. The *obfuscation enforcement module* then retrieves the actual user's position through the *positioning system interface*. Since the details of how the position is obtained are not relevant for our work, we assume that the user's position is known with a high level of accuracy and precision and is provided in the form of georeferenced coordinates x and y and that the user positions are communicated securely. Then the obfuscation enforcement module matches the actual position against the obfuscation space associated with the session; finally the resulting obfuscated location along with the query is forwarded to the LBS provider. As it is the operational core of the system, the obfuscation enforcement module must be trusted.

As a result, the LBS provider processes the query based on the imprecise user's position. There are diverse approaches in literature for the computation of spatial queries based on imprecise positions, such as [Cheng et al. 2006; Mokbel et al. 2006; Kalnis et al. 2007]. It is important to observe, however, that the investigation of those techniques is beyond the scope of this paper, because our goal is to define a location generalization method which takes into account the semantics of places for the purpose of protecting against location inferences based on background knowledge. Such a generalization is independent from the issued queries.

4. THE OBFUSCATION MODEL

We now present the basic concepts of the obfuscation model. We first introduce some assumptions on space. Then we introduce the important notions of sensitivity level and obfuscated spaces, which are the key notions of our approach.

Space S is a coordinate space of 1, 2 or higher dimension. The *position* of a user is a point in S. Throughout the paper, we use the term *location* to broadly denote a portion of space containing the user's position. A *region* in S is a bounded, connected subset of space of the same dimension of S. Without loss of generality, in the rest of the paper we consider 2D-spaces; therefore a region is represented by a polygon.

Space is populated by spatial objects representing real world entities such as buildings and roads. Following a popular data model, spatial objects are modeled in terms of OGC *simple features*. A *feature* has an unique name, say *Milano*, and a unique *feature type*, say *City*. Furthermore, a feature has attributes and a geometric extent of type point, line, polygon or collection of disjoint geometries [Open GeoSpatial 2005]. An advantage of our approach is that spatial objects can be stored in commercial spatial DBMSs and easily displayed as maps. Furthermore, it supports fine-grained and application-dependent privacy requirements. Our approach is also extensible in that new feature types can be easily created from the initial spatial database. In what follows, without significant loss of generality, we assume that the features of interest are of polygonal type. 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.

Users specify their preferences about the feature types that are to be considered *sensitive* and *unreachable* for the generation of the obfuscated space. We say that 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. We say instead that a features type is *non-reachable* when it denotes a set of places where, 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 civilian.

The user specifies sensitive and unreachable features in the *privacy profile*. As we will see, the user can add further parameters in this profile. The privacy profile is then used by the obfuscation system for the generation of the obfuscated space. Ideally a user can define multiple privacy profiles and thus be associated with multiple obfuscation spaces.

4.1 Sensitivity Level

Assume that a user has chosen a set of sensitive (unreachable) features types. The next step is to define a metric to quantify how much sensitive an arbitrary location is with respect to the user's privacy requirements. Such a metric is called *sensitivity level* (SL). The value of SL depends on: a) the classification of features types. For example if only few features are sensitive, it is likely that the sensitivity level in the region is low. b) The nature of the location. For example a location enclosing a large hospital is likely more sensitive than a region enclosing a small clinic. c) The perception of users. For example, typically a hospital is not a sensitive feature for a doctor while it may be so for an individual with health problems. To model SL we introduce the notions of *sensitive location probability* and *sensitivity score*.

- *Sensitive location probability*. The sensitive location probability P_{sens} with respect to a region r is the probability that the user, if known to be in r, is actually located in the extent of any sensitive feature in r. We assume that a user has equal likelihood of being located in whatever point in space. The value P_{sens} is computed by the system based on the extent of the sensitive feature types in the region.
- *Sensitivity score*. The sensitivity score of a feature type defines "how much private" the information of being in a region of that type 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 the privacy of medical information than with information about his/her preferred restaurants. The score is assigned a quantitative value ranging between 0 and 1: 0 means that the feature type is not sensitive, while 1 means that the feature type has the highest sensitivity. The score of each sensitive feature type is specified directly by the user in the privacy profile.

4.2 The Computation of the Sensitivity Level

Based on the previous concepts, we now specify how to compute the value SL for a

region, given a privacy profile.

We use the following notation: E is the set of regions in the reference space; the pair (FT, F) is the geographical database, namely the set of feature types and features; $FT_{Sens} \subseteq FT$ is the set of sensitive feature types, while $FT_{Nreach} \subseteq FT$ is the set of non-reachable features, with $FT_{Nreach} \cap FT_{Sens} = \emptyset$. Given a region r, 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. $Score(ft)$ is the score assigned to feature type *ft*.

We now introduce $Area_{Rel}(r)$, the function computing the relevant area of r, that is, the portion of region not covered by unreachable features:

$$Area_{Rel}(r) = Area_{Geo}(r) - \sum_{ft \in FT_{Nreach}} Area_{Fea}(r, ft)$$

Next we define the function $P_{Sens}(r)$ computing the *sensitive location probability* of r as the ratio of the area covered by sensitive features to the relevant area of the region:

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

If r only consists of unreachable features we define $P_{sens}(r) = 0$.

