Privacy of Location Obfuscation

Debajyoti Bera
IIIT-Delhi
Dwarka, New Delhi, India
Email: dbera@iiitd.ac.in

Vikram Goyal
IIIT-Delhi
Dwarka, New Delhi, India
Email: vikram@iiitd.ac.in

Anuj S. Saxena
KCC Institute of
Technology and Management
Greater Noida, U.P., India
Email: anuj102@gmail.com

***Abstract*—There are many results on the possibilities of privacy leak in location-based services, and many remedies have also been suggested. In this paper, we look at this problem from an altogether different perspective - claiming that privacy leaks are most likely inevitable for a lot of remedial mechanisms, and therefore it is necessary to provide means to a user to detect such leaks in advance. We specifically target the mechanisms using location obfuscation (aka. blurring) for providing location privacy for continuous queries. We give a definition to measure leak of location privacy. We also give a framework to compute it for a wide range of location obfuscation strategies. We then evaluate a few common strategies both analytically and experimentally and use the results to propose provably efficient algorithms for them.**

I. Introduction

Personalised services for mobile devices are currently hot on wheels due to the ubiquity of these devices and the increasing popularity of personalised services. The most popular form of these services enable us to obtain information on-the-go, information which depend on the spatio-temporal location and the identity of the requesting user. No wonder these services have caught the fancy of global population [8], and currently, there are many commercial ventures engaged in large business around the world [6, 26].

On the other hand, privacy advocates claim that such services come with an inherent risk of privacy breach [2, 7, 28], in the sense that service is provided based of private information. Though they accept that private information has to be necessarily shared, they frown upon the fact that the general population is usually unaware of the breach, but instead heavily mind-washed about the usefulness. An acceptable solution to them would be to inform users about the risks involved and allow them to make an informed decision.

The potential of the privacy breach hinted above has recently spurred a lot of research in the privacy of database community [21, 24]. The focus can be privacy of identity, of location, of queries or other sensitive information. Various possible breaches have been studied, and remedies have been suggested for most of them. The problem becomes more complicated and critical for mobile users[1] sending continuous queries [23, 9]. It has been shown that it is possible to correlate them for mining significant amount of sensitive information [4, 12]. See Section V for a brief overview of previous work in this area.

A. Problem Setting

We choose a scenario where location is the only sensitive information of a user. We also assume an hypothetical adversary who knows the details of the particular mechanism in use. With this setting, we investigate the problem of preserving location privacy for a moving user who is continuously sending queries to some location-based service provider; we specifically address the problems arising due to leakage of information from multiple location updates.

[1]Pun intended! Users who are moving as well as also carrying a mobile.

B. Overview and Contributions

One way to resolve above mentioned privacy breach is to mask the sensitive information in a query in a way to prevent, or at least reduce, privacy leak yet allow the query to be executed. While most current research is focused on how to encode (different parameters of the) queries to prevent leakage [22, 16, 10], we take a different point of view motivated by the following observation.

> Sometimes, even the best privacy preserving mechanism may fall short of user expectation.

Therefore, we propose that privacy preserving mechanisms should operate in a best-effort manner and actively involve the end-users in the process – at least, for deciding how much privacy to preserve and reverting when the desired level cannot be met. This immediately mandates a *definition of privacy* which is general enough to be applicable to a variety of (similar) mechanisms, yet, computable in an efficient way. Furthermore, since end-users would have to adapt to this definition, it should be concise (preferably a single number) and easy to relate to. Our first contribution is a usable definition of privacy, and later we also build a framework for computing it.

One problem privacy preserving mechanisms often face is protection against indirect leakage. Indirect leakage could come from various sources, ranging from external information to hidden correlation in a series of related queries. Consider, for example, a user who is asking for nearby drugstores continuously after every 5 minutes. It might be possible to derive a rough outline of his route with reasonable detail by correlating all his queries. Can he still seek information without revealing his current or any past location?

A common approach to the above problem is to obfuscate (*aka.* blur) the actual location by a larger region and send this blurred region to the service provider [1, 12, 22]. The obvious limitation of this approach is the degradation of response quality – since now the possible location of the user has multiplied. But, the problem of leakage from correlated queries remain. We will show that location information could still be obtained from multiple continuously made queries, since, we assume that the algorithm to determine the blurred area is known to everyone [2].

Take, for instance, the scenario of User-C in Figure 1. Location is represented by the smaller squares and location anonymisation is obtained by reporting the bigger square region (e.g., C_{03}) which he belongs to. As User-C moves along the shown path, he reports to the service provider the region C_{22} for the first two moves and the region C_{33} in the next two moves. The service provider can then easily figure out that the actual locations of the second move (lower right corner square of C_{22}) and the third move (upper left corner square of C_{33}). This is a very simplistic example, and in fact, the most simple case of privacy leak. More of these is discussed in Section IV-E.

Our specific focus in this paper is on how much information can be inferred by correlating obfuscated regions. Most solutions using

[2]Privacy through obscurity has significant practical drawbacks.

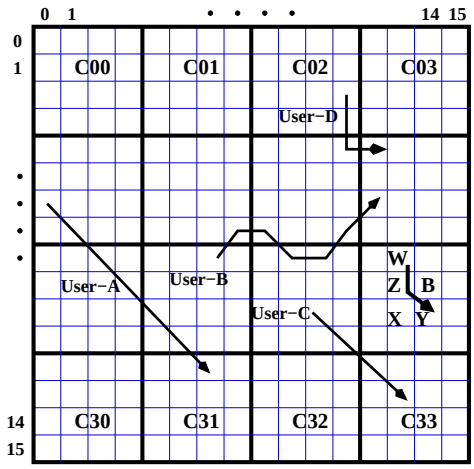

Fig. 1. 4×4 Disjoint Square Encoding

location obfuscation have used square/rectangular or circular regions in the past. We show that privacy properties differ significantly when using one shape of the region vs another, and this can be exploited to derive better obfuscation mechanisms suiting a user's need. With these aims in mind, we initiate the analytical study of privacy leakage of blurring-based obfuscation mechanisms and then experimentally evaluate mechanisms using obfuscation regions of the shapes of square (Figure 1), triangle (Figures 4, 5) and hexagon (Figure 6)[3].

We will later show how *crossing the boundaries* between two obfuscated regions might reveal enough information to exactly predict the current (and/or) previous location(s). This and other problems with location obfuscation have been dealt with various heuristics like delaying location update, or specially treating the crossing area. To get more insight about these schemes, we consider mechanisms which use overlapping obfuscated areas (Figures 2 and 8) – when a user moves from one region to another, this explicit information is hidden by reporting a third region obfuscating the crossover area. We show that such obfuscation mechanisms also suffer from information leakage but significantly less than their non-overlapping versions.

Finally, given the increasing popularity of these services and mechanisms, the need for user-friendly privacy levels has been often been pointed. But to the best of our knowledge this issue has not been adequately answered. Quoting from [9],

> "Existing privacy-preserving LBS frameworks are designed from the technology's prospective. There is still need to study the location privacy issue from the user's prospective. For example, how can a casual user define privacy requirements. Is it possible to define privacy levels as low, medium, and strict, and then users would choose among them. How can a user achieve a trade-off between the privacy requirements and the quality of services. How can the user evaluate the privacy risk she has from using a certain LBS."

Our final contribution is a scheme for user-friendly privacy levels that capture the essential trade-offs between service cost and quality.

C. Outline

We describe the architectural setup used in this paper in Section II, where we first describe the query model (Section II-A), then followed by the notion of location obfuscation (Section II-B), service provider and proxy (II-C). In Section III, we discuss the theoretical results for any general obfuscation mechanism fitting our scheme of things. We begin with definitions and notations (III-A) and the proceed to give a definition of privacy to suit our needs (Section III-B). It is then followed by our main set of general results on properties of privacy of location obfuscation (Sections III-C, III-D). We then do a case study of different mechanisms using overlapping and non-overlapping (disjoint) obfuscation regions in Section IV. We first describe few disjoint cases (Section IV-A), analytically derive some properties (Section IV-B) and then use these results to construct an efficient algorithm to compute privacy for the corresponding disjoint cases (Section IV-C). We then also discuss a few of the overlapping cases (Section IV-D) and different types of privacy leaks (Section IV-E). Then we give a comparative study of the different mechanisms in the case study (Section IV-F) and propose one possible way to define levels of privacy suitable for end-users. We finish this paper by discussing some important future directions of this work in Section VI. Related work is discussed in Section V.

[3]One reason behind considering these three shapes is that they are the only three possible regular tilings of a plane [30].

II. Architecture

In this section we describe the different components of the architectural setup used in this paper – our model of a location-based service, a query model and our model of location obfuscation.

We consider the standard client-server model for location-based services for our purpose [5]. Following existing research, privacy is preserved by using a *privacy-preserving proxy* service through which all queries are relayed. This service could be provided by a trusted third-party or by the client itself.

A. Query Model

A user can request information by sending a query to the service (which, in our case, is received, modified and retransmitted by a proxy) consisting of the following parameters: a user id, current timestamp, location information, query text (and other information needed to answer the query). We will henceforth ignore query text and other information since we are only interested in location privacy.

User Id: It is usually assumed that users of a location-based service are assigned unique ids which serve two purpose: keep track of the source of a query and maintain the context for multiple queries in a session. For multiple query sessions, it might be necessary to use the same id for billing purposes or other technical reasons. Even for short sessions, it has been proved that multiple queries by the same user in short timespans can be correlated to deduce that they belong to the same user [3]. So, we make the worst-case assumption that every user uses a unique id for all queries (during a session); however, the id is opaque and it reveals no information other than correlating multiple queries. Furthermore, here we will be dealing with the privacy of a particular user, and therefore, will not explicitly mention the user id as part of the query.

