An Online Framework for Publishing Privacy-Sensitive Location Traces

Wen Jin*, Kristen LeFevre*, Jignesh M Patel†

*Electrical Engineering & Computer Science, University of Michigan, 2260 Hayward Ave. Ann Arbor, MI 48109 USA
†Computer Sciences, University of Wisconsin, 1012 West Dayton St. Madison, WI 53706 USA

ABSTRACT

This paper studies the problem of protecting individual privacy when continuously publishing a stream of location trace data collected from a population of users. Fundamentally, this leads to the new challenge of anonymizing data that evolves in *predictable ways* over time. Our main technical contribution is a novel formal framework for reasoning about privacy in this setting. We articulate a new privacy principle called *temporal unlinkability*. Then, by incorporating a probabilistic model of data change (in this case, user motion), we are able to quantify the risk of privacy violations. Within this framework, we develop an initial set of algorithms for continuous privacy-preserving publishing. Finally, our experiments demonstrate the shortcomings of previous publishing techniques that do not account for inference based on predictable data change, and they demonstrate the feasibility of the new approach.

Categories and Subject Descriptors

H.2.7 [**Information Systems**]: Security, integrity, and protection

General Terms

Security

1. INTRODUCTION

Streaming location data from sensors and GPS devices is driving a broad new class of applications. This paper considers an organization that collects and continuously publishes a stream of location trace information from a population of users. For example, cellular phone providers can track users' locations using the GPS devices attached to modern phones. Often, there is a compelling reason to share or sell this information to a third party. For example, GPS traces provide valuable real-time traffic information. At the same time, there are also concerns for the privacy of GPS users.

Much of the past work in location-based privacy has focused on static snapshots, applying techniques like spatial k-anonymity to mask the locations of users at a single point in time [10, 14, 20, 24]. However, we are interested in repeatedly publishing the locations of users as they move, across an indeterminate amount of time. This poses a difficult challenge because users' locations change in ways that are often predictable, which means that it is often possible to infer the location of a user by analyzing historical locations, even if individual snapshots satisfy conventional notions of privacy. This basic idea, which we call *motion-prediction inference* is illustrated with a simple example.

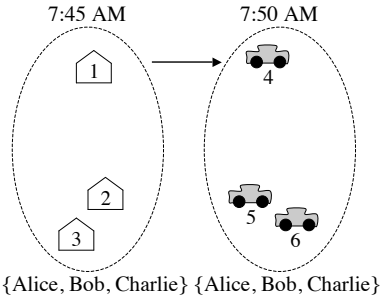

Figure 1: Example of Motion Prediction Inference

EXAMPLE 1.1. *Consider the scenario in Figure 1, where three users (Alice, Bob, and Charlie) live in the same neighborhood, and their locations are being tracked via GPS devices. At 7:45 AM, all three are at their homes, but in the interest of privacy, the cell phone company releases a "bucketized" snapshot of their locations, including the* anonymization group *indicating that Alice, Bob, and Charlie are at locations 1, 2, and 3, but eliminating the association between users and locations.* $(\{Alice, Bob, Charlie\}, \{1, 2, 3\})$,

Through access to auxiliary information*, an adversary can often associate certain individuals with unique locations (at certain points in time). For example, using the telephone book, the adversary may determine that location 1 is Alice's home.*

Suppose that at 7:50 AM, the phone company releases another snapshot containing $(\{Alice, Bob, Charlie\}, \{4, 5, 6\})$. *By conventional definitions, this snapshot would be considered* 3*-anonymous. However, this approach is clearly flawed. By carefully examining locations 4, 5, and 6, the adversary may discover based on his knowledge of motion patterns (e.g., speed limits and traffic patterns) that locations 5 and 6 are too far away from location 1 to have been reached in 5 minutes. Thus, he can infer that Alice is at location 4.*

1.1 Contributions

Motion-prediction inference is a significant challenge in privacy-preserving online location-trace publishing. While we describe our results in this context, this is just one example of a more general class of problems in which it is necessary to repeatedly publish data that evolves in semi-predictable ways.[1] In this paper, we propose a

[1]As a second example, consider a longitudinal social science study that tracks a set of subjects over several years, and in which it is important to publish new data every year. Just as the location of Alice at 7 AM is correlated with her location at 7:05, certain personal attributes (e.g., age, residence, personal habits) tend to vary in predictable ways over time.

MobiDE'10, June 6, 2010, Indianapolis, Indiana, USA.

formal framework for dealing with this problem. The framework is highlighted by the following key contributions:

1. We formalize a high-level privacy principle called *temporal unlinkability* for dynamically-evolving data. (Section 2.1)
2. To address the challenge of motion-prediction inference, our framework incorporates an extensible ("plug-and-play") *motion model* that probabilistically describes the movements of users. (Section 3) This approach is more flexible and general than past work, which was only capable of expressing inference based on maximum speed [11].
3. Using the motion model, we can compute a *breach probability*, which measures the certainty with which an adversary can violate temporal unlinkability. (Section 3)
4. Finally, we propose two protocols for continuously publishing location traces without causing breaches. (Section 4)

Our experimental study (Section 5) confirms that static anonymization tools (e.g., spatial k-anonymity) are indeed vulnerable to privacy breaches if we fail to account for motion-prediction inference. However, the experiments indicate that our protocols can be used to mitigate this problem.

2. PRELIMINARIES

2.1 Threat Model & Privacy Principle

In our problem setting, we expect to encounter an adversary who has access to some source of *auxiliary information* associating specific users with particular locations at specific points in time. For example, the Yellow Pages list the home addresses of many people, and can be used to identify their locations during the night. More generally, auxiliary information can associate individual users with spatio-temporal paths, such as a user's route home from the office. Of course, the data publisher does not always have full knowledge of the auxiliary information available to the adversary.

Fundamentally, we view the association between individual users and locations as private. However, because of auxiliary information, we recognize that we will not be able to prevent an adversary from determining the location of a particular user at a time when this information is already known as part of the auxiliary information. (In Example 1.1, the adversary *already knows* Alice's location at 7:45 AM.) However, we should be able to prevent the adversary from further leveraging this information to learn the locations of users at *other* points in time. (In Example 1.1, the adversary was able to infer Alice's location at 7:50 AM.)