Finally, we define the sensitivity level of the region $SL_{Reg}(r)$. $SL_{Reg}(r)$ is computed as sum of the ratios of weighted sensitive areas to the relevant area in the region.

Definition 4.1 (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:

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

If r *only contains non-reachable features, we define* $SL_{reg}(r) = 0$.

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 II reports for each feature type* ft_i *and region* c_j *the area occupied by* ft_i *in* c_j, *with* i *ranging over* $\{0, 1, 2, 3, 4\}$, *and* j *over* $\{0, 1, 2, 3\}$. *Note that two regions include a portion of area, occupied by features of type* ft_4,

Figure II. Area and sensitivity scores of feature types.

$Area(c, ft)$	c_0	c_1	c_2	c_3
ft_0	200	0	100	0
ft_1	100	0	0	50
ft_2	300	450	200	0
ft_3	0	200	100	400
ft_4	100	0	0	50
$Area_{rel}$	400	200	200	500
SL_{reg}	0.425	0.900	0.700	0.790

$Score(ft)$
0.5
0.7
0
0.9
0
-
-

which is not sensitive. *The row labeled as* $Area_{rel}$ *reports the size of the reachable areas of each region. Finally, the column labeled* $Score(ft)$ *reports the sensitivity score assigned to each feature type.*

The sensitivity level for regions c_0 *and* $c1$ *is:*

- $SL_{reg}(c_0) = \frac{0.5 \cdot 200 + 0.7 \cdot 100 + 0.9 \cdot 0}{400} = 0.425$
- $SL_{reg}(c_1) = \frac{0.9 \cdot 200}{200} = 0.9.$

According to our metrics 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.*

4.3 Obfuscated Space

Before introducing the notion of obfuscated space, we need to extend the notion of sensitivity level from regions to space partitions. A *space partition* consists of a set of disjoint *cells* covering the reference space S. The idea is to define a metric for comparing the sensitivity not only of regions but also of space partitions. The sensitivity level of a partition is defined as the maximum value of the sensitivity levels of the partition cells. Denoted with $\Pi(S)$ the set of partitions over space S, we formally define the sensitivity of a partition as follows:

Definition 4.2 (Sensitivity level of partitions) *Let* $\mathcal{C} \in \Pi(S)$ *be a partition over* S. *The sensitivity level of a partition is defined by the function:* $SL_{Par}: \Pi(S) \rightarrow [0, 1]$ *such that:*

$$SL_{Par}(\mathcal{C}) = \max_{c \in \mathcal{C}} SL_{Reg}(c)$$

Now we define the obfuscated space as a partition with a sensitivity level less than or equal to a *threshold value*. Each cell of the partition represents an obfuscated location of each position inside the cell. The threshold value expresses the degree of location privacy requested by the user and is specified directly in the privacy profile. Therefore given an obfuscated space OS, for each point p in S, exactly one cell containing p exists in OS which has a level of sensitivity less than or equal to the specified threshold. The formal definition of obfuscated space and privacy profile is reported below.

Definition 4.3 (Obfuscated space) *Let* (FT, F) *be the geographical database. Moreover let:*

- *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 in* $\Pi(S)$ *such that:* $SL_{Par}(OS) \leq \theta_{sens}$

(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_4 *represents public gardens and is not a sensitive feature type*
- $FT_{Nreach} = \{ft_2\}$ *where* ft_2 *represents a military zone*
- $Score(ft_0) = 0.5$, $Score(ft_1) = 0.7$, $Score(ft_2) = 0$, $Score(ft_3) = 0.9$, $Score(ft_4) = 0$
- $\theta_{\text{sens}} = 0.9$

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

5. COMPUTATION OF THE OBFUSCATED SPACE

After having presented the privacy model, we discuss how to generate an obfuscated space. The basic idea is to segment the space in locations starting from an initial fine-grained discretization of space and repeatedly merging adjacent regions until a termination condition is met. Specifically, this strategy is articulated in three main steps.

1. **Specification of the initial partition**. The space is subdivided in a set of small cells which constitute the *initial partition* denoted as $\mathcal{C}_{in} \in \Pi(S)$. The initial partition can result, for example, from a regular tessellation of space, or from an application-driven subdivision. The granularity of the initial partition, that is, how small the cells are, is an application-dependent issue which is not addressed here.

2. **Setting of input parameters**. Input parameters are specified in the privacy profile $< FT_{Sens}, FT_{Nreac}, Score, \theta_{sens} >$.

3. **Iterative method**. The *current partition* is checked to assess whether it is an obfuscated space for the given profile. If this is the case, the process terminates. Otherwise each cell which has a level of sensitivity higher than θ_{sens} is merged with an adjacent 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.