Continuous Query: We will consider a discrete form of time and assume that there is one query per discrete time interval throughout the session. Though our results can be extended to deal with missing queries infrequently during a session, (which can also be modelled by allowing the user to speed up/down during certain intervals), but this assumption will help us to derive precise bounds on the privacy leak during a query session. For a similar reason, we require that the user should be moving at a uniform speed. For sake of brevity, we assume that a query happens at the *beginning of a time interval*, i.e., *after the t-th request* and *at time t* will mean the same interval.

B. Location Obfuscation

Location obfuscation is a very common technique of reporting location such that (1) the exact location is not reported (2) information obtained using the obfuscated location can be used to derive the specific information related to the exact location. The gain in privacy by sending obfuscated location is balanced by the loss in quality of results, e.g., the returned result set might contain irrelevant results; studying this trade-off is another line of work that we do not consider in this paper. Here we are concerned about the obfuscation itself.

The most common approach for obfuscation involves dividing the entire region into smaller areas and instead of reporting the exact location, reporting the corresponding area containing that location. Naïvely following this approach might not actually preserve privacy, as illustrated before in Figure 1.

Our investigation revealed that different ways to build the regions lead to different type of privacy leakage. Therefore, it becomes crucial to understand the properties of regions with different shapes, sizes and layout – and choose the best one based on user requirement. Observe that once an area is *tiled* into regions, leakage of location information becomes a property of the *tiling* (aka. tessellation). It is well known in tessellation research community that "...stringent restrictions must be imposed on tilings if any meaningful results are to be obtained..." [18]. Yet, we want our results to be as general as possible, so that they are applicable to different forms of tilings and corresponding mechanisms.

Therefore, we begin with a general form of obfuscation, where, the entire region is tiled into uniform polygonal *cells* which are further subdivided into smaller disjoint uniform polygonal *blocks*. These blocks form the *basic unit of location* i.e. the exact location of a user is going to be represented by the containing block – we will interchangeably use location and block to mean the same thing. Observe that the blocks represent location, so they can be as small as necessary and are necessarily disjoint. On the other hand, we don't a priori require the cells to be disjoint. Also, the cells are constructed a priori and known to everyone, unlike in some obfuscation algorithms where the blurred region is determined based on current location and other parameters (e.g., k-anonymity [27]). Obfuscation tries to minimise the loss of privacy by reporting the cell instead of the block as the location of any user. We denote the obfuscation *mapping* by G, i.e., the cell $G(b)$ is reported for the location b. The definition and essential properties of G are discussed later in Section III.

There are several approaches, all based on blurring current location, that have been proposed for location obfuscation. Take for example, using mix zones [4] and delaying queries [15] – both of these withhold sending a blurred region when there is an increased chance of disclosure of the current location. We will prove later that the only risk of disclosure happens when crossing cells – there is absolutely no risk when moving inside a cell. Therefore, to get an idea of such mechanisms which employ different behaviour at the border, we allow our cells to be overlapping – so that, locations at the border belong to two (or more) cells. We observe that overlapping also does not completely remove, even though it significantly reduces, the chances of a privacy breach. Our definitions of privacy are nevertheless applicable to mechanisms with delayed queries, anonymous zones, dynamically computed blurred regions, or other approaches and some of our theoretical results can also be extended to the same.

C. Service Provider & Proxy

We denote the honest-but-curious location-based service by $\mathcal{R}$. Of course, $\mathcal{R}$ has to continue providing service and thus cannot deviate from its correct behaviour; but, it may try to infer location information of users based on the information it has. We make no assumption on the computation power of $\mathcal{R}$; in fact, all our lower bounds will be information theoretic. Similarly for storage, we will let $\mathcal{R}$ to store all locations reported in the past. However, we need to make one necessary assumption: $\mathcal{R}$ does not have access to additional background/contextual knowledge. This assumption may not be valid for real services but is essential for our analysis because there can be no bound on the entire background knowledge possessed by any person or service [20].

Any query to the service provider is modified by the proxy to enhance privacy. Its first and foremost goal is to convert the actual location b to an obfuscated cell $G(b)$. It then performs an additional proactive check to determine if reporting $G(b)$ may still cause privacy violation. It is allowed to use the location history of the user for this detection; of course, we would like this storage requirement to be minimal. A typical application scenario would be to inform the user about any violation and seek input from him about whether to abort or continue (by modifying the obfuscation level or otherwise).

III. Theoretical Framework

In this section, we give the definitions and results to build the necessary theoretical framework for analysing privacy of location obfuscation schemes. The main focus is to come up with the right notions that can be efficiently computed. We apply the framework to a few specific obfuscation schemes in the later sections; however, it is designed to accommodate a wide variety of possible schemes.

A. Basic Definitions and Notations

The entire region is tiled using (topologically closed) congruent polygons called *blocks*, denoted by roman literals, e.g., b, c_2. Location of a particular point is represented by its enclosing block. Two blocks are defined to be *neighbouring* if it is possible to move from one to another (by crossing a side or by crossing a corner). The blocks are then grouped into (topologically closed) polygonal regions called *cells* such that every block belongs to at least one cell. The cells will be denoted by Greek literals, e.g., β, π_2. We allow overlapping cells, so, a block can belong to multiple cells. *Encoding* is a function that specifies which blocks constitute a cell and is denoted by the mapping M; therefore, $M(b)$ is the *set* of all cells that block b belongs to. if a block b belongs to a single cell, then $M(b)$ is a singleton, in which case we simplify the notation $M(b)$ to denote that single cell. We may sometimes identify a cell β as the set of blocks contained in β.

The encoding can be disjoint – the cells are disjoint, or overlapping – the cells are overlapping. As an illustrative example of disjoint encoding, refer to Figure 1. Cells are large squares shown by darker borders, and each cell contains 16 square blocks. In disjoint encoding each block is included in only one cell, which is reported as the obfuscated region for that block.

In overlapping encoding, each block may belong to more than one cell. An example is shown in Figure 2. Here, the entire region is partitioned using two overlapping grids of cells – one shown by darker lines, and other shown using dotted line. Each block here belongs to exactly two cells, and which cell to report as the obfuscated region is decided by the mechanism. We describe them in more detail in Section IV-D.

We want our results to hold true for any kind of tiling and any possible encoding. However, as evident from the literature about tiling research, it is very difficult, often impossible, to prove results for a generic tiling [4][18]. We workaround this problem by specifying

[4]For a pictorial list of various kind of tilings, refer to http://www2.stetson.edu/ efriedma/mathmagic/0701.html

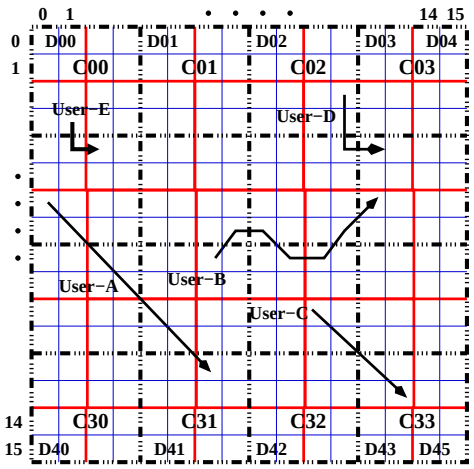

Fig. 2. Overlap Square Grid Encoding

certain properties, as and when required, that a tiling or an encoding should meet for our results to hold.

Property 1: A cell is a connected group of blocks and cannot be too small, i.e., every block in a cell must have at least two neighbouring blocks belonging to the same cell.

We will also make use of the following properties about the tiling and the encoding. Even though they do not hold for all cases, they certainly hold for all the different tilings we explicitly analyse later.

Property 2 (Compact Tiling): Consider any three blocks x, y, z which are neighbours of each other. The tiling of blocks is called *compact* if for any neighbour w of z, there is a common block b which is a neighbour of both w and z as well as (at least) one of y and x.

Property 3 (Compact Encoding): For any compact tiling, consider blocks x, y, z and w as in Property 2. Then, if all of them, x, y, z and w, belong to the same cell β, then the block b as mentioned in that property is also in the cell β.

As an example, consider the encoding in Figure 1. We have shown one possible b given x, y, z, w, but it is easy to verify that both the tiling and the encoding are compact. This property will become useful in trying to construct an alternative sequence of blocks which would look the same as the actual sequence of blocks (e.g., W-B-X is an alternative path corresponding to the actual path W-Z-Y). Not every tiling is compact, a counterexample is shown in Figure 3.

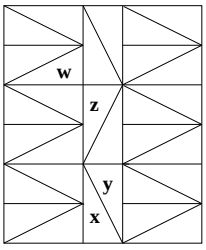

Fig. 3. Example of tiling which is not compact

When a user is in block b, one of the cells in $M(b)$, according to an obfuscation strategy, is reported. A *path* is sequence of blocks denoting a possible movement of some user. Given a path, the corresponding reported cells is defined as the *cell-path* – the first cell in a cell-path is one of the mapped cells of the first block, second cell is one of the mapped cells of the second block and so on. The i-th block of a path p is denoted by p_i and the length of a path, denoted by $|p|$, is the number of blocks in the path (similarly for cell-path). Since we assume constant speed of movement, we recognise the length of the current path as also the current time and the i-th block as the block during time i (the first block is at time 1). By any *location obfuscation strategy*, we mean all of the above, namely, the tiling, the encoding, and the block selection strategy.