Formally, we consider location information collected from a finite population of n users, each with a unique identifier in $\{u_1, ..., u_n\}$. We assume that the system publishes location-based data in discrete time *epochs* labeled $t_0, t_1, ...$

DEFINITION 1 (PRINCIPLE OF TEMPORAL UNLINKABILITY). *Consider an adversary who knows the location of a target user u during m sequential epochs $t_i, ..., t_{i+m}$. Using the published data, and under reasonable assumptions of inference, the adversary should not be able, with high confidence, to determine the location of u during some other epoch $t_j \notin \{t_i, ..., t_{i+m}\}$.*

Given this guiding principle, it is instructive to consider a couple of naive publishing protocols, which illustrate the challenge in satisfying temporal unlinkability:

- First, consider a strawman in which unique identifiers $u_1, ..., u_n$ are replaced with pseudonyms $p_1, ..., p_n$ in the published data. For example, Alice's name can be replaced with a unique hash value. Clearly, this approach violates temporal unlinkability; once an adversary "unmasks" a user (learns her pseudonym using auxiliary information), he is able to learn the user's location during all other epochs.
- A more clever strawman would eliminate the use of pseudonyms. However, this approach must still be applied with caution. Using multi-target tracking tools (which implicitly model user motion), it is often still possible to track a particular user across epochs [15, 21], again violating temporal unlinkability.

In the remainder of this paper, we will develop a framework for reasoning about temporal unlinkability in the presence of motion prediction, as well as publishing tools to prevent violations thereof.

2.2 Cloaking Mechanism

In privacy, like security, it is useful to draw a distinction between the privacy *policy* (i.e., the desired set of formal guarantees) and the *mechanism* used to enforce the policy. The primary focus of this paper is in developing an appropriate policy for time-evolving data. However, out of necessity, we chose to work with a particular mechanism, which we selected because it generalizes several other proposals.

Consider a finite population of users, and suppose that each user has been assigned a unique pseudonym in $\{p_1, ..., p_n\}$. These values do not identify the users externally, but they are consistent across time. During each epoch, a *location snapshot* associates each user's pseudonym with the user's location.

DEFINITION 2 (LOCATION SNAPSHOT). *A location snapshot associates each user with a single location during a particular epoch. During epoch t_j, $D(t_j) = \{(p_1, l_1^{(j)}), ..., (p_n, l_n^{(j)})\}$ indicates that user p_i is at location $l_i^{(j)}$.*

Throughout this paper, we consider "bucketized" location snapshots. A *release candidate* is modeled as a set of *anonymization groups*. Each anonymization group contains a non-overlapping set of pseudonyms and a multiset of locations, but within each group, the association between pseudonyms and locations is broken.

DEFINITION 3 (RELEASE CANDIDATE). *A release candidate $D^*(t_j)$ for location snapshot $D(t_j)$ is of the form $\{(C_1(t_j), L_1(t_j)), ..., (C_B(t_j), L_B(t_j))\}$, such that $\cup_{i=1..B} C_i(t_j) = \{p_1, ..., p_n\}$, $C_i(t_j) \cap C_m(t_j) = \emptyset$ for $i \neq m$, and $L_i(t_j)$ contains the locations at time t_j of all users with pseudonyms in $C_i(t_j)$.*

Of course, a variety of masking mechanisms have been proposed in the literature; we are careful to note the relationship between this approach and other proposals:

- **Spatial Cloaking** Spatial cloaking techniques replace the precise locations of individuals with coarsened regions. Such representations (e.g., minimum bounding rectangles [10]) can be computed from the anonymization groups in our release candidates, and thus reveal no more information. Further, cloaking techniques may still be vulnerable to attacks based on motion models. In Example 1.1, if we replaced the precise locations in each anonymization group with bounding regions, it is still possible to infer that Alice could not have reached the lower portion of the second region by 7:50 AM.
- **No Pseudonyms** Another approach would eliminate the use of pseudonyms entirely. This can be modeled as a special case of our mechanism, where each $D^*(T_j)$ contains just one anonymization group. Again, this approach may still be vulnerable to motion prediction inference, as illustrated by Example 1.1.
- **Location Densities** A third alternative would publish maps of user-densities. For example, such a map might indicate that at 7:45 AM, there are three users in a particular region. Such maps, at epoch T_j, can be computed from a release candidate $D^*(T_j)$ that contains just one anonymization group.

2.3 Data Quality

When anonymizing data, there is often a tradeoff between privacy and the *quality* or *utility* of the resulting data. In the case of the bucketization mechanism, there are two dimensions of data quality to be considered: On one hand, we want to publish data with spatially-compact anonymization groups (maximize *spatial precision*). On the other hand, we would like to publish a release candidate during as many epochs as possible (maximize *publication frequency*). To maintain temporal unlinkability, however, an increase in spatial precision often leads to a reduction in publication frequency. The relative importance of these two dimensions varies based on the application. While this tradeoff is largely orthogonal to our framework for reasoning about privacy, we will revisit data quality when describing publication protocols in Section 4.3.

3. LOCATION TRACE PRIVACY

In this section, we describe our framework for reasoning about location trace privacy. The framework involves two components: a probabilistic *motion model*, and a *breach probability function*.

The motion model describes the adversary's knowledge with respect to user motion patterns. While recent work has considered very specific and rudimentary forms of knowledge about motion patterns (e.g., [11] assumed that the adversary knows only the maximum speed of users), our framework is flexible enough to capture a much broader class of motion patterns. Specifically, we allow the motion model to be "plugged-in" to the framework, provided that it satisfies a general form. This enables us to capture knowledge about speed (as in [11]), but also an array of other kinds of knowledge, including directionality (e.g., objects tend to continue moving in the same directions), minimum speed, and speed distribution, that still pose a threat in existing work.

Using the motion model, we formalize the idea of a *privacy breach*, based on the temporal unlinkability principle.