We now detail the iterative method. Given two adjacent cells $c_1, c_2 \in \mathcal{C}$, the operation which merges the two cells generates a new partition $\mathcal{C}'$ in which cells c_1 and c_2 are replaced by cell $c = c_1 \cup c_2$. We say that partition $\mathcal{C}'$ *is derived* from partition $\mathcal{C}$, written as $\mathcal{C}' \succcurlyeq \mathcal{C}$. Consider the set P_{Cin} of partitions derived directly or indirectly from the initial partition C_{in} through subsequent operations of merge. It can be easily shown that the poset $H = (P_{Cin}, \succcurlyeq)$ constitutes 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 claim that an obfuscated space can be generated by progressively aggregating cells in coarser locations. To prove our claim we show that the level of sensitivity of a partition remains identical or decreases when cells are aggregated in coarser

regions. Therefore, starting from the initial partition which has the highest level of sensitivity, one can proceed by merging adjacent cells until either the obfuscated space is found or no aggregation is possible. The demonstration is articulated as follows: a) Lemma 5.1 shows that the *SL* (i.e. sensitivity level) of the cell resulting from a merge operation is less or equal than the sensitivity level of the starting cells. b) Theorem 5.2 shows that the sensitivity level of the partition resulting from subsequent merge operations is less or equal the sensitivity level of the starting partition. c) Corollary 5.3 specifies necessary and sufficient condition for at least one obfuscated space to exist. Proofs are reported in appendix.

Lemma 5.1 *Let* c_1, $c_2 \in \mathcal{C}$ *be two cells and* $c = c_1 \cup c_2$ *the cell resulting from the merge operation. The following inequality holds:*

$$SL_{reg}(c) \leq \max\,(SL_{reg}(c_1),\, SL_{reg}(c_2)).$$

In essence, the Lemma states that the region resulting from the merging of two cells is "equally or less sensitive" than the initial cells. The following theorem extends this result to a sequence of merge operations.

Theorem 5.2 *Consider the two partitions* $\mathcal{C}_A$ *and* $\mathcal{C}_B$*. The following implication holds:*

$$\mathcal{C}_A \succcurlyeq \mathcal{C}_B \Longrightarrow SL_{par}(\mathcal{C}_A) \leq SL_{par}(\mathcal{C}_B)$$

We say that the sensitivity level of a partition is weakly anti-monotonic with respect to the "be derived" relation.

Theorem 5.2 states that a sequence of merge operations does not increase the sensitivity level *SL* of the initial partition. As a consequence, the maximal partition, consisting of a unique region, has the lowest attainable sensitivity level among the partitions of the set P_{Cin}. Therefore, if such a value is higher than the sensitivity threshold, the problem has no solution. On the other hand, if such a constraint is satisfied, the maximal partition results to be one of the possible solutions. Formally:

Corollary 5.3 *Consider the maximal partition, that is, the partition consisting of a unique region, denoted as Max. The following hold:*

(1) $SL_{par}(Max) \leq \theta_{sens}$ *implies that at least one obfuscated space can be generated.*

(2) If $\mathcal{C}$ *is an obfuscated space, then*

$$SL_{par}(Max) \leq SL_{par}(\mathcal{C}) \leq SL_{par}(\mathcal{C}_{in})$$

(3) $SL_{par}(Max) > \theta_{sens}$ *implies that no obfuscated space can be generated.*

6. THE *SensFlow* ALGORITHM

Multiple obfuscated spaces can be generated for the same profile. We consider *optimal* the obfuscated space with the maximum cardinality, thus possibly consisting of the finest-grained regions. The problem we address can be formulated as follows:

Given an initial partition $\mathcal{C}_{in}$ *and a profile*

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

determine, if it exists, the sequence of merge operations such that the resulting partition $\mathcal{C}$ is an obfuscated space with the maximum number of cells, that is:

(1) $SL_{par}(\mathcal{C}) \leq \theta_{sens}$
(2) $\mathcal{C} \succcurlyeq \mathcal{C}_{in}$
(3) $|\mathcal{C}| \geq |\mathcal{C}'|$ *where* $\mathcal{C}'$ *is an obfuscated space with the same profile.*

Since we are not aware of any efficient algorithm computing the optimal solution, we present an algorithm which computes an approximated solution for this problem. The idea is to progressively expand each cell which is over-sensitive (that is, for which the level of sensitivity exceeds the threshold value) until a terminating condition is met. This approach raises a number of issues. The first issue is how to choose the cell to be merged. A reasonable approach is to select the adjacent cell which most reduces the sensitivity level of partition. A second issue concerns the criteria for the expansion of cells. To address such issue, we have identified two basic strategies: the first strategy is to expand one over-sensitive cell at a time, until the sensitivity level 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 achieves a better control over the size of each location. In what follows we outline the algorithm.

6.1 Implementation

We represent a space partition through a *Region Adjacency Graph* (RAG) [Molenaar 1998]. In general a RAG is defined from a partition by associating one vertex to each region and by creating an edge between two vertices if the associated regions share a common boundary. In addition we label each vertex v of the RAG with a tuple $< a_1, a_2, ..., a_n >$ specifying for each feature type $ft_i \in FT$, with $i \in [1, n]$ and $n = |FT|$, the size a_i of the area covered by features of type ft_i in v. Within this framework, the edge information is interpreted as a possibility to merge the two regions identified by the vertices incident to the edge. Such a merge operation collapses the two vertices incident to the edge into one vertex and removes this edge together with any double edge between the newly created vertex and the remaining vertices [Brun and Kropatsch 2006].