***Definition** 1 (Location Obfuscation):* Location obfuscation, denoted by G, is given by a tiling, an encoding, and a length-preserving function from the set of paths (over blocks) to the set of cell-paths (over cells).

For the sake of simplifying notation, we will let $G(p)$ denote the cell-path corresponding to a path p where the tiling and the encoding are clear from the context.

B. Privacy

There are two kinds of privacy violation possible with the reporting of obfuscated location information. The first and the obvious kind is where the obfuscated location (cell) somehow reveals the actual location (block). However, this kind of leak of information can be prevented almost always by careful obfuscation. The second violation can occur due to privacy leaks from the context or the history of previous queries. We want our proxy to be able to compute any possible privacy violation - specifically, of the second kind – using context information from past queries of a user who is continuously making queries on the move. We need a few definitions to quantify this kind of leakage. First, we define our basic idea of privacy when a new request is made.

***Definition** 2 (Privacy):* After reporting a cell-path $\pi_1 \dots \pi_t$ while traversing a path $b_1 \dots b_t$ in t steps, the privacy of location at a past time i ($1 \leq i \leq t$) is defined as the probability that $\mathcal{R}$ is not able to predict the actual block at time i:

$$\rho_i^t = \Pr[\mathcal{R}(\pi_1 \pi_2 \dots \pi_t, i) \neq b_i]$$

We say that the location at time i after the t-th request is ϵ-private if $\rho_i^t \geq \epsilon$.

We define the *privacy* after the t-th request as

$$\rho^t = \min_{1 \leq i \leq t} \rho_i^t$$

Note that, $\rho^t \in [0, 1]$ and allows mechanisms to offer different levels of privacy based on its value. We leave the question of how to construct these levels for future work, and, in this work, look at the simplest possible privacy guarantees that our proxy can hope to meet.

***Definition** 3 (Minimal Privacy):* A location obfuscation function G is said to ensure *minimal privacy* at time t, if $\mathcal{R}$ is not able to predict *any* of the previous locations *with certainty*, i.e., $\rho^t > 0$.

There is a limitation of this definition – once minimal privacy is unsatisfied, it remains unsatisfied forever because ρ_i^t, by definition, is a non-increasing function for a fixed i and increasing t. However, for practical reasons, it may happen that a user knowingly allows $\mathcal{R}$ to know some locations in his path, and still wants to retain privacy on other locations. To capture this requirement, we define a more meaningful but weaker notion of privacy, which we use henceforth.

***Definition** 4 (Disclosure):* The t-th request is said to *disclose* a past location i if $\mathcal{R}$ is able to predict the i-th location with *certainty* after receiving the t-th request: $\rho_i^t = 0$.

***Definition** 5 (Weak Minimal Privacy):* G is said to preserve *weak minimal privacy* at time t, if the t-th request did not create any *new disclosure* of any past location i.e.

$$\rho_i^{t-1} > 0 \implies \rho_i^t > 0$$

C. Results

Determining if the latest request preserves weak minimal privacy, ab initio, requires computing the disclosure at all of the previous times. This itself, notwithstanding the amount of computation/resource required to compute disclosure at some particular time in the past, could make it difficult to compute location privacy. However, in this section we show how to use some properties of location obfuscation to efficiently detect disclosure.

Detecting disclosure and computing privacy depends very much on the exact strategy of location obfuscation. Here we look at some of the theoretical properties of general obfuscation mechanisms, with an eventual goal of applying them to the disjoint tiling based mechanisms we explicitly discuss in Section IV-B. However, instead of discussing the properties of the specific obfuscation functions, we analyse them in a more generic manner in this subsection – our results can be applied to, often with minor or no modifications, to other similar obfuscation strategies (for example, considering triangular cells instead of square cells). Following is a list of essential properties of obfuscation mechanisms that we require for our results.

Usually, cells are created in a disjoint manner, and in such cases, every block is uniquely mapped to a particular cell. For these types of obfuscation, the mapped location depends *only* on the current block. Any such G has the following nice property.

Property 4 (Independent): Let $b_i, b_{i+1}, \ldots$ represent the locations at time $i, i+1, \ldots$. Then G is independent if

$$G(b_i \cdot b_{i+1} \cdot b_{i+2} \ldots b_j) = \big(G(b_i) \cdot G(b_{i+1}) \cdot G(b_{i+2}) \ldots G(b_j)\big)$$

Beside the usual disjoint cells, we will also consider overlapping cells in this paper. If cells are overlapping, then some blocks may belong to multiple cells. For this case and even more complicated types of obfuscation, the mapping of a block might also depend on other factors, such as, previous block(s), current path, time of day etc. We will use $G(b)$ to denote the obfuscated cell for a length 1 path consisting of the only block b and $G(b_t|b_1 b_2 \ldots b_{t-1})$ to denote the obfuscated cell at time t for the path $b_1 b_2 \ldots b_t$. We will only consider one type of obfuscation function for overlapping cells; the details are described later in Section IV-D, but a basic property of that function is given below. Informally, for a *local* obfuscation function, the mapping of a block depends only the current block and the mapping of the earlier block.

Property 5 (Local): Consider two paths of equal length ending at the same block $p_1 a_1 b$ and $p_2 a_2 b$. Then, for a local G, $G(b|p_1 a_2) = G(b|p_2 a_2)$ iff $G(a_1|p_1) = G(a_2|p_2)$.

For a G which is either independent or local, if we know the mapping of two paths that cross at some point, then we can construct the mapping of) the two other valid paths obtained) by concatenating the segments before and after the crossing point from different paths.

***Definition** 6 (Composable):* An obfuscation function G is called *composable* if it satisfies the following property. Consider two paths pq and $p'q'$ such that $G(pq) = G(p'q') = \pi\psi$ (π has same length as p). Then, assuming pq' is also a valid path, $G(pq') = \pi\psi$.

***Lemma** 1 (Composability):* Consider any G which is independent or local. Then, G is composable.

Proof: Obvious for Independent. Simple for local. ∎

D. Computing Privacy

We begin with a few concepts needed to compute the disclosure at any particular time.

Notation: We will use the notation $\Gamma(a, b)$ to mean that the blocks a and b are neighbours, and extend this notation to $\Gamma(a) = \{b \mid \exists b,\ \Gamma(a, b)\}$ to denote the blocks which are neighbours of a. Similarly, $\Gamma(A)$ will denote the blocks which are neighbour to any block in set A.

***Definition** 7 (Consistent Path):* A path p is consistent with a cell-path π if $G(p) = \pi$.

***Definition** 8 (Possibility):* For a cell-path π of length l, the set of possible blocks the user could be at an earlier time $t \leq l$ is defined as

$$\text{poss}(t|\pi) = \{b \mid \text{there is some path } p \text{ consistent with } \pi \text{ such that } p_t = b\}$$

While consistent paths represent different paths that would look the same after encoding, possibilities represent different blocks that could be used in an indistinguishable way in place of a particular block. Computing possibilities is therefore a basic operation and our algorithm to compute privacy also relies of efficient computation of possibilities.

Possibility depends on how G treats neighbouring blocks, specially at boundaries. We want G to be well-behaved in this respect.

Property 6: Possibility is always a set of *connected* blocks.

***Lemma** 2:* For any cell-path $\pi_1 \ldots \pi_t$ and for all i,

1) $\text{poss}(1|\pi_1) = \pi_1$ (for a cell-path consisting of a single cell, the possibility is clearly all blocks in the cell)
2) $\text{poss}(i|\pi_1 \ldots \pi_t) \subseteq \pi_i$ (by definition)
3) $\text{poss}(i|\pi_1 \ldots \pi_t) \subseteq \text{poss}(i|\pi_1 \ldots \pi_{t-1})$ (possibility is non-expanding with time)
4) $\text{poss}(t+1|\pi_1 \ldots \pi_t \pi_{t+1}) \subseteq \Gamma(\text{poss}(t|\pi_1 \ldots \pi_t))$ (consistent paths can only be extended by moving to neighbouring blocks)

Consider any path p consistent with some cell-path π. Let, $a = p_i$ and $b = p_{i+1}$ for some $i < |p|$. Then, $a \in \Gamma(b)$. Since, $\text{poss}(i|\pi)$ is the set of all such a and $\text{poss}(i+1|\pi)$ is the set of all such b, it is clear that possibility maintains the notion of neighbourhood at all times.

***Lemma** 3:* For any cell-path π, the following hold for any $i \leq |\pi|$:

1) $\text{poss}(i-1|\pi) \subseteq \Gamma(\text{poss}(i|\pi))$
2) $\text{poss}(i|\pi) \subseteq \Gamma(\text{poss}(i-1|\pi))$

The next two lemmas tell us how to compute possibilities when a new cell is reported. Lemma 4 follows directly from Lemma 2.

***Lemma** 4:* Consider a cell-path π of length t, and let the next reported cell be π_{t+1}. Denote $\pi_1 \ldots \pi_{t+1}$ by π'. Then, the new current possibility satisfies the following:

$$\text{poss}(t+1|\pi') \subseteq \pi_{t+1} \cap \Gamma(\text{poss}(t|\pi))$$

Unfortunately Lemma 4 does not tell us how to compute the exact current possibility when a new cell is reported – it is because, not all neighbouring blocks may be encoded equally by G. We defer the expression for actual computation to the case-studies in Section IV.

***Lemma** 5 (Update Lemma):* Consider a cell-path π of length t, and let the next reported cell be β. Denote the new cell-path $\pi \cdot \beta$ by π'. Then, for any time $i < t$, $\pi_i = \pi'_i$ and the updated possibility for time i is given by

$$\text{poss}(i|\pi') = \text{poss}(i|\pi) \cap \Gamma(\text{poss}(i+1|\pi'))$$

Proof: It is straight forward to show that the LHS is a subset of the RHS. We will give a brief sketch of the proof that the RHS is a subset of the LHS.