3.1 Motion Models

Central to our framework is a probabilistic *motion model*. As illustrated in Example 1.1, the location of a user at epoch t_j is often correlated with the user's location at surrounding points in time. We use the motion model to define the probability distribution of locations for a particular user at time t_j, given the location of the user at the preceding h epochs (*forward motion model*), or the following h epochs (*backward motion model*).[2]

In the following definitions we use capital letters to denote variables (locations, epochs, and users), and we use lower-case letters to denote instances.

DEFINITION 4 (FORWARD MOTION MODEL TEMPLATE). *A forward motion model is a conditional probability mass function of the following form, where $1 \leq h \leq j$ and $Loc(P, T_j) = L_j$ indicates that the location of user P at epoch T_j is L_j:*

$$\Pr[Loc(P,T_j) = L_j \quad | \quad Loc(P,T_{j-1}) = L_{j-1}, ..., Loc(P,T_{j-h}) = L_{j-h}]$$

We will view the forward motion model as an h^{th}-order Markov chain. That is, we assume $\Pr[\text{Loc}(P,T_j) = L_j | \text{Loc}(P,T_{j-1}) = L_{j-1}, ..., \text{Loc}(P,T_{j-h}) = L_{j-h}] = \Pr[\text{Loc}(P,T_j) = L_j | \text{Loc}(P,T_{j-1}) = L_{j-1}, ..., \text{Loc}(P,T_0) = L_0]$.

Similarly, we define the backward motion model, which we will also view as an h^{th}-order Markov chain.

DEFINITION 5 (BACKWARD MOTION MODEL TEMPLATE). *A backward motion model is a conditional probability mass function of the following form, where $1 \leq h \leq j$:*

$$\Pr[Loc(P,T_j) = L_j \quad | \quad Loc(P,T_{j+1}) = L_{j+1}, ..., Loc(P,T_{j+h}) = L_{j+h}]$$

The *symmetry property* says that a motion model can be read forwards and backwards.

DEFINITION 6 (MOTION MODEL SYMMETRY). *Backwards and forwards motion models are said to be symmetric if*

$$\begin{aligned} &\Pr[Loc(P,T_j) = L_j \quad | \quad Loc(P,T_{j-1}) = L_{j-1}, ..., Loc(P,T_{j-h}) = L_{j-h}] \\ &= \Pr[Loc(P,T_{j-h}) = L_{j-h} \quad | \quad Loc(P,T_{j-h+1}) = L_{j-h+1}, ..., Loc(P,T_j) = L_j] \end{aligned}$$

The motion model is an independent and replaceable component of our framework. In our experiments (Section 5), we will use a sample linear motion model, which instantiates the more general template for $h = 1$. The sample motion model is based on velocity (speed and directionality) distribution assumptions, assuming that the speed of each user P is uniformly distributed in the range $[v_1, v_2]$, and that the angle of motion is uniformly distributed in $[\theta_1, \theta_2]$. The sample motion model satisfies the symmetry property; details can be found in the extended paper [19].

Of course, there are many ways of modeling user motion (e.g., [16, 28, 18]). Many of these models rely on using the previous locations of an object to predict future locations, and can thus be plugged into our framework.

3.2 Privacy Breaches

Using the motion model as a building block, we formally define what constitutes a breach of privacy, based on the unlinkability principle. Intuitively, the forward (respectively, backward) *breach probability* represents the certainty with which an adversary can identify the location associated with a particular user P during epoch T_j, using the motion model, given that he knows the locations of *all users* during the m preceding (respectively, following) sequential epochs, as described by fully-identified snapshots $D(T_{j-1}), ..., D(T_{j-m})$ (respectively, $D(T_{j+1}), ..., D(T_{j+m})$).

DEFINITION 7 (FORWARD BREACH PROBABILITY). *The forward breach probability for user P, epoch T_j and location L_j is defined by the conditional probability*

$$\Pr[Loc(P,T_j) = L_j | D(T_{j-1}), ..., D(T_{j-m}), D^*(T_j)] \quad (1)$$

The forward breach probability can be expressed in terms of the forward motion model. Note that the snapshots $D(T_{j-1}), ..., D(T_{j-m})$ identify the locations of each pseudonym P at the m previous epochs. (Denote these locations $l^P_{j-1}, ..., l^P_{j-m}$.) Assuming $h \leq m$, based on the h-step Markov assumption, we have:

$$\begin{aligned} &\Pr[Loc(P,T_j) = L | D(T_{j-1}), ..., D(T_{j-m})] \\ = \ &\Pr[Loc(P,T_j) = L | Loc(P,T_{j-1}) = l^P_{j-1}, ..., Loc(P,T_{j-h}) = l^P_{j-h}] \end{aligned}$$

In order to compute the breach probability, we must also condition on $D^*(T_j)$:

$$\begin{aligned} &\Pr[Loc(P,T_j) = L | D(T_{j-1}), ..., D(T_{j-m}), D^*(T_j)] \\ = \ &\frac{\Pr[Loc(P,T_j) = L \wedge D^*(T_j) | D(T_{j-1}), ..., D(T_{j-m})]}{\Pr[D^*(T_j) | D(T_{j-1}), ..., D(T_{j-m})]} \end{aligned}$$

The resulting probabilities can be computed based on the forward motion model. In the following, for simplicity of notation, the past locations of each pseudonym P are assumed, and we will

[2]Note that this assumes the movements of specific users are independent of one another.

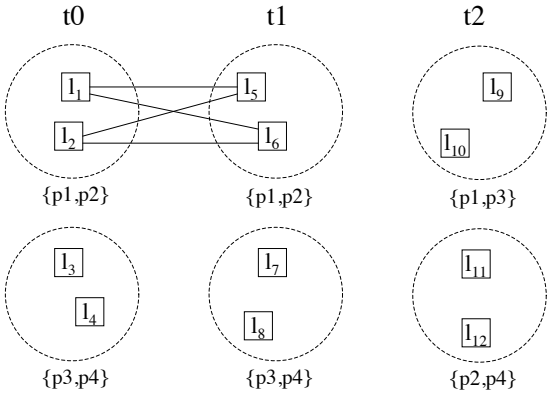

Figure 2: Example of breach probability computation

simply refer to the conditional probability that the location of P is L at T_j (as computed form the motion model) as $\Pr[(P, L)]$.