Starting from the RAG corresponding to the initial partition, the algorithm shrinks the graph by merging adjacent cells until the sensitivity level of the partition is less or equal the threshold value or the RAG is degenerated into a unique vertex. At each step, the iterative algorithm selects the pairs of cells to merge. In case a cell c can be merged with different cells, the algorithm analyzes each possible merge operation and selects the one which minimizes the sensitivity level of the resulting region in case such a value is less than or equal to the SL of c. Note that two cells can be merged although the resulting region has the same sensitivity to enable the combination of cells which are separated by unreachable cells (e.g. an island separated from the mainland).

Algorithm 1 *SensFlow* Algorithm

```
1: procedure ObfuscatedSpace(G, θ_sens)
2:     repeat                                                    ▷ loop 1
3:         modified ⟵ false
4:         for v_1 ∈ V(G) do                                     ▷ loop 2
5:             if v_1.SL > θ_sens then
6:                 best_sens ⟵ v_1.SL
7:                 best_vertex ⟵ nil
8:                 for each v_2 ∈ v_1.adjacent do                ▷ loop 3
9:                     if SL(v_1, v_2) ≤ best_sens then
10:                        best_sens ⟵ SL(v_1, v_2)
11:                        best_vertex ⟵ v_2
12:                    end if
13:                end for
14:            end if
15:            if best_vertex <> nil then
16:                G.merge(v_1, best_vertex)
17:                modified ⟵ true
18:            end if
19:        end for
20:    until ¬modified
21: end procedure
```

The pseudo-code of the algorithm is reported in Algorithm 1. The input parameters are: 1) the initial RAG built on the initial partition (parameter G); 2) the sensitivity threshold (parameter θ_{sens}). The algorithm returns an obfuscated space if it exists, otherwise the maximal partition consisting of a unique vertex. The internal representation of the RAG is based on adjacency lists. The algorithm repeatedly examines the over-sensitive cells; each over-sensitive cell is then merged with at most one cell at each iteration following a breadth-first strategy.

Specifically the input graph is processed as follows: Loop 2 repeatedly scans the list of vertices $V(G)$, to find those which have a level of sensitivity higher than the input threshold value. Once a vertex is found, say v, the algorithm analyzes the adjacency list of v to find a vertex to merge with v (Loop 3). If the sensitivity of the vertex resulting from the merge, e.g. $SL(v_1, v_2)$ is less than the sensitivity of the current vertex, the algorithm proceeds with the merge operation. In case more than one candidate exists, the vertex which determines the minimum level of sensitivity is selected. Loop 2 then proceeds to scan the remaining vertices. The whole loop is repeated until no vertex is modified. Termination occurs when either every vertex has a level of sensitivity less or equal the threshold value or it is not possible to further improve the sensitivity of vertices.

We measure the complexity of this algorithm based on the number of two key operations: (a) the merge of two vertices, (b) the number of edges analyzed. Let n be the number of vertices, and m the number of edges. We observe that no more than n merge operations can be performed before a single vertex is found. Thus, loop 1 is executed at most n times and, since at each iteration we perform at least one merge, the total number of merge operations (a) is $O(n)$. At each step we consider all the

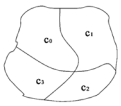
edges incident to over-sensitive vertices, each corresponding to a potential merge ($O(m)$ edges analyzed, in the worst case). Thus, the total number of edges considered before termination is $O(m*n)$. Since we are considering a planar graph, the number of edges is proportional to the number of vertices. Therefore the number of edges analyzed (b) is $O(n^2)$.

6.2 Example

The major steps of the algorithm are illustrated in Figure 2. We consider an initial partition and assume sensitivity threshold to be set to $\theta_{sens} = 0.75$. Each sub-figure in Figure 2 represents the current partition at each step of the algorithm; the tables report the area per cell and feature type. The rightmost column reports the score of feature types and the last row the SL of cells.

We observe that in Figure 2.A the sensitivity level of cell c_1 and c_3 is higher than the threshold $\theta_{sens} = 0.75$. The algorithm thus evaluates which merge operation would be the most convenient; merging cells c_1 and c_0 would result in a new cell with sensitivity level 0.583, whereas merging cells c_1 and c_2 would yield sensitivity level

A) Initial partition

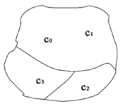


$SL(c_1) > \theta_{sens}$

$Area(c, ft)$	c_0	c_1	c_2	c_3	$score(ft)$
ft_0	200	0	100	0	0.5
ft_1	100	0	0	50	0.7
ft_2	300	450	200	0	0
ft_3	0	200	100	400	0.9
ft_4	100	0	0	50	0
$SL(c)$	0.425	0.900	0.700	0.790	

$SL(c_0 + c_1) = 0.583$
$SL(c_1 + c_2) = 0.800$

B) Merge c_0 and c_1

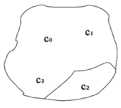


$SL(c_3) > \theta_{sens}$

$Area(c, ft)$	$c_0 + c_1$	c_2	c_3	$score(ft)$
ft_0	200	100	0	0.5
ft_1	100	0	50	0.7
ft_2	750	200	0	0
ft_3	200	100	400	0.9
ft_4	100	0	50	0
$SL(c)$	0.583	0.700	0.790	

$SL(c_2 + c_3) = 0.764$
$SL(c_0 + c_1 + c_3) = 0.677$

C) Merge $c_0 + c_1$ and c_3



$Area(c, ft)$	$c_0 + c_1 + c_3$	c_2	$score(ft)$
ft_0	200	100	0.5
ft_1	150	0	0.7
ft_2	750	200	0
ft_3	600	100	0.9
ft_4	150	0	0
$SL(c)$	0.677	0.700	

Figure 2. Evolution of partitions ($\theta_{sens} = 0.75$).