Take any a in both $\text{poss}(i|\pi)$ and $\Gamma(\text{poss}(i+1|\pi'))$. The first inclusion implies the existence of a path p consistent with π such that $p_i = a$. Denote the path segment $p_1 \dots p_i$ by q.

The second inclusion implies the existence of a block b such that $\Gamma(a, b)$ and existence of a path p' consistent with π' such that $p'_{i+1} = b$. Denote the path segment $p'_{i+1} \dots p'_{t+1}$ by q'.

Now, consider the series of blocks qq' – it clearly forms a path, and is consistent with π' since G is composable. Also, $(qq')_i = a$. Therefore, $a \in \text{poss}(i|\pi')$. ∎

Next we state two most important theorems, disclosure can happen only when some border is crossed (Theorem 1), and only for boundary blocks (Theorem 2). But first we need a technical lemma which stipulates that if possibility at some past instant t remains unchanged after a location update, then the possibilities at all instants earlier than t also remains unchanged.

***Lemma** 6:* Consider a cell-path π of length t. If for some i, $\text{poss}(i|\pi_1 \dots \pi_{t-1}) = \text{poss}(i|\pi_1 \dots \pi_t)$, then, for all $j \leq i$, $\text{poss}(j|\pi_1 \dots \pi_{t-1}) = \text{poss}(j|\pi_1 \dots \pi_t)$ – equivalently, if for some i, $\text{poss}(i|\pi_1 \dots \pi_t) \subsetneq \text{poss}(i|\pi_1 \dots \pi_{t-1})$, then for all $j \geq i$, $\text{poss}(j|\pi_1 \dots \pi_t) \subsetneq \text{poss}(j|\pi_1 \dots \pi_{t-1})$.

Proof: Denote $\pi_1 \dots \pi_{t-1}$ by π. The claim is true for $j = i$; we will prove it for $j = i - 1$ – the rest of the claim will follow inductively.

$$\begin{aligned} \text{poss}(i-1|\pi \cdot \pi_t) &= \text{poss}(i-1|\pi) \cap \Gamma(\text{poss}(i|\pi \cdot \pi_t)) \\ &= \text{poss}(i-1|\pi) \cap \Gamma(\text{poss}(i|\pi)) \\ &= \text{poss}(i-1|\pi) \end{aligned}$$

The last equality holds since $\text{poss}(i-1|\pi) \subseteq \Gamma(\text{poss}(i|\pi))$ (Lemma 3). ∎

This theorem says that, if the current reported cell is same as the last one, then weak minimal privacy is preserved. In fact, for such cases, all existing possibilities remain completely unchanged after the current cell is reported.

***Theorem** 1:* Consider a cell-path π of length t such that $\pi_{t-1} = \pi_t$ (last two moves were in the same cell). Then, there is no new disclosure; in fact, for all $i \leq t-1$, $\text{poss}(i|\pi_1 \dots \pi_t) = \text{poss}(i|\pi_1 \dots \pi_{t-1})$.

Proof: We first prove the claim for $i = t - 1$. Recall, $\text{poss}(t-1|\pi_1 \dots \pi_t) = \text{poss}(t-1|\pi_1 \dots \pi_{t-1}) \cap \Gamma(\text{poss}(t|\pi_1 \dots \pi_t))$ by Lemma 5.

By Lemma 4, $\text{poss}(t-1|\pi_1 \dots \pi_{t-1}) \subseteq \Gamma(\text{poss}(t|\pi_1 \dots \pi_t))$, which shows that $\text{poss}(t-1|\pi_1 \dots \pi_t) = \text{poss}(t-1|\pi_1 \dots \pi_{t-1})$.

Now, applying Lemma 6, we get that for $i \leq t-1$, $\text{poss}(i|\pi_1 \dots \pi_t) = \text{poss}(i|\pi_1 \dots \pi_{t-1})$. ∎

The previous theorem shows that only moves which change cell could potentially reveal any past location. The next theorem shows its dual; only locations involved during change of cell could potentially be disclosed (anytime). Compactness seems like a necessary condition for this result to hold; it is possible to construct an encoding which is not compact and in which a block not involved in boundary crossing is disclosed (we omit a counterexample due to space constraints).

***Lemma** 7 (Neighbourhood Lemma):* For any compact encoding, consider any cell-path of length t. Then, the following properties hold for any $1 < i < t$ such that $\pi_{i-1} = \pi_i = \pi_{i+1}$.

- If $\text{poss}(i-1|\pi)$ contains at least two neighbouring blocks, then $\text{poss}(i|\pi)$ contains at least two neighbouring blocks.
- If $\text{poss}(i+1|\pi)$ contains at least two neighbouring blocks, then $\text{poss}(i|\pi)$ contains at least two neighbouring blocks.

Proof: Follows from the definition of compact encoding. ∎

***Theorem** 2:* Consider a cell-path π of length t, and let the next reported cell be β; denote, $\pi \cdot \beta$ by π'. Consider any sequence of reported cells from time i to j ($j \geq i+2$) such that $\pi_i, \dots, \pi_j$ are all same cell α and boundary was crossed at i and j (i.e., $\pi_{i-1} \neq \pi_i = \dots = \pi_j \neq \pi_{j+1}$).
If none of $\pi_t : i < t < j$ were disclosed earlier, and, π_j is not disclosed after the last move, then none of $\pi_t : i < t < j$ is disclosed after the last move.

Proof: Since π_j is not disclosed, therefore $\text{poss}(j|\pi')$ has at least two neighbouring blocks. Now, apply Lemma 7 backwards for $\pi_j, \pi_{j-1}, \pi_{j-2}$, then $\pi_{j-1}, \pi_{j-2}, \pi_{j-3}$ and so on until $\pi_{i+2}, \pi_{i+1}, \pi_i$ to show that $\text{poss}(t|\pi')$ also has at least two (neighbouring) blocks for each $t : i+1 \leq t \leq j-1$. ∎

The previous results are useful to any proxy to decide when and where to look for potential privacy violations based on locations. The next theorem further narrows down potentially vulnerable movements. It uses the idea of a shortest path between two blocks. We also need to introduce another concept called the *span* of a path to easily state our results.

***Definition** 9 (Distance):* The *span* of a path p is $|p| - 1$ [5].

The *distance* between two blocks a and b is the span of the shortest path from a to b:

$$\delta(a, b) = \min\{|p| - 1 \mid p \text{ is a path from } a \text{ to } b\}$$

***Lemma** 8 (Triangle Inequality):* As expected, the above notion of shortest path follows triangle inequality. For any blocks a, b, c, we have $\delta(a, b) \leq \delta(a, c) + \delta(c, b)$.

Note that, distance is symmetric and shortest path may not be unique. Also, the above property implies that, for neighbouring blocks a, b, $\delta(a, c) \leq \delta(b, c) + 1$ for any block c. Furthermore, observe that it is always possible to construct a path of span $l \geq \delta(a, b)$ between any two blocks a and b [6]. We now state the theorem which essentially says that movements which do not follow shortest paths are not vulnerable to privacy attacks.

***Theorem** 3 (Shortest Path):* Consider a path $p = p' \cdot a \cdot b_1 b_2 \dots b_k \cdot c_1 \cdot p''$ of length t (where $k > 1$), where the blocks a and b_1 are on a border, i.e., $G(a \mid p') \neq G(b_1 \mid p' \cdot a)$ and the next border crossing is when moving from the block b_k to c_1.

For any composable G, if b_1 is disclosed first at time t, then $b_1 \dots b_k$ is a shortest path from b_1 to b_k and, therefore, $\delta(b_1, b_k) = k - 1$.

Proof: Consider any neighbour b'_1 of both a and b_1 such that $p' \cdot a \cdot b'_1$ is consistent with $G(p' \cdot a \cdot b_1)$. Note that such a neighbour always exists, because otherwise b_1 would be disclosed right when the user moved from a to b_1 – but this is not possible according to the conditions of the lemma.

Let us assume, for the sake of contradiction, that the shortest path from b_1 to b_k is of length $l \leq k - 1$. In that case, it is possible to construct a path $b_1 b'_2 \cdots b_k$ from b_1 to b_k of span $k - 1$. Now consider the path $p^* = p' \cdot a \cdot b'_1 \cdot b_1 b'_2 \cdots b_k \cdot c_1 \cdot p''$. Since both p and p^* are consistent with $G(p)$, b_1 cannot be disclosed at time t, leading to a contradiction. ∎

[5] Essentially, this is the number of edges or corners a user following the path has to cross.

[6] This can be easily proved by induction on l.

IV. Case Study

Most research that uses blurring for location obfuscation is not very precise about the shape/size of the blur region. In this section, we show that shape and size of the basic blocks and blurred cells vastly differ in terms of properties. We apply our notion of privacy to block based location obfuscation mechanisms varying in shapes and sizes and compare their privacy properties, both analytically and experimentally. Obviously this study is by no means exhaustive – we only consider a few representatives; our goal is to show the kind of issues that should be considered while deciding a scheme.

We consider four disjoint-tiling based mechanisms – one with square tiles, two with triangular tiles and one other with hexagonal tiles, and, two overlapping-tiling based mechanisms – one each with square and triangular tiles.

A. Disjoint Tilings

Disjoint tiling forms the basis of most common forms of obfuscation used in location privacy. In this, each block is contained in exactly one cell, and is encoded simply as its containing cell. This is in contrast to the overlapping tilings in which encoding depend on previous locations. Computation of current possibilities is also simplified by a tight version of 4.