Consider the anonymization group in $D^*(T_j)$ that contains the pseudonym P, and let $C_P^{(j)}$ and $L_P^{(j)}$ denote the sets of pseudonyms and locations, respectively, contained in this anonymization group. Let $M : C_P^{(j)} \rightarrow L_P^{(j)}$ be a one-to-one function mapping pseudonyms to locations. Notice that there are $g = |L_P^{(j)}|!$ such functions, and $D^*(T_j)$ implies that one such mapping must be true.

Each unique mapping M can be viewed as a disjoint event. Since we assume that the movements of users are independent of one another, the probability of one such mapping $M = \{(c_1, \ell_1), ..., (c_k, \ell_k)\}$ is $\Pr[M] = \Pr[(c_1, \ell_1)] \cdot ... \cdot \Pr[(c_k, \ell_k)]$.

Finally, the forward breach probability for pseudonym P and location L can be computed as the sum of probabilities of mappings M_i such that $M_i(P) = L$, divided by the sum of probabilities over all such mappings. In the following, $I(M_i)$ is an indicator variable, which takes the value 1 if $M_i(P) = L$, and 0 otherwise.

$$BP = \frac{\sum_{i=1}^{g} \Pr[M_i] \cdot I(M_i)}{\sum_{i=1}^{g} \Pr[M_i]} \quad (2)$$

EXAMPLE 3.1. *Consider the simple example in Figure 2, and suppose $m = h = 1$. Suppose that the true snapshot at epoch t_0 is $D(t_0) = \{(p_1, l_1), (p_2, l_2), (p_3, l_3), (p_4, l_4)\}$.*

Using the 1-step forward motion model, we can compute the following:

$e_{15} = \Pr[Loc(p_1, t_1) = l_5 | Loc(p_1, t_0) = l_1]$;
$e_{16} = \Pr[Loc(p_1, t_1) = l_6 | Loc(p_1, t_0) = l_1]$;
$e_{25} = \Pr[Loc(p_2, t_1) = l_5 | Loc(p_2, t_0) = l_2]$;
$e_{26} = \Pr[Loc(p_2, t_1) = l_6 | Loc(p_2, t_0) = l_2]$.

Then, we can compute the breach probabilities.

$\Pr[Loc(p_1, t_1) = l_5 | D(t_0), D^*(t_1)] = \frac{e_{15}e_{26}}{e_{15}e_{26}+e_{25}e_{16}}$;
$\Pr[Loc(p_2, t_1) = l_6 | D(t_0), D^*(t_1)] = \frac{e_{15}e_{26}}{e_{15}e_{26}+e_{25}e_{16}}$;
$\Pr[Loc(p_1, t_1) = l_6 | D(t_0), D^*(t_1)] = \frac{e_{25}e_{16}}{e_{15}e_{26}+e_{25}e_{16}}$;
$\Pr[Loc(p_2, t_1) = l_5 | D(t_0), D^*(t_1)] = \frac{e_{25}e_{16}}{e_{15}e_{26}+e_{25}e_{16}}$.

In addition to the forward breach probability, we have the backward probability, which is similarly defined, and can be computed in terms of the backward motion model.

DEFINITION 8 (BACKWARD BREACH PROBABILITY). *The backward breach probability for user P, epoch T_j and location L_j is defined by the conditional probability*

$$\Pr[Loc(P, T_j) = L_j | D(T_{j+1}), ..., D(T_{j+m}), D^*(T_j)] \quad (3)$$

Finally, a release candidate is said to cause a privacy breach if there is some (forward or backward) breach probability that is higher than a user-specified threshold.

DEFINITION 9 (PRIVACY BREACH). *A release candidate $D^*(T_j)$ is said to cause a privacy breach if either of the following statements is true for user-defined breach threshold T: (This definition is also easily adapted to use a different threshold for each user.)*

$$max_{P,L_j} \Pr[Loc(P, T_j) = L_j | D(T_{j-1}), ..., D(T_{j-m}), D^*(T_j)] > T$$
$$max_{P,L_j} \Pr[Loc(P, T_j) = L_j | D(T_{j+1}), ..., D(T_{j+m}), D^*(T_j)] > T$$

In the remainder of the paper, we will refer to a release candidate $D^*(T_j)$ that fails to satisfy the first condition as causing a *forward breach*; similarly, if it fails to satisfy the second condition, it causes a *backward breach.*

4. PUBLISHING LOCATION TRACES

Using the framework described in the last section, the remaining challenge is to develop a protocol for continuously publishing location traces in a way that does not breach privacy. In this section, we first discuss how to check a release candidate for breaches. Then, we present some initial ideas for publishing protocols that maintain data utility without compromising privacy.

4.1 Checking for Breaches: Brute-Force

Algorithm 1 provides a brute-force method for checking a release candidate $D^*(T_j)$ for forward privacy breaches, given snapshots $D(T_{j-h}), ..., D(T_{j-1})$, using the forward motion model. The time complexity is $O(\frac{n}{k} \cdot k!) = O(nk^k)$, where k is the maximum size of an anonymization group. In some cases, k is a small constant, in which case the complexity is $O(n)$. To manage cases when this is not true, we provide some additional heuristic optimizations in Section 4.2.

noend 1 Forward Check (Brute-Force)

Input: $D^*(T_j), D(T_{j-1}), ..., D(T_{j-h}), T$
Output: $true$ if there is a breach, $false$ otherwise

```
1: for each anonymization group (C < L) ∈ D*(T_j) do
2:     denom = 0
3:     numer[|C|][|L|] = initialize all entries to 0
4:     for each unique mapping M : C → L do
5:         Pr[M] =(compute from forward motion model)
6:         for each p ∈ C do
7:             numer[p][M(p)] += Pr[M]
8:         denom += Pr[M]
9:     for each p ∈ C do
10:        for each ℓ ∈ L do
11:            BP = numer[p][ℓ] / denom
12:            if BP > T then
13:                return true
14: return false
```

The brute-force algorithm for checking for backward breaches is analogous, but takes as input $D^*(T_j), D(T_{j+1}), ..., D(T_{j+h})$, and uses the backward motion model.