0.800. Thus, the algorithm replaces cells c_1 and c_0 with a new cell labeled as $c_0 + c_1$.

Figure 2.B shows the result of the merge operation. It can be noticed that the sensitivity level of cell c_3 is still higher than the threshold. Since c_3 is adjacent to $c_0 + c_1$, and c_2, the levels of sensitivity associated with the resulting regions are, respectively, 0.677 and 0.764. The algorithm thus collapses $c_0 + c_1$ and c_3 into a new cell denoted as $c_0 + c_1 + c_3$. At this point the algorithm stops, because the sensitivity level of every cell in the resulting partition (Figure 2.C) has a sensitivity level which is below the threshold 0.75. The obfuscated space is thus found.

7. EXPERIMENTAL EVALUATION

We have carried out both a qualitative and a quantitative analysis of the approach. The qualitative analysis is to analyze the shape and the extent of obfuscated regions for different values of the sensitivity threshold θ_{sens}. The quantitative analysis reports statistics about the size of the generalized regions and the computational cost of the proposed algorithm for different values of θ_{sens}.

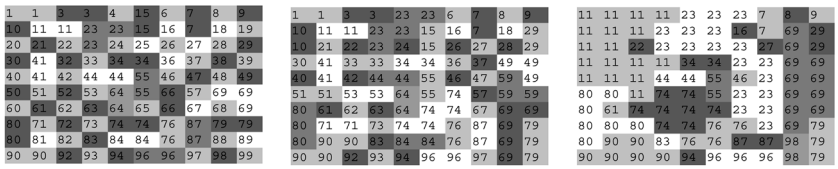
We have run the experiments over a space populated by sensitive areas generated on a random basis. The experimental setting and the results of the analysis are described in the following.

7.1 Experimental Setting

We assume an initial partition consisting of a regular grid of 100 squared cells. The neighbors of cells that we consider are exclusively the four cells on the north, south, east and west side; therefore each vertex of the RAG has degree four. Each cells may include sensitive features. If so the cell is termed sensitive. We consider a unique sensitive feature type with score 1. The extent of the sensitive area in a cell has a value between 1 and 10 randomly assigned. Similarly, each cell is assigned a value in the same range for the area which is relevant but non sensitive. The density of sensitive cells in a grid is a parameter of the experiment denoted as p; p=0.8 means that 80% of cells contains sensitive features and those cells are uniformly distributed. For each value of p, ranging from 0.9 to 0.01 (the figures report only 4 values), we have generated 100 grids, each with a different spatial distribution of sensitive features.

7.2 Qualitative Evaluation

Figure 3 shows the obfuscated spaces generated for different values of the sensitivity threshold. The result is visualized as follows. Cells are assigned a color and a number.



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

Figure 3. Visual representation of the obfuscated regions for different values of θ_{sens}.