***Lemma** 9:* Disjoint-tiling based encodings are independent and are therefore, composable.

***Lemma** 10 (Lemma 4 for Disjoint Tiling):* Let π denote a cell-path of length t and π_{t+1} be the next reported cell. Then,
$\text{poss}(t+1|\pi \cdot \pi_{t+1}) = \pi_{t+1} \cap \Gamma(\text{poss}(t|\pi))$.

We consider three different types of disjoint tiling for further analysis.

- Figure 1 shows an example of 4×4 Disjoint Square tiling - blocks and cells are both squares. There, the path followed by the User-A is $b_{6,0} \cdot b_{7,1} \cdot b_{8,2} \cdot b_{9,3} \cdot b_{10,4} \cdot b_{11,5} \cdot b_{12,6}$ which will be encoded to the cell-path $C_{10} \cdot C_{10} \cdot C_{20} \cdot C_{20} \cdot C_{21} \cdot C_{21} \cdot C_{31}$.
- Figure 4 shows an example of Disjoint Triangle (Eq) tiling where blocks and cells are equilateral triangles. Similar to the previous example, User U2's path $b_{3,5} \cdot b_{3,6} \cdot b_{4,9} \cdot b_{4,10}$ will be encoded to the cell-path $C_{00} \cdot C_{00} \cdot C_{13} \cdot C_{13}$.
- Figure 6 is an example of Disjoint Hexagon tiling where hexagonal cells are composed of equilateral triangle blocks. The path of user U1 $b_{1,6} \cdot b_{2,7} \cdot b_{3,5} \cdot b_{4,4} \cdot b_{5,7}$ has $C_{00}, C_{11}, C_{11}, C_{10}, C_{11}$ cell path as its encoding.

Encodings based on the above tilings differ in the shape of the blocks and/or in the shape of the cells and basically effects the computation of possible locations and neighbourhood of a block. well as variation in possibilities for the encoding. We compare them for the prevalence of disclosures in IV-F. For the purpose of comparison only, we were able to construct a Disjoint Triangle (Rt) tiling, in which the blocks are right-angled triangles and the cells are equilateral triangles. The unique structure and high degree of neighbourhood drastically lowers the number of disclosures for it. It is illustrated in Figure 5, where e.g., U1's path $b_{1,2} \cdot b_{2,9} \cdot b_{1,9} \cdot b_{1,14}$ will be encoded to the cell-path $C_{00} \cdot C_{13} \cdot C_{02} \cdot C_{04}$.

B. Analysis of Disjoint Tilings: Square, Triangle (Eq), Hexagon

The results in section III tell us that privacy violations can occur only for blocks at the border and happens only when crossing a border. However, for long paths the number of border crossings can be quite large [7], which makes detecting violations a compute intensive

[7]In the worst-case, $O(|p|)$, for a path p.

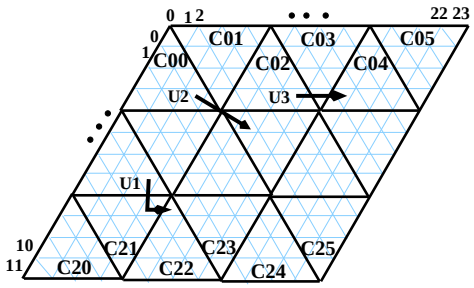

Fig. 4. Disjoint Triangle (Equilateral) Encoding

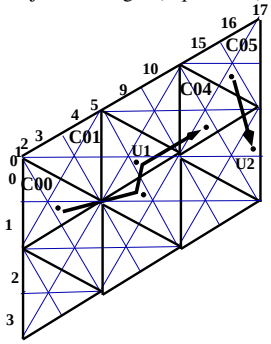

Fig. 5. Disjoint Triangle (Right Angle) Encoding

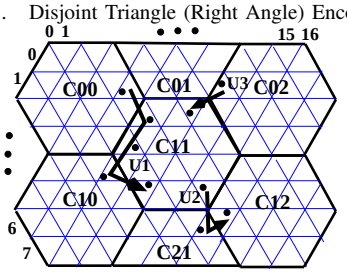

Fig. 6. Disjoint Hexagon Encoding

effort. In this section, we will extend Theorem 3 to give an analytic upper bound on the number of past positions the proxy should store to detect every privacy violation. Some of the results hold for *any* disjoint tiling, and the rest are proved only for the three cases we are explicitly interested in, namely, the Disjoint Square, the Disjoint Hexagon and the Disjoint Triangle (Eq).

***Definition** 10 (Stretch):* Consider any disjoint tiling. The stretch of a cell π, denoted by $\sigma(\pi)$, is defined by:

$$\sigma(\pi) = \max\{1 + \delta(a, b) \ : \ \text{any blocks } a, b \in \pi\}$$

Stretch is basically the maximum number of blocks on the shortest possible path from one border block to another border block in the same cell. The mechanisms we consider are based on regular disjoint tiling, so, it is same for all the cells and will be denoted by σ. For tilings using non-regular polygons, σ will depend on the cell(s) used to compute privacy (an upper bound could be obtained in this case by taking the maximum possible stretch for all cells in the tiling).

We will next prove the main result of this section.

***Theorem** 4:* $1 + \sigma$ past locations (including current) is enough to detect all privacy violations for disjoint grid, disjoint triangle (Eq) and disjoint hexagon encodings.

The number of past locations necessary is now essentially a *constant* (depending only on the structure of the cells which is fixed

a priori in our architecture). This is a vast improvement compared to the naïve algorithm to compute privacy violations. For illustrations, the number of locations needed is 5 for the examples in Figures 1, 4 and 6.

Corollary 1: The number of past locations (including current) necessary to detect *all* privacy violations are,

- For Disjoint Square encoding, $1 + l$.
- For Disjoint Triangle (Eq) encoding, $1 + \lceil \frac{l}{2} \rceil$
- For Disjoint Hexagon encoding, $1 + 2\lceil \frac{l}{2} \rceil$

where, l denotes the number of blocks along a side of the corresponding cell.

We suspect the theorem to be true for *most* natural tilings; however, in this paper we only show it to hold for tilings which satisfy the following property. It basically says that if any block is far from some cell on the boundary, then it is far from all cells on that boundary.

Property 7 (Border Distance): A tiling satisfies the *border distance* property if the following holds for any cell π:
Consider any two blocks b_1, b_k on any two different boundaries of π. Take any block b'_1 which is a neighbour of b_1. Then, it holds that if $\delta(b'_1, b_k) \geq \delta(b_1, b_k) + 1$, then for all b'_k on the same boundary as b_k, $\delta(b'_1, b'_k) \geq \delta(b_1, b_k) + 1$.

This property holds for all tilings in which the blocks are arranged in layers, and these layers run parallel to the boundaries of the cells. It is easy to verify that it holds for all the three tilings considered in this section.

Proof sketch of Theorem 4: Consider any path $p = p' \cdot a \cdot b_1 b_2 \cdots b_k \cdot c_1$ of length t where $G(a) \neq G(b_1) = \cdots = G(b_k) \neq G(c_1)$ ((a, b_1) is a border crossing and the next border crossing happens at (b_k, c_1)). Then of course, $\delta(b_1, b_k) \leq k - 1$. Denote the cell $G(c_1)$ by γ and $G(b_1)$ by β and the cell-path $G(p)$ by π. Denote the time when the user was at b_1 by s. There can be three possible cases for the second crossing (b_k, c_1).

1) It is a corner crossing – the user crosses via a corner of a block.
2) It an edge crossing – the user crosses via a side of a block and b_1 and b_k are on different border.
3) It an edge crossing and b_1 and b_k are on the same border.

We will prove the above theorem by considering all these possible cases.

***Lemma** 11 (Corner Crossing):* If (b_k, c_1) is a corner crossing, then either b_1 is disclosed at or before t or never disclosed ever after.

Proof: Assume that b_1 is not disclosed at or before time t.

Then, at time t, there must be some path q which is consistent with π, but $q_s \neq b_1$. q_{t-1} is the corner block in q just before crossing to γ. Since G is composable, consider the path $p^* = q_1 \cdots q_{t-1} c_1$; p^* is a valid path since both q_{t-1} and c_1 are corner blocks meeting at the common corner of their corresponding cells. Also, p^* consistent with π by construction. Therefore, for any time $t' > t$, there will always be a consistent path (whose first t blocks will be $q_1 \cdots q_s c_1$) whose s-th block is not b_1 and therefore, b_1 will not be disclosed. ■

***Lemma** 12 (Different Border Crossing):* If b_1 and b_k are on different borders, then either b_1 is disclosed at or before t or never disclosed ever after.

Proof: Assume, there exists some $b'_1 \in \beta$ such that b'_1 is a neighbour of both b_1 and a and also $\delta(b'_1, b_k) \leq \delta(b_1, b_k) \leq k - 1$. Therefore, we can construct a path q' from b'_1 to b_k of span $k - 1$ (length k). Now take the path $q = p' \cdot a \cdot q' \cdot c_1$ – q is a valid path and is consistent with π, but $q_s \neq b_1$. Therefore, for any time $t' > t$, there will always be a consistent path (whose first t blocks will be q) whose s-th block is not b_1 and therefore, b_1 will not be disclosed.