4.2 Checking for Breaches: Pruning

The brute-force checking algorithm is exponential in k, the maximum size of an anonymization group. In this section we describe a fast pruning algorithm that is often able to identify anonymization groups that do and do not cause breaches (have breach probabilities above and below threshold T, respectively), heuristically reducing the amount of computation.

Recall the formula for computing the breach probability in Equation 2. If k is the size of the anonymization group, then the numerator of this formula is the sum of $(k-1)!$ elements, each of which is the product of k different $Pr[(C, L)]$ values: $Pr[(c_1, l_1)] \cdot ... \cdot$

$Pr[(c_k, l_k)]$. The denominator is the sum of $k!$ elements, each of which is the produce to k different $Pr[(C, L)]$ values. By choosing the maximum and minimum values of $Pr[(C, L)]$, we can find (loose) upper and lower bounds for the breach probability in G.

4.2.1 Basic Pruning Approach

The basic pruning procedure consists of the following three steps. (For simplicity, we describe forward breach probability computation, but the procedure for backward breach probabilities is completely analogous.)

1. Consider the locations $l_1, ..., l_k$ for the set of objects $c_1, ..., c_k$ in anonymization group G at T_j. Applying the forward motion model, we compute $PR_i = \{\Pr[(c_1, l_i)], \ldots, \Pr[(c_k, l_i)]\}$ for $1 \leq i \leq k$ at T_j. (Again, we assume that the locations of each object at the previous m epochs are known, so these probabilities are easily obtained from the motion model.) This step takes $O(k^2)$.
2. For $1 \leq i \leq k$, let $P_i = max(PR_i)$, and let $p_i = min(PR_i)$.
3. Finally, we can obtain (loose) upper and lower bounds for the breach probability BP in anonymization group G.

$$BP \leq \frac{(k-1)! \cdot P_1 \cdot \ldots \cdot P_k}{k! \cdot p_1 \cdot \ldots \cdot p_k} = \frac{1}{k} \cdot \frac{P_1 \cdot \ldots \cdot P_k}{p_1 \cdot \ldots \cdot p_k}$$
$$BP \geq \frac{1}{k} \cdot \frac{p_1 \cdot \ldots \cdot p_k}{P_1 \cdot \ldots \cdot P_k}$$

Since there are, on average, n/k anonymization groups, the total time complexity is $O(\frac{n}{k} \cdot k^2) = O(nk)$.

EXAMPLE 4.1. *To illustrate the pruning procedure, consider a simple example. Suppose the following probabilities are computed during Step 1:* $\Pr[(c_1, l_1)] = 0.5, \Pr[(c_2, l_1)] = 0.35, \Pr[(c_3, l_1)] = 0.4$, $\Pr[(c_1, l_2)] = 0.31, \Pr[(c_2, l_2)] = 0.45, \Pr[(c_3, l_2)] = 0.35$, $\Pr[(c_1, l_3)] = 0.19, \Pr[(c_2, l_3)] = 0.2, \Pr[(c_3, l_3)] = 0.25$

Upper and lower bounds can be computed as follows:

$$BP \leq \frac{1}{3} \cdot \frac{0.5 \cdot 0.45 \cdot 0.25}{0.35 \cdot 0.31 \cdot 0.19} = 90.9\%$$
$$BP \geq \frac{1}{3} \cdot \frac{0.35 \cdot 0.31 \cdot 0.19}{0.5 \cdot 0.45 \cdot 0.25} = 12.2\%$$

Suppose that the breach threshold $T = 95\%$. Since $BP \leq 90.9\% \leq T$, we know that there is not a breach.

4.2.2 An Improvement

The basic pruning approach uses $P_1 \cdot \ldots \cdot P_k$ and $p_1 \cdot \ldots \cdot p_k$ to estimate the probabilities of the most and least likely assignments of objects to locations. By plugging these values into Equation 2, we can obtain upper and lower bounds for the breach probability. However, if the difference between the maximum and minimum estimates is large, the estimated bounds can be quite loose. To improve these bounds, we make the following observation: In Equation 2, notice that each M_i (assignment of objects to locations) must be unique. Rather then finding the single maximum- and minimum-probability assignment, we can improve the tightness of the bounds by finding the x most-probable and x least-probable assignments, and incorporating these into the bound. The improved pruning algorithm consists of the following steps:

1. Let $S = \{s_1 \cdot \ldots \cdot s_k : s_1 \in PR_1, \ldots, s_k \in PR_k\}$ denote the multiset of probabilities obtained by assigning one object per location. Let $max[x]$ denote the x^{th} largest value in S, and let $min[x]$ denote the x^{th} smallest value in S.
2. Next, we must compute $max[1], ..., max[x]$ and $min[1], ..., min[x]$. There is a polynomial-time algorithm. We omit the details for space, but they can be found in [19].
3. Finally, we can compute upper and lower bounds. (The following assumes that $x \leq (k-1)!$.)

$$BP \leq \frac{max[1] + \ldots + max[x] + ((k-1)! - x) \cdot max[x]}{min[1] + \ldots + min[x] + (k! - x) \cdot min[x]}$$
$$BP \geq \frac{min[1] + \ldots + min[x] + ((k-1)! - x) \cdot min[x]}{max[1] + \ldots + max[x] + (k! - x) \cdot max[x]}$$

EXAMPLE 4.2. *Consider again the probabilities in Example 4.1, and suppose $x = 2$. In this case, we compute the following:*

$$max[1] = 0.5 \cdot 0.45 \cdot 0.25 = 0.05625$$
$$max[2] = 0.4 \cdot 0.45 \cdot 0.25 = 0.045$$
$$min[1] = 0.35 \cdot 0.31 \cdot 0.19 = 0.020615$$
$$min[2] = 0.35 \cdot 0.31 \cdot 0.2 = 0.0217$$

Then, upper and lower bounds can be computed as follows. Notice that the bounds are tighter than those obtained using the basic pruning approach in Example 4.1.

$$BP \leq \frac{0.05625 + 0.045}{0.020615 + 0.0217 + 4 \cdot 0.0217} = 78.42\%$$
$$BP \geq \frac{0.020615 + 0.0217}{0.05625 + 0.045 + 4 \cdot 0.045} = 15.05\%$$

4.3 Publishing Protocols

Recall that we selected the cloaking mechanism in Section 2.2 for flexibility. Generally-speaking, this mechanism gives us two tools to work with in order to guarantee that a published stream of location trace data does not breach privacy. First, we can increase the size, or vary the composition, of anonymization groups. Second, we can limit the frequency with which we publish a release candidate. (We can also use these two tools in combination.) In this section, we provide an initial exploration of the space, considering the problem from the perspective of a fixed publication schedule and from the perspective of a fixed set of anonymization groups.