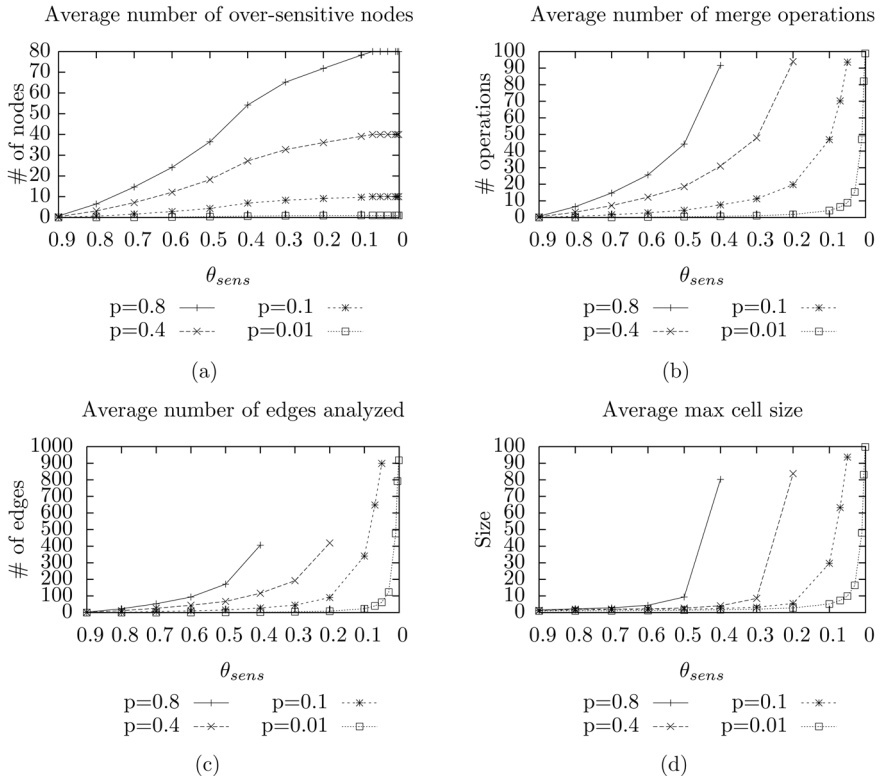


Figure 4. (a) Average number of cells having a sensitive value above the threshold in the original partition. (b-c) Cost of the algorithm: average number of merge operations and analyzed edges. (d) Average maximal size of obfuscated regions.

The adjacent cells which are aggregated in an obfuscated region have the same color and the same number. Numbering is used just to better highlight the cells composing the same obfuscated region. We can observe that the granularity of obfuscated locations (i.e. a set of cells with identical label) is coarser for lower values of θ_{sens}.

7.3 Quantitative Evaluation

The plot in (Figure 4.a) shows the average number of cells of the initial partition exceeding the varying sensitivity threshold θ_{sens}. For example, it can be noticed that for $p = 0.8$ (i.e. high density of sensitive cells), there are, on average, about 65% of cells which have a sensitivity level greater than $\theta_{sens} = 0.3$.

The additional plots in Figure 4 report the results of three experiments that we now describe in some detail. Note that the results are not defined for the values of θ_{sens} for which no meaningful obfuscated space is found, that is, the solution either does not exist or degenerates into the whole space. Notice that, if a meaningful solution does not exist for a value of the sensitivity threshold, then such a solution does not exist for all values that are smaller than this value. The first two tests

(Figure 4.b, Figure 4.c) evaluate the computational cost of the algorithm. Such cost depends on two factors: the number of merge operations which are performed before a termination condition is met; the number of edges which are analyzed. For example, if $p = 0.8$ and $\theta_{sens} = 0.6$, an obfuscated space is found, on average, after 25 merge operations and the analysis of 100 edges; instead if $\theta_{sens} = 0.3$ the algorithm cannot generate a meaningful solution. We observe that both measures respect the worst case complexity analysis for $n = 100$.

The third test (Figure 4.d) computes the average size of the largest obfuscated location in resulting spaces. Such a value is determined by the number of original cells in the region. Such a measure gives an idea of the quality, i.e. the precision, of the obfuscated space: the higher is its value and the more imprecise are the obfuscated locations. For example, if $p = 0.8$ and $\theta_{sens} = 0.6$, then the largest obfuscated location consists on average of 5 cells. It can be observed that the value of the variable increases rapidly as θ_{sens} gets closer to the sensitivity level of the maximal partition (i.e. the whole space).

8. OPEN ISSUES AND CONCLUDING REMARKS

We have presented a location privacy-preserving infrastructure for the protection of privacy against inferences over sensitive locations. Such an infrastructure can be improved in several ways. In this conclusive section, we discuss the major issues we plan to address in our future research and report some conclusive remarks.

8.1 Open Issues

Open issues can be grouped in three main classes depending on whether they pertain to the privacy model, the computation of the obfuscated space or the architectural framework.

1. **Issues related to the privacy model.** A first issue concerns the extension of the privacy model towards a probabilistic model. In this work we have assumed that positions inside a relevant area are equally likely and that the likelihood is expressed by the probability density function (pdf) $1/Area$ where $Area$ is the size of the area. This assumption seems too restrictive. For example actual positions of individuals do not necessarily correspond to geometric points, as we have assumed in this model. For instance, in a football stadium the probability for a spectator to be inside the stadium likely depends on the maximum number of seats. To generalize the approach and thus account for arbitrary pdfs, the idea is to revise the model and then redefine the algorithm SensFlow to merge cells whose sensitivity values depend on arbitrary distributions of probabilities. Note that in such new model, the unreachable cells would be those for which the probability function has value 0. Another important problem concerns the specification of the threshold value (i.e. the value of θ_{sens}) in the privacy profile. Although ideally the threshold value ranges between 0 and 1, in practice the actual range depends on several factors including the nature and extent of the territory under consideration. Therefore it may be difficult for the user to define which value to assign to such variable in order to express his/her privacy

requirements. To deal with such an issue, the idea is to adopt the following heuristics: consider the two partitions, representing respectively the initial and the top partition and compute the corresponding sensitivity values G and L. Based on the shown properties, G is the greatest and L the least value of sensitivity of space partitions. Then we sample the interval $[L, G]$ to obtain a set V of candidate values for θ_{sens}, with $V = v_1 \ldots v_n$. For each value $v_i \in V$, an obfuscated space, if it exists, is generated, along with statistics pertaining also to the average size of obfuscated locations. The idea is that a user selects the value in V, based on the statistical properties of obfuscated spaces and the desired trade off between privacy and quality of service.