Now consider the opposite case: for all $b'_1 \in \beta \setminus \{b_1\}$, if $\Gamma(a, b'_1)$, $\Gamma(b_1, b'_1)$, then $\delta(b'_1, b_k) = \delta(b_1, b_k) + 1$. Since the tiling has border distance property, this implies that for all b^* on the same border as b_k, $\delta(b'_1, b^*) \geq \delta(b_1, b_k) + 1 = k$. In other words, there cannot be any path from b'_1 to any cell in γ. So, $b'_1 \notin \text{poss}(t|\pi)$. This holds for all $b'_1 \in \beta \setminus \{b_1\}$. Therefore, b_1 is disclosed at time t. ■

***Lemma** 13:* (Same Border Crossing) Consider the case where b_1 and b_k are on the same border and none of them are disclosed at time $t - 1$. Then, if b_k is not disclosed at time $t' \geq t$, then b_1 is also not disclosed at t'.

Proof: Assume, for the sake of contradiction, that b_1 is disclosed first at some time $t' \geq t$ but b_k is still not disclosed at t'. Denote the cell-path at time t' by ψ. Theorem 3 immediately implies that $\delta(b_1, b_k) = k - 1$. Since b_k is not disclosed at t', therefore $poss(t - 1|\psi)$ consists of blocks other than b_k – denote this set of blocks by $B = poss(t - 1|\psi) \setminus \{b_k\}$. Therefore, by Triangle Inequality, $\forall b'_k \in B,\ k - 2 \leq \delta(b_1, b'_k) \leq k$. However, there cannot be $b'_k \in B$ such that $\delta(b_1, b'_k) = k$. This is because, b_1 is disclosed at t' and there cannot be any consistent path of span k from b_1 to b'_k. Therefore, $\delta(b_1, b'_k) \in \{k - 2, k - 1\}$.

Now, if there exists any b'_k such that $\delta(b_1, b'_k) = k-2$, then we will construct a path q consistent with ψ such that $q_s \neq b_1$, and therefore, b_1 cannot be disclosed at t'. Since b_1 is not revealed at s, therefore there exists some block b'_1 which is neighbour to both a and b_1. Using the Triangle Inequality, $\delta(b'_1, b'_k) \leq \delta(b'_1, b_1) + \delta(b_1, b'_k) = k - 1$. Then, we can construct a path of span $k - 1$ (length k) from b'_1 to b'_k – denote it by q'.

Since b'_k is a possible block at time $t - 1$, there is some path consistent with ψ whose $t - 1$-th block is b'_k; represent this path by $r' \cdot b'_k \cdot r''$ (here b'_k is the $t - 1$-th block). Now, q can be constructed by $p' \cdot a \cdot q' \cdot r'$ since G is composable (it is easy to verify that $q_s = b'_1 \neq b_1$).

The only case now left is – for all $b'_k \in B$, $\delta(b_1, b'_k) = k - 1$. The proof for this case depends upon the shape of the blocks used in the encoding.

Square Blocks: When the blocks are squares, $\delta(b_1, b'_k) \neq k - 1$ for any $b'_k \in B$ since there is *exactly* one block on a boundary at some fixed distance from another block at the same boundary.

Triangular Blocks: The analysis is significantly more complicated when the blocks are equilateral triangles and requires a case analysis over the different orientations of b_1 and b_k. Without loss of generality, let us assume that we are approaching b_1 to b_k left from right.

1) b_1 and b_k both are upward triangles as shown in the Figure 7-1: Consider the block b'_k, a downward triangle and left-neighbour of b_k along the same border; It has $\delta(b_1, b'_k) = k - 1$. Observe that the neighbours of b_k in γ are subset of neighbours of b'_k in γ implying $b'_k \in B$. It is possible that there is no $b^* \in B$ such that $\delta(b_1, b_*) < k - 1$ (for example, if left most c_1 is not in $\text{poss}(t|\psi)$ as shown in Figure 7-1). In this case all the neighbours b'_1 of b_1 along the same border will be neighbours of a and will have $\delta(b'_1, b'_k) \leq k - 1$. Further, the block b'_1 at the right of b_1 will always be in $\text{poss}(s|\psi)$.
2) b_1 is upward triangle and b_k is downward triangle as shown in the Figure 7-2: Consider the two blocks b'_k which are upward triangles and are neighbours to b_k with $\delta(b_1, b'_k) \leq k - 1$. For the left-neighbour b'_k of b_k we will have $\delta(b_1, b'_k) = k - 2$. If it is in B then b_1 is not disclosed (discussed earlier). If it is not in B, the right-neighbour b'_k will have $\delta(b_1, b'_k) = k - 1$. The right-neighbour b'_1 of b_1 will have $\delta(b'_1, b'_k) = k - 1$ and will also be a neighbour of a. Therefore b'_1 should be in $\text{poss}(s|\psi)$.

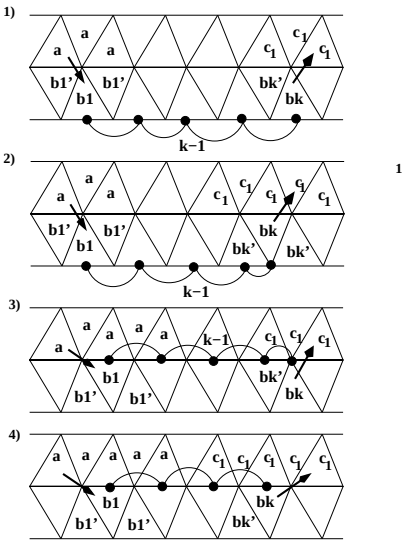

Fig. 7. Proof of Lemma 13 for triangular blocks

3) b_1 is downward triangle and b_k is upward triangle as shown in the Figure 7-3: Consider the block b'_k which is a downward triangle and is left-neighbour of b_k. It must be a block on the shortest path from b_1 to b_k. Therefore $\delta(b_1, b'_k) = k-2$, and also $b'_k \in B$ (since the neighbours of b_k in γ are subset of neighbours of b'_k in γ). We have already taken care of this case.
4) b_1 and b_k are both downward triangles as in Figure 7-4: In this case for block to the right of b_k will have distance greater than $k-1$. Therefore it can not be in the possibility set of b_k (even before). The left-neighbour of b_k, i.e., b'_k must be in B (as b_k is not disclosed) with $\delta(b_1, b'_k) = k-1$. Now for both the neighbours of b_1 ,i.e., b'_1 will have $\delta(b'_1, b'_k) \leq k-1$. At least one of these b'_1 has a as its neighbour (otherwise b_1 must have disclosed before). Hence b_1 can not be disclosed at time t'.

■

The last three lemmas essentially say that after crossing a border into block b_1, either b_1 never gets disclosed ever after or one of the following two cases will always happen

1) If the next border crossed is a different one, b_1 is disclosed right after crossing it (at c_1). Since in that case, $b_1 \ldots b_k$ is the shortest path, this implies an upper bound of $1+\sigma$ number of past locations necessary to detect the disclosure.
2) If the next border crossed is the same one, b_1 cannot be disclosed if b_k is also not disclosed. Therefore, it suffices to store the locations until and including the next border crossing via the same boundary. This again gives an upper bound of $1+\sigma$ by a similar argument as above.

This finishes the proof of Theorem 4. ■

C. Computing Privacy Violations for Disjoint Tilings

We can use Theorem 4 and other results from Section III-C to construct an efficient algorithm (to be used by the proxy) for detecting privacy violations for the three disjoint encodings. The central idea of Algorithm 1 given below are the facts that if any move violates weak minimal privacy, then the last disclosure must happen at or after the previous boundary crossing and that disclosure happens only when some border is crossed. It calls these external functions which we omit, their implementation depends on the actual encoding of the disjoint tiling: $G()$ to denote the encoding function, $\mathrm{poss}()$ to denote a function to compute the current possibility and $\delta()$ to denote a function to compute the minimum distance between two blocks.

Algorithm 1 Algorithm for detecting privacy disclosure

Input: u: User-Id
b: Current block of u
t: Current time
Output: Safe / Unsafe (weak minimal privacy not preserved)
State: $c(u)$: Last reported cell of u
$B(u)$: Last boundary block of u
$m(u)$: Number of moves after $B(u)$
$p(u)$: Possibility of u for $B(u)$
$P(u) = [P_i, P_{i-1} \ldots]$: History of past possibilities of u
Method:

1: $status \leftarrow$ Safe
2: $c \leftarrow G(b)$
3: $m(u) \leftarrow m(u)+1$
4: $P_t \leftarrow \mathrm{poss}(b, P_{t-1})$ {// compute current possibilities}
5: **if** $B(u)$ is null and b is a boundary block: **then**
6: $B(u) \leftarrow b$
7: $P(u) \leftarrow [P_t]$
8: **else**
9: **if** $c(u) \neq c$: **then** {// u crossed a boundary}
10: **if** $m(u) = \delta(b, B(u))$: **then** {// u on shortest path}
11: **Update** $p(u)$ & each $P_i \in P(u)$ using Update Lemma
12: **if** $p(u)$ or any $P_i \in P(u)$ has one block: **then**
13: $status \leftarrow$ Unsafe {// disclosure found}
14: $B(u) \leftarrow b$
15: $P(u) \leftarrow [P_t]$
16: **else** {// u is in same cell}
17: **if** $m(u) \leq \delta(B(u), b)$: **then**
18: **Append** P_t to $P(u)$
19: **else** {// not along shortest path}
20: $B(u) \leftarrow$ null
21: $P(u) \leftarrow [P_t]$
22: $c(u) \leftarrow c$
23: **return** $status$

The length of $P(u)$ is always going to be upper bounded by σ. The possibility of each P_i is at most the number of blocks in a cell, which is $O(\sigma^2)$. Therefore, the running time is $O(\sigma^3)$ while the amount of extra space necessary for storing global state is $O(\sigma)$. Since σ is fixed for a specific encoding, this essentially gives us an algorithm running in constant time/space.