4.3.1 Fixed Publication Schedule

First, consider the case in which we publish release candidates on a fixed schedule. In other words, we must publish *some* release candidate at every epoch T_j (assuming, of course, that there exists a release candidate that does not cause a breach), but we can vary the size and composition of anonymization groups.

In this case, if we want to publish a release candidate $D^*(T_j)$ at epoch T_j, we need to check for backward breaches, and we must have future snapshots $D(T_{j+1}), ..., D(T_{j+h})$ in hand to do this. A simple solution is to delay publishing for h subsequent epochs, after which $D^*(T_j)$ is easily checked for (forward and backward) breaches.

For the case of the fixed publication schedule, we can then view the problem of selecting a release candidate in terms of constrained optimization: Given an objective function (i.e., a measure of utility), find the *best* release candidate that does not cause a breach. There are many ways to measure utility, one of which is based on the idea of *spatial precision*, or the idea that anonymization groups should be spatially compact.[3] In this case, the optimization problem can be stated as follows:

PROBLEM 1. *Given current snapshot $D(T_j)$, historical snapshots $D(T_{j-h}), ..., D(T_{j-1})$, future snapshots $D(T_{j+1}), ..., D(T_{j+h})$, forward and backward motion models, and breach threshold T, find $D^*(T_j) = \{(C_1(T_j), L_1(T_j)), \ldots, (C_B(T_j), L_B(T_j))\}$ such that*

1. *$D^*(T_j)$ does not cause a (forward or backward) privacy breach, and*

[3]Related objective functions, based on area or volume of resulting clusters, have been used in prior work [2, 3].

2. *The objective* $\max_{i=1..B} R(C_i)$ *is minimized, where* $R(C_i)$ *is the radius of* C_i.

THEOREM 1. *Problem 1 is NP-hard. (The proof can be found in the extended paper [19].)*

In light of this result, and the combinatorial nature of the checking algorithms described in Sections 4.1 and 4.2, it is not likely that we will be able to provide an optimal solution to Problem 1. From a practical perspective, a compromise solution leverages an existing (heuristic or approximation) algorithm for k-anonymity (e.g., [2, 3, 10, 22]) to generate a release candidate $D^*(T_j)$. If $D^*(T_j)$ does not cause a breach, it can be published; otherwise, do not publish during epoch T_j.

4.3.2 Durable Anonymization Groups

While the last section considered a fixed publication schedule, in this section we consider the case in which the anonymization groups are fixed, and the only decision to be made at each epoch is whether or not to publish the release candidate. We will refer to an anonymization group as *durable* if it contains the same pseudonyms at all epochs across time. That is, C_i is considered durable across epochs $t_i, ..., t_j$ if $C_i(t_i) = ... = C_i(t_j)$. Intuitively, in this case, the data utility goal is simply to publish a release candidate as often as possible.

Publication protocols involving only durable clusters have several appealing properties. In particular, while the approach described in the previous section (for evolving anonymization groups) requires that we check for forward and backward breaches, this is not necessary in the case where we require durable groups and where the motion model is symmetric. Not checking for backward breaches has several advantages: (1) It reduces the checking time by half, and (2) More importantly, there is no need to delay publishing for h epochs as in the general case.

THEOREM 2. *If all anonymization groups are durable, and the forward and backward motion models are symmetric, then it is sufficient to check just for forward breaches. (The proof can be found in the extended paper [19].)*

EXAMPLE 4.3. *Again, consider the example in Figure 2, and notice that the anonymization groups* $\{p_1, p_2\}$ *and* $\{p_3, p_4\}$ *are durable across* t_0 *and* t_1. *If the one-step motion model is symmetric, then the forward breach probabilities at* t_1 *are the same as the backward breach probabilities at* t_0. *Thus, it is sufficient to check only for forward breaches.*

In practice, when using a durable approach, a "burn-in" period can be used to discover *flocks* of users with similar motion patterns. (We could use an existing trajectory clustering algorithm such as [29] to find the flocks.) Also, note that these anonymization groups do *not* need to be durable in perpetuity. It is possible to re-cluster the users, temporarily reverting to the general case (forward and backward checks).

5. EXPERIMENTAL RESULTS

This section describes our experiments, which investigate the following issues:

- We use our framework to analyze the occurrence of the motion prediction inference problem. Much prior work has focused on applying k-anonymous cloaking to protect the locations of users at a single point in time [10, 12, 14, 20, 24]. However, to the best of our knowledge, all of these tools are vulnerable to motion prediction inference. Analyzing the output of two representative k-anonymization algorithms illustrates the importance of explicitly considering this threat.
- We evaluate the effectiveness of our publishing algorithms, including the pruning approach and the effect of using durable vs. non-durable clusters.

5.1 Experimental Data

For the experiments presented in this paper, we used real GPS traces from a study conducted by a Transportation Research Institute at Michigan. The dataset contains two-hour traces for 87 users. The data sampling rate is one centisecond (0.01 seconds). From these 87 trajectories, we were only able to use 72 trajectories because this is the maximum number of trajectories that have common time ranges. For the motion model, we assumed a uniform distribution over a range of speeds (0 to 170 km/hr) and angles (0 to 180 degrees), which were computed from the trajectory dataset.