2. **Scalability** A major issue that will be addressed concerns the scalability of the obfuscation system. Such a concern is motivated by the fact that the proposed graph-based algorithm has complexity $O(n^2)$ where n is the number of cells. Therefore for large reference spaces or for fine-grained grids containing a significant number of sensitive places, obfuscation can result into a very expensive process. A promising direction of research that we are currently investigating is towards the development of algorithms which employ more efficient data structures for space representation such as space filling curves.

3. **Deployment onto a distributed architecture.** The obfuscation system consists of a number of building blocks that in the present work are only described from a functional point of view. The issue is how to distribute those functionalities over an LBS client-server architecture. Critical questions are: where to keep the repository of obfuscation space/s; and where to generate the obfuscation upon an LBS request.

A straightforward approach is to use a *trusted obfuscation server* (TOS). The TOS stores obfuscation spaces and their associated and univocally identified privacy profiles. Obfuscated spaces can be generated as follow. Users specify a privacy profile, for example by filling in a form. Then based on such input the TOS creates the obfuscated space and stores it in the local repository. Users might even be allowed to browse and select privacy profiles from the TOS repository. The client locally maintains a copy of the privacy profiles of interest. At run time, when the user issues a request, the client forwards the identifier of the privacy profile to the TOS along with the query and the client identity. The TOS gets the actual position p of the client and through the obfuscation enforcement module determines the cell of the obfuscated space containing p. The TOS finally forwards that cell to the LBS provider which answers the query based on the imprecise position being sent. The drawback of this scheme is the need of a dedicated and trusted server.

To overcome this limitation, an alternative approach is to distribute functionalities between the client and the server. Assume clients to be location-aware and have enough intelligence and resources. The idea is to let the client store locally the obfuscated spaces of interest. Obfuscated spaces can be generated directly by the LBS server or by a third party and then returned to the client in the form of maps. The user must trust the provider of obfuscated spaces; for example maps can be

accompanied by a digital certificate. The run-time operations are implemented at the client. Therefore upon a service request, the client determines the obfuscated location and then can directly forward the cell to the LBS provider without the need of intermediaries.

8.2 Concluding Remarks

Personal location information has peculiar characteristics with respect to privacy because representing both a quasi-identifier and a sensitive information. So far, research on location privacy has mostly focused on the former aspect, that is how to protect the users identity. In this paper we have presented a different perspective. In particular the focus is on how to prevent inferences on sensitive locations. Thus our research has been directed towards the specification of an architectural framework comprehensive of a privacy model and an obfuscation algorithm for the generation of obfuscated spaces. It is important to emphasize that the aforementioned viewpoints are complementary. On the one hand, k-anonymity techniques do not protect against sensitive location inferences, because the resulting generalized location can be sensitive; on the other hand, our obfuscation technique does not protect against data linkage and thus cannot be used to hide user's identity. A challenge for the future is the integration of those viewpoints.

References

Atallah, M. and Frikken, K. 2004. Privacy-preserving location-dependent query processing. In *ACS/IEEE Intl. Conf. on Pervasive Services (ICPS)*, pages 9–17. IEEE Computer Society.

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

Bettini, C., Mascetti, S., Wang, X. S., and Jajodia, S. 2007. Anonymity in location-based services: Towards a general framework. In *2007 International Conference on Mobile Data Management*, pages 69–76. IEEE.

Bonchi, F., Pedreschi, D., Turini, F., Malin, B., Verykios, V. S., Moelans, B., and Saygin, Y. 2007. Privacy protection: Regulations and technologies, opportunities and threats. In *Mobility, Data Mining, and Privacy*, pages 101–122. Springer.

Brun, L. and Kropatsch, W. 2006. Contains and inside relationships within combinatorial pyramids. *Pattern Recognition*, 39(4):515–526.

Cheng, R., Zhang, Y., Bertino, E., and Prabhakar, S. 2006. Preserving user location privacy in mobile data management infrastructures. In *Proceedings of the 6th Workshop on Privacy Enhancing Technologies (PET'06)*, volume 4258 of *Lecture Notes in Computer Science LNCS*, pages 393–412. Springer Berlin/Heidelberg.

Du, W. and Atallah, M. J. 2001. 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. ACM.

Duckham, M. and Kulik, L. 2005. 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.

Gruteser, M. and Grunwald, D. 2003. 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. ACM Press.

Kalnis, P., Ghinita, G., Mouratidis, K., and Papadias, D. 2007. Preventing location-based

identity inference in anonymous spatial queries. *IEEE Transactions on Knowledge and Data Engineering*, 19(12):1719–1733.

MACHANAVAJJHALA, A., GEHRKE, J., KIFER, D., AND VENKITASUBRAMANIAM, M. 2006. l-diversity: Privacy beyond k-anonymity. In *22nd IEEE International Conference on Data Engineering*. IEEE Computer Society.

MOKBEL, M. F., CHOW, C.-Y., AND AREF, W. G. 2006. 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.

MOLENAAR, M. 1998. An *Introduction to the Theory of Spatial Object Modelling for GIS*. CRC Press.

Open GIS Consortium. Open GIS simple features specification for SQL, 1999. Revision 1.1.

SWEENEY, L. 2002. Achieving *k*-anonymity privacy protection using generalization and suppression. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5):571–588.