Our experiments showed that the upper bound of σ holds true for other disjoint encodings as well; unfortunately, there is no reasonably small bound on the length of past history for overlapping encodings. A direct adaptation of this algorithm for general encodings (disjoint or overlapping) without any known bound on the length of past history may not be very efficient since it has to store all previous locations. But even then, it is possible to perform better than straight-forward checking of all past locations by using the facts about boundary crossing.

D. Overlapping Encoding: 2 cases

The encoding we used for overlapping tiling can be described by overlaying two different disjoint tilings of cells (say, tiling A and B)

on one arrangement of blocks such that each cell of tiling A partially overlaps some cell(s) of tiling B and vice versa. This makes each block belong to two cells, one each from A and B. The first block in a path is encoded as the cell from A containing it, and a global state C (for *current-tiling*) set to A. On any subsequent move, it is checked against the cell containing the current block according to tiling C. If this move was inside the cell, this cell is reported as the encoding. If it crossed a boundary, C is set to the other tiling and then the cell according to the new C is reported. The intention is to ensure that with respect to its encoded cell, a block is never at the boundary. Interestingly, we found that the very switching of tilings leak enough information to cause embarrassing disclosures.

For these overlapping encodings, the cell reported for the current location depends on the last location and the last reported cell but nothing earlier. Computation of new possibility is accordingly modified.

***Lemma** 14:* Overlapping Square and Triangle (Eq) encodings are local, and, therefore composable.

***Lemma** 15 (Lemma 4 for Overlapping Tiling):* Let π denote a cell-path of length t and π_{t+1} be the next reported cell. Then,
$\text{poss}(t+1|\pi \cdot \pi_{t+1}) = \pi_{t+1} \cap \Gamma(\text{poss}(t|\pi)) \setminus \pi_t$.

The two overlapping encodings we evaluated are the overlapping versions of their disjoint versions.

- Figure 2 is an example of 4×4 Overlap Square tiling. One square tiling is shown by dotted lines and the other by solid lines. Consider the path followed by user User-A i.e., $b_{6,0} \cdot b_{7,1} \cdot b_{8,2} \cdot b_{9,3} \cdot b_{10,4} \cdot b_{11,5} \cdot b_{12,6}$. The cell encoding for this is $C_{10} \cdot C_{10} \cdot D_{21} \cdot D_{21} \cdot C_{21} \cdot C_{21} \cdot D_{32}$. For the first move, the tiling with dotted lines is chosen as the active tiling, which results into reporting of cell C_{10}. Next move of the user $b_{7,1}$ does not cross any boundary of active cell C_{10} and hence the same cell is reported. The move $b_{8,2}$ then crosses the boundary of this cell, and hence, the other tiling shown with solid lines is used to get the encoding D_{21}.
- Figure 8 shows an example of Overlap Triangle encoding. Similar to the previous example, dotted and solid lines are used to indicate the two tilings. Path of user U1 $b_{3,5}, b_{2,6}, b_{3,9}, b_{4,10}, b_{4,12}, b_{4,13}$ is similarly encoded to the cell-path $C_{00}, D_{02}, D_{02}, C_{01}, D_{13}, D_{13}$.

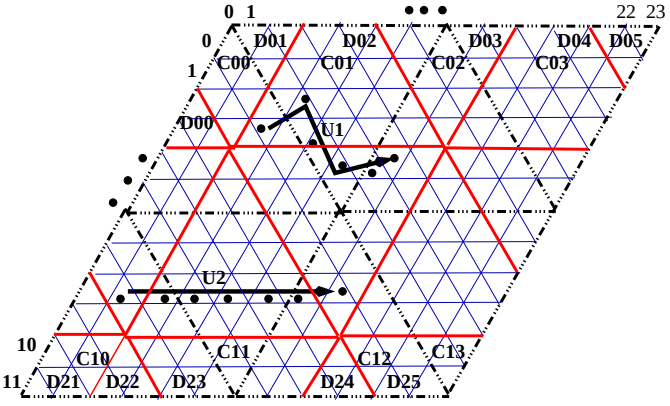

Fig. 8. Overlap Triangle (Equilateral) Encoding

E. Privacy Leaks

Here we discuss and classify the privacy leaks that we observed during our analysis. They are classified based on, loosely, ease of detecting. Specifically, detecting some leaks require more resource (in terms of both computing time and storage of past information) than others. This kind of classification may come handy for users trying to compare mechanisms based on their privacy-cost implications.

D-Leak: D-leak happens due to the current move and the last move where both the moves are in different cells (usually, diagonally opposite cells) and which discloses one of the two locations. As an example, in Figure 1, right after the third move, the locations at the second and the third moves become evident (block (11,11) at t_2 and (12,12) at t_3).

L-Leak: L-leak happens due to the current move and the last two moves where each of the three is in a different cell and which discloses the last location. As an example, in Figure 1, the last three moves of User-D result into an L attack and the attacker will know that the User-D was at block (3,11) at time t_2.

Long-Leak: Any other type of leak is classified as a Long-leak. It normally requires larger number of moves. As an example, in Figure 1, the movement of user-A results into a long attack – after the t-th move, it is possible to determine the location at time t_4 (block (9,3)) and t_5 (block (10,4)).

D-leaks and L-leaks are easier to detect, they happen due to the last few steps and easier to conceptualise. However, detecting long-leaks requires more effort and is not immediately obvious. Encodings based on triangular blocks have no L-leaks by design. Also, the construction of our overlapping encodings ensures absence of D-leaks. Lastly, the Disjoint Triangle (Rt) encoding was constructed to avoid both D-leaks and L-leaks. We evaluate the encodings in terms of these privacy leaks and present a summary of the result in Section IV-F.

F. Comparison

Here we present the result of our experiments where we compared tested the above mechanisms with respect to the types and number of privacy violations. Square obfuscation region being a common choice in location-based services, we tested both its disjoint and the overlapping versions for different cell sizes – 4, 6 and 8 blocks a side. We also tested the triangle encodings and the hexagon encoding with cells of comparable size and stretch. We generated 2000 paths (of random block length) using the Random Waypoint mobility model and checked for privacy violations for movements along those paths. The results are shown in Table I.

Encoding	**Num blocks per side (stretch)**	**% Paths with *L*-Leak**	**% Paths with *D*-Leak**	**% Paths with Long-Leak**	**Total**
4×4 Disjoint Square	4 (4)	40.05	20.45	18.8	79.3
4×4 Overlap Square	4 (4)	29.35	0	1.2	30.55
6×6 Disjoint Square	6 (6)	25.75	13.45	30.55	69.75
6×6 Overlap Square	6 (6)	24.35	0	1.75	26.1
8×8 Disjoint Square	8 (8)	23.25	10.4	35	68.65
8×8 Overlap Square	8 (8)	16.35	0	2.7	19.05
Disjoint Triangle (Eq)	11 (6)	0	25.45	1.1	26.55
Overlap Triangle (Eq)	11 (6)	0	0	13.3	13.3
Disjoint Hexagon	5 (4)	0	0	24.75	24.75
Disjoint Triangle (Rt)	4 (2)	0	0	2.9	2.9

TABLE I
COMPARISON OF DIFFERENT ENCODINGS

We can make a few observations from the comparative study. It seems that the Disjoint Triangle (Rt) encoding suffers from the least number of privacy violations despite having small size. Note that,

smaller size means more blocks on the border and increases the front for violations. Square encodings seem to be the worst among all with lot of leaks of all possible types. Overlapping encodings seem to perform better than their disjoint counterparts, justifying the use of strategies to hide boundary crossings.

From another perspective, it is much easier to detect D and L-leaks than long leaks; so, if any user is interested only in keeping his current location (or last few locations) private, then encodings with fewer D and L-leaks but many long-leaks may be a good choice. Those based on triangle and hexagon are very much suitable for this purpose.

G. Measure of Privacy

Here we make an attempt to define a level of privacy that is well-defined and captures its essential requirements. We propose to use *stretch* to define levels of privacy. For example, a lower level of privacy would mean a tiling with small stretch. Not only it is intuitive and simple to understand, but it also has the following useful properties which capture the cost-quality trade-offs:

- The area of a cell depends on the stretch. So, increasing it will lead to lowering of the quality of query results.
- We have proved that cells with larger stretch increase the amount of computation and extra storage necessary to detect privacy violations.
- Since privacy violations happen only at borders and due to border crossing, intuitively, the number of privacy violations should decrease for larger cells. As discussed in Section IV-F, this fact is supported by our experiments.

V. Related Work

Enforcing privacy of personal information in public databases or remote untrusted services (which are similar to public databases with respect to privacy concerns) has been in focus since 1990's [23, 29, 7], and since then a lot of work has been going on to address various aspects of it. Some proposed approaches use user specified *privacy policies* to restrict the *use of collected information* [25], e.g., anonymisation of identities *before disclosure of information* [27] for mobile cellular systems. The limitation of policy based solution is that users have to trust the service provider for correct execution of their policies. Since users practically lose control over the use of their information once the information is disclosed, it is desirable to have more control over what information is published/collected to prevent any misuse of private information.