We also conducted a similar set of experiments using the Network-based Generator of Moving Objects (NG-MO) [6], which simulates points moving in a road network. The results are omitted for space, but can be found in the extended paper [19].

5.2 Implementation and Experimental Setup

We implemented two protocols for data publication:

- **Durable Clusters** In the first protocol, the data is initially clustered into anonymization groups at epoch 1 using the clustering method in [2], which we call k-Condense. This method takes as input a parameter k, and uses a heuristic to cluster the points into groups based on their proximity, such that each resulting group contains at least k points. With durable clusters, once the cluster is produced at the first epoch, the clusters are retained and simply checked at subsequent epochs for forward breaches. Data is published if the forward breach probability for each cluster is below the threshold T. (see Definition 9 and Theorem 2)
- **Reclustering** In the second protocol, the data is reclustered at each epoch, using the k-Condense algorithm. At each epoch the breach probability is computed and the snapshot at an epoch is published if the forward and backward breach probability for each cluster is below the threshold T.

In addition, to illustrate the motion prediction inference problem, we also tried the r-Gather algorithm [3]. Like k-Condense, r-Gather was proposed for clustering generic microdata in a metric space. The algorithm clusters n points into a set of groups, each of which contains at least k points. (In other words, the algorithm guarantees k-anonymity for $k = r$.) We chose these two particular algorithms as representatives of the class of static publishing techniques that do not consider motion prediction inference.

All of our code is written in C++, and all experiments were run on an Intel Pentium 4 2.2 GHz duo workstation with 2GB of main memory and a 160 GB hard disk, running Windows Vista Ultimate.

In our experiments we use a 1-step linear motion model, incorporating both speed and directionality, as described in Section 3.1.

5.3 Motion Prediction Inference in Practice

Much prior work on location privacy has focused on applying k-anonymity to protect the locations on users at a single point in time. However, these techniques are all potentially vulnerable to motion prediction inference. To illustrate this point, we ran the static k-anonymization algorithms on location snapshots for epochs 1 to 10. (In order to effectively check for breaches at epoch 1, we also generated an initial snapshot at an epoch 0, which is not published.)

The results for the GPS data are shown in Figure 3, which plots the proportion of anonymization groups generated by the k-Condense method at each epoch that result in a privacy breach. These results are shown for $k = 4, 8$ and for breach probability threshold $T = 25\%$. From this figure, we observe that every published snapshot results in a privacy breach! We also observe that the number of

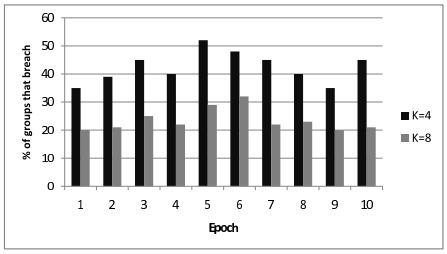

Figure 3: % of groups exceeding breach threshold T, k-Condense, k=4 and k=8, T=25%

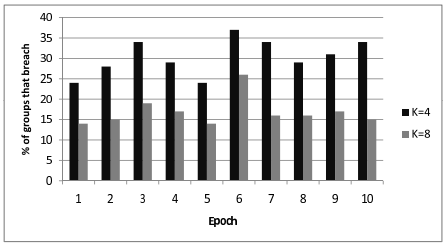

Figure 4: % of groups exceeding breach threshold T, r-Gather, k=4 and k=8, T=25%

groups exceeding the breach probability threshold is inversely proportional to the value of k; the release candidate with smaller k has more clusters that exceed breach probability T. This is intuitive; we expect larger clusters to provide better anonymization.

In addition to k-Condense, we performed the same experiment using r-Gather, and we observed similar results (see Figure 4). The small difference between the two results can be attributed to a simple observation: while the cluster size constraint is the same in both cases, on average, r-Gather generally produces clusters that are larger than those produced by k-Condense. Nonetheless, some clusters produced by r-Gather still exceed the breach probability threshold at all epochs.

5.4 Publishing with Durable and Non-Durable Clusters

Next, we tested the effectiveness of our publishing protocol using both durable and non-durable clusters, as described in Section 4.3.2. (For the results reported in this section, we use the pruning techniques described in Section 4.2.)

For non-durable clusters, we generated a new clustered release candidate at epochs 1 to 10, and we tested to see whether the release candidate could be published. For non-durable clusters, this check involved both forward and backward checks. For durable clusters, we generated a single clustering at epoch 1; in this case, we only need to check for forward breaches.

The results are shown in Figures 5 and 6 for $k = 4, T = 75\%$ and $k = 12, T = 25\%$. (We conducted similar experiments for additional values of k and T, but the results are omitted for space.) In all cases, the time to check the breach probabilities is smaller with the durable clusters than with the non-durable clusters, as expected. The performance measurements for non-durable clusters include the cost of re-clustering at each epoch, as well as forward and backward breach checking. In contrast, in the case of durable clusters, we only cluster the data once, at epoch 1. In the remaining epochs, we must only perform a forward breach check.

We found that for $k = 4, T = 25\%$ (not shown), we could not publish any release candidates. However, if we increase k to 12, or increase T to 75% (both shown), we can publish during nearly every epoch. We also observed that, when we kept k constant, and increased T, the total computation time decreased due to more effective pruning.

Next, we examine the effects of increasing k. We found that increasing k allows more release candidates to be published, but that it also increases computation time. As discussed previously, larger values of k tend to lead to better anonymization. However, increasing k also increases the computational cost of checking for privacy breaches.

5.5 Efficiency and Effectiveness of Pruning

The final set of experiments evaluate the effectiveness of the pruning described in Section 4.2. Due to space constraints, we will only present results for $k = 8$ and $T = 50\%$. The results are shown in Figure 7, and we observe that our pruning method results in significant performance improvements (by 2X or more in most cases). The reason for this is that the pruning method can save the (expensive) computation of the exact maximum breach probability.

Notice that in Figure 7, when not using the pruning method, regardless of the durable or the non-durable case, when a release candidate can be published, the processing time is the same. For example, in the non-durable case (without pruning) when release candidates can be published, the processing time is about 2.4 seconds. The reason for this behavior is that in these cases the computation cost is the same as exact breach probabilities have to be computed for all groups.