VERYKIOS, V., DAMIANI, M. L., AND GKOULALAS-DIVANIS, A. 2007. Privacy and security in spatio-temporal data. In F. Giannotti and D. Pedreschi, editors, *Mobility, Data Mining, and Privacy-Geographic Knowledge Discovery*, pages 213–242. Springer.

WORBOYS, M. F. AND CLEMENTINI, E. 2001. Integration of imperfect spatial information. *Journal of Visual Languages and Computing*, 12(1):61–80.

XIAO, X. AND TAO, Y. 2006. Personalized privacy preservation. In *SIGMOD '06: Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 229–240. ACM Press.

A Proofs

Lemma 5.1 proof: The thesis can be restated as

$$SL_{reg}(c) \leq SL_{reg}(c_1) \vee SL_{reg}(c) \leq SL(c_2)$$

We define

$$W(c) = \sum_{ft \in \mathcal{FT}_{ses}} score(ft) Area_{fea}(c, ft)$$

The first inequality can be rewritten, using the sensitivity definition, as

$$\frac{W(c)}{Area_{rel}(c)} \leq \frac{W(c_1)}{Area_{rel}(c_1)}$$

Since $W(c) = W(c_1) + W(c_2)$, and $Area_{rel}(c) = Area_{rel}(c_1) + Area_{rel}(c_2)$, the inequality is equivalent to

$$\frac{W(c_1) + W(c_2)}{Area_{rel}(c_1) + Area_{rel}(c_2)} \leq \frac{W(c_1)}{Area_{rel}(c_1)}$$

and, with some simple algebraic operation, to

$$W(c_2) \cdot Area_{rel}(c_1) - W(c_1) \cdot Area_{rel}(c_2) \leq 0$$

The same expansion, applied to the second inequality of the thesis, gives

$$W(c_2) \cdot Area_{rel}(c_1) - W(c_1) \cdot Area_{rel}(c_2) \geq 0$$

Since one of the two inequalities must be necessarily true, the thesis is true.

Theorem 5.2 proof: Assume $\mathcal{C}_A \neq \mathcal{C}_B$ otherwise the demonstration is trivial. Now,

suppose that $\mathcal{C}_A$ is directly derived from $\mathcal{C}_B$, using a single merge operation. Lemma 5.1 states that the resulting cell c has an equal or a smaller sensitiveness level than the replaced cells. Based on definition 4.2, the level of sensitiveness of a partition is determined by the higher level of sensitiveness of its cell. It follows that $SL(\mathcal{C}_A) \leq SL(\mathcal{C}_B)$. Consider now a sequence of merging operations each generating an intermediate partition I_0, I_1..., I_k such that $\mathcal{C}_A$:

$$\mathcal{C}_A \succcurlyeq I_1 \succcurlyeq \ldots \succcurlyeq I_k \succcurlyeq \mathcal{C}_B$$

Based on the previous lemma it follows that:

$$SL(\mathcal{C}_A) \leq SL(I_1) \leq \ldots \leq SL(I_k) \leq SL(\mathcal{C}_B)$$

therefore $SL(\mathcal{C}_A) \leq SL(\mathcal{C}_B)$ that is what we wanted to demonstrate.

Corollary 5.3 proof: In case $SL_{par}(Max) \leq \theta_{sens}$, then Max is an obfuscated space, hence the proposition (1) is true. The assertion (2) follows from the anti-monotonicity of $SL(\mathcal{C})$ and the ordering of the three partitions involved: $Max \succcurlyeq \mathcal{C} \succcurlyeq \mathcal{C}_{in}$. Finally, (3) is a direct consequence of (2): for every partition $\mathcal{C}$, $SL_{par}(\mathcal{C}) \geq SL_{par}(Max) > \theta_{sens}$

Maria Luisa Damiani received the Laurea cum laude in Computer Science from the University of Pisa, Italy and the Phd in Computer Science from Ecole Polytechnique Fédérale de Lausanne. Currently she is Assistant Professor at the University of Milan. Main research interests include: spatio-temporal data modeling, environmental applications, security and privacy in mobile computing.

Claudio Silvestri received the MS degree in Computer Science summa cum laude from the University Ca' Foscari of Venezia, and the PhD in Computer Science from the same University in 2006. He is currently a research fellow at Department of Computer Science and Communication, University of Milan. His research interests include: position based secure access to information systems, privacy preservation in location based services and spatio-temporal data disclosure, algorithms for frequent pattern mining and techniques for approximate data mining on distributed data and data streams. Claudio Silvestri is a member of the program committee of the Track on Data Mining of the ACM Symposium on Applied Computing since 2006.

Elisa Bertino is professor of computer science at Purdue University and research director of CERIAS. Her main research interests include security, database systems, object technology, multimedia systems, web-based information systems. Elisa Bertino is a Fellow member of IEEE and a Fellow member of ACM. She served as co-editor in chief of the VLDB Journal. She received the 2002 Technical Achievement Award by the IEEE Computer Society "For outstanding contributions to database systems and database security and advanced data management systems" and the 2005 Tsutomu Kanai Award by the IEEE Computer Society for "Pioneering and innovative research contributions to secure distributed systems".