The issue of privacy in location-based services is very similar to the above and in fact, research in privacy for location-based services adopt most of the concepts from the area of privacy in database publishing. User queries can contain sensitive information which may be misused by a service provider. Furthermore, information revealed in different queries can be correlated to draw vast number of surprising inferences. There are several techniques that have been proposed to control what information gets published (sent to the service). Identity anonymisation approach maps the true identity of a user to a pseudonym but does not anonymise the location of a user; special care needs to be taken so that the provider is unable to correlate multiple queries e.g. Beresford and Stajano solve this problem by proposing a concept called mix zones [3]. Location anonymisation is another approach that tries to preserve the location privacy by anonymising the exact location information before querying the service [10]. Two concepts borrowed from the database research are very popular with respect to anonymisation: k-anonymity and l-diversity. In k-anonymity the anonymisation is done in a way to ensure that no adversary can narrow down any particular user beyond a group of k possible users [27]. An improvement on it, l-diversity states that there should be at least l different type of sensitive values within a group to make difficult for the adversary to associate a sensitive value with a specific user [19]. However, the solutions using k-anonymity and l-diversity in general decrease the granularity of actual location information to a bigger region called cloaked area, which meets k-anonymity and l-diversity constraints, but reduces quality.

Chow et al. categorised these techniques into three broad categories [10]:

1) *False Location:* A set of locations including user's exact location and other false locations is reported to the server in a query [17].
2) *Spatial Cloaking:* A user's exact location is blurred into a cloaked spatial area according to the user's specified privacy preferences [1, 11, 15, 16].
3) *Cryptographic techniques:* This technique uses cryptographic *private information retrieval* techniques [13] to encode location information along with the query.

All these privacy preserving techniques are implemented in one of the two following architectures.

1) Client Server Architecture: Discussed extensively in [1, 5, 12, 22], in this architecture, a trusted *location anonymisation server* anonymises different components of each user query before submitting the query.
2) Peer to Peer Architecture: Introduced in [11], here a mobile device itself computes its cloaked area with the help of its nearby peers [14]. Two types of systems have been discussed within this setting: a) proactive, and b) lazy. In proactive system, each mobile device consistently keeps current location information of each neighbour device with it all the time. Whereas in lazy system, a mobile device first figures out its peers then computes it cloaked area.

VI. Conclusion

We investigated in this paper the question of appropriately choosing obfuscation regions for location obfuscation in the context of continuous queries. We showed that the quality of privacy preservation differs widely among different shapes and sizes. We also considered strategies to prevent information leakage and, using the model of overlapping obfuscation regions, showed that some of them can significantly improve privacy properties. We constructed privacy detection constant time/space algorithms for some strategies; such algorithms are therefore suitable for implementation on mobile devices to provide location obfuscation. Finally, we proposed how to define levels of privacy suitable for users of location-based services.

There exists many possible directions arising from our work. We conclude this paper by discussing some of them.

We defined overlapping obfuscation regions with the goal of understanding smart strategies to deal with information leakage due to crossing of boundaries. It would be interesting to analyse such strategies (e.g., mix zones) and do a comparative analysis of their privacy properties with respect to simpler strategies (e.g., disjoint square cells).

Our analysis was based on worst-case scenario in two sense, definition of privacy and definition of neighbourhood. The worst-case definition of privacy in terms of weak minimal privacy is to too strict for an adversary; even if $\mathcal{R}$ is able to predict a location with 99% probability, our definition would not consider it a privacy leak.

A useful future work would be to extend this worst-case analysis to average-case analysis in which a prediction would be considered a disclosure if its success probability is more than some constant (e.g., more than 0.5).

For neighbourhood too, we allow the user to move to any adjacent block which shares a vertex or an edge. In essence, this worst-case behaviour makes it harder for $\mathcal{R}$ to predict exact locations. Under usual scenario, users can only move along already existing roads; this information is common knowledge and could significantly reduce the possibilities at any time. One would require smarter obfuscation strategies in the presence of the knowledge of road network.

A few other questions we leave open are how to extend our two-level hierarchy (cell, block) to multiple levels of obfuscation and how to use the ideas in this paper to give proper privacy definitions for other sensitive parameters, e.g., query obfuscation.

REFERENCES

[1] B. Bamba, L. Liu, P. Pesti, and T. Wang. Supporting anonymous location queries in mobile environments with privacygrid. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 237–246, 2008.

[2] L. Barkhuus and A. Dey. Location-Based Services for Mobile Telephony: a Study of Users' Privacy Concerns. In *INTERACT 2003: 9th IFIP TC13 International Conference on Human-Computer Interaction*, pages 709–712, 2003.

[3] A. R. Beresford and F. Stajano. Location Privacy in Pervasive Computing. *IEEE Pervasive Computing*, 2(1):46–55, 2003.

[4] A. R. Beresford and F. Stajano. Mix zones: User privacy in location-aware services. *PERCOMW04: 2nd IEEE International Conference on Pervasive Computing and Communications*, 0:127–131, 2004.

[5] C. Bettini, S. Mascetti, X. S. Wang, and S. Jajodia. Anonymity in Location-Based Services: Towards a General Framework. In *MDM '07: Proceedings of the International Conference on Mobile Data Management*, pages 69–76, 2007.

[6] A. Burak and T. Sharon. Usage patterns of FriendZone: mobile location-based community services. In *MUM '04: Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia*, pages 93–100, 2004.

[7] M. A. Caplinger, M. L. Campbell, and M. A. Capstick. Cover story: they know where you are. *IEEE Spectrum*, 40:20–25, July 2003.

[8] cellular news. Mobile Subscribers to Hit 5.9 Billion in 2013, Driven by China, India, Africa. http://www.cellular-news.com/story/40439.php?s=h.

[9] C.-Y. Chow and M. F. Mokbel. Privacy in location-based services: a system architecture perspective. *SIGSPATIAL Special*, 1(2):23–27, 2009.

[10] C.-Y. Chow, M. F. Mokbel, and T. He. Tinycasper: a privacy-preserving aggregate location monitoring system in wireless sensor networks. In *SIGMOD '08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1307–1310, 2008.

[11] C.-Y. Chow, M. F. Mokbel, and X. Liu. A peer-to-peer spatial cloaking algorithm for anonymous location-based service. In *GIS '06: Proceedings of the 14th annual ACM international symposium on Advances in geographic information systems*, pages 171–178, 2006.

[12] B. Gedik and L. Liu. Protecting Location Privacy with Personalized k-Anonymity: Architecture and Algorithms. *IEEE TMC: IEEE Transactions on Mobile Computing*, 7:1–18, 2008.

[13] G. Ghinita, P. Kalnis, A. Khoshgozaran, C. Shahabi, and K.-L. Tan. Private queries in location based services: Anonymizers are not necessary. In *SIGMOD '08: Proceedings of the 2008 SIGMOD International conference on Management of data*, pages 121–132, 2008.

[14] G. Ghinita, P. Kalnis, and S. Skiadopoulos. MobiHide: A Mobile Peer-to-Peer System for Anonymous Location-Based Queries. In *Proceedings of the 10th international symposium on Advances in Spatial and Temporal Databases*, pages 221–238, 2007.

[15] 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, 2003.

[16] P. Kalnis, G. Ghinita, K. Mouratidis, and D. Papadias. Preventing Location-Based Identity Inference in Anonymous Spatial Queries. *IEEE TKDE: IEEE Transaction on Knowledge and Data Engineering*, 19(12):1719–1733, 2007.

[17] H. Kido, Y. Yanagisawa, and T. Satoh. An anonymous communication technique using dummies for location-based services. *International Conference on Pervasive Services*, 0:88–97, 2005.

[18] H. Li. Important properties of planar normal tiling. In *Proceedings of the 3rd WSEAS international conference on Circuits, systems, signal and telecommunications*, pages 140–144. World Scientific and Engineering Academy and Society (WSEAS), 2009.

[19] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. L-diversity: Privacy beyond k-anonymity. *ACM TKDD: ACM Transaction on Knowledge Discovery in Databases*, 1(1), 2007.

[20] D. J. Martin, D. Kifer, A. Machanavajjhala, J. Gehrke, and J. Y. Halpern. Worst-Case Background Knowledge for Privacy-Preserving Data Publishing. In *ICDE07:23rd International Conference on Data Engineering*, pages 126–135, 2007.

[21] G. Miklau and D. Suciu. A formal analysis of information disclosure in data exchange. *Journal of Computer and System Sciences*, 73(3):507–534, 2007.

[22] M. F. Mokbel, C.-Y. Chow, and W. G. Aref. The new Casper: query processing for location services without compromising privacy. In *VLDB '06: Proceedings of the 32nd international conference on Very large data bases*, pages 763–774, 2006.

[23] P.A.Karger and Y. Frankel. Security and Privacy Threats to ITS. *Proceedings of the Second World Congress on Intelligent Transport Systems*, 5:2452–2458, 1995.

[24] V. Rastogi, M. Hay, G. Miklau, and D. Suciu. Relationship privacy: output perturbation for queries with joins. In *PODS '09: Proceedings of the twenty-eighth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 107–116, 2009.

[25] E. Snekkenes. Concepts for personal location privacy policies. In *EC01: Proceedings of the 3rd ACM conference on Electronic Commerce*, pages 48–57, 2001.

[26] L. Srivastava. Mobile phones and the evolution of social behaviour. *Behaviour and Information Technology*, 24(2):111+, 2005.

[27] L. Sweeney. Achieving k-Anonymity Privacy Protection using Generalization and Suppression. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5):571–588, 2002.

[28] J. Voelcker. Stalked by satellite: An alarming rise in GPS-enabled harassment. *IEEE Spectrum*, 7(47):15–16, 2006.

[29] R. Want, A. Hopper, V. F. ao, and J. Gibbons. The active badge location system. *TOIS: ACM Transaction on Information System*, 10:91–102, January 1992.

[30] Tiling by regular polygons. http://en.wikipedia.org/wiki/Tiling_by_regular_polygons.