In addition, we found that pruning is more effective for larger T (e.g., 75% vs. 25%) and smaller k (e.g., 4 vs. 12).

6. RELATED WORK

Privacy and anonymity have drawn considerable recent interest in location-aware applications. The majority of this work has focused on location-based services (LBS), applying techniques such as spatial k-anonymity to disguise locations of individual users in static snapshots (i.e., single points in time)[10, 12, 14, 20, 24, 30].

In contrast, relatively little work has considered the challenges posed by continuously publishing a stream of evolving location data. Two of the first proposals for addressing this problem were *mix-zones* [4] and *uncertainty-aware path cloaking* [17], which sought to maintain properties similar to temporal unlinkability in a less formal way, but neither provided any formal privacy guarantee.

Ghinita et al. considered an attack on static cloaking mechanisms, in which an adversary uses background knowledge of *maximum speed* to infer more specific location information [11]. Our framework is more flexible in that we can incorporate a variety of different types of motion-based background knowledge (e.g., including directionality, minimum speed, etc.) to which the techniques in [11] are not resilient. Our threat model (temporal unlinkability) is also somewhat different from the threat model in this paper, which is based on reducing the size of cloaking regions.

Yarovoy et al. [33] also consider the online location-publishing problem. Their solution addresses the problem posed by overlapping cloaking regions, but it does not take into account inference based on motion prediction. In the context of anonymizing requests to location-based service providers, Bettini et al. [5] describe the problem posed service-request linkability (guessing that two requests came from the same user), but they do not provide any formal guarantees against motion prediction inference. Chow et al. [8] consider a similar problem for continuous queries, but also do not address the problem of motion prediction. Gkoulalas-Divanis et al. proposes anonymizing LBS request using frequent trajectories [13]. The idea is that the user's location should not just be k-anonymous at the time of the request, but also for a surrounding window of time. It is not clear, however, whether this approach can be applied in our setting, where location updates are published frequently and in real-time.

The problem of continuous location-trace publishing is in some ways related to the problem of k-anonymous trajectory publishing

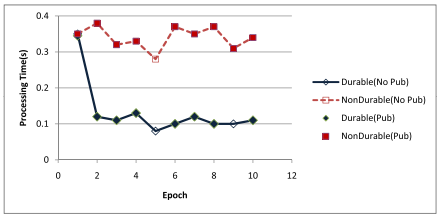

Figure 5: Durability Test, k=4, T=75%

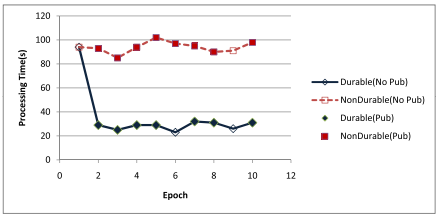

Figure 6: Durability Test, k=12, T=25%

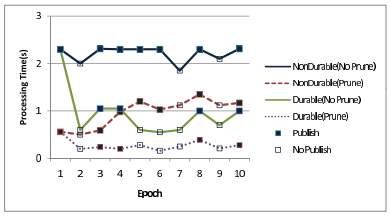

Figure 7: Time Comparisons, k=8, T=50%

[1, 25, 29]. However, there are some notable differences, which prevent the application of these techniques to our problem. In particular, this work is focused on anonymizing (offline) a database of *fully-specified* trajectories. However, after doing this, it is unclear whether we would be able to publish future location information for the same users without causing a privacy breach.

Beyond spatial data, privacy has been studied extensively for publishing generic personal data (e.g., in demographic research) [23, 26, 27, 31]. Recently, several techniques have been proposed to extend these static one-time publishing techniques to a dynamic setting, involving incrementally-updated data sets [7] or multiple releases [32, 9]. While the locations in our work can be viewed as "quasi-identifiers," to the best of our knowledge, none of the past work has considered the issue of tracking quasi-identifier values that evolve in predictable (non-random) ways over time.

Finally, considerable research has focused on motion modeling, trajectory prediction, and tracking [28, 18, 16].

7. CONCLUSION

In this paper, we developed the first formal framework for reasoning about privacy in the context of continuously publishing location traces. Our framework is based on the idea of *temporal unlinkability*: Given an adversary who already knows the location of a user at certain points in time, we want to limit the certainty with which he can identify this user at other times. Technically, the main challenge in achieving temporal unlinkability is *motion prediction inference*. Our framework addresses this problem using a pluggable motion model, which predicts the movements of a population of users; using the motion model, we provide a formal characterization of what constitutes a violation of temporal unlinkability (a *privacy breach*).

Using this framework, we developed several simple and effective protocols for continuously publishing location traces. Our experimental results both confirm the problem of motion prediction inference and indicate the feasibility of our new approach.

8. REFERENCES

[1] O. Abul, F. Bonchi, and M. Nanni. Never walk alone: Uncertainty for anonymity in moving objects databases. In *ICDE*, 2008.

[2] C. Aggarwal and P. Yu. A condensation approach to privacy-preserving data mining. In *EDBT*, 2004.

[3] G. Aggarwal, T. Feder, K. Kenthapadi, R. Panigrahy, D. Thomas, and A. Zhu. Achieving anonymity via clustering in a metric space. In *PODS*, 2006.

[4] A. Beresford and F. Stajano. Location privacy in pervasive computing. *IEEE Pervasive Computing*, 2003.

[5] C. Bettini, X.S. Wang, and S. Jajodia. Protecting privacy against location-based personal identification. In *VLDB Workshop on Secure Data Management*, 2005.

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

[7] J. Byun, Y. Sohn, E. Bertino, and N. Li. Secure anonymization for incremental datasets. In *SIAM Data Mining*, 2006.

[8] C.-Y. Chow and M. Mokbel. Enabling private continuous queries for revealed user locations. In *Advances in Spatial and Temporal Databases*, 2007.

[9] B. Fung, K. Wang, A. Fu, and J. Pei. Anonymity for continuous data publishing. In *EDBT*, 2008.

[10] B. Gedik and L. Liu. Location privacy in mobile systems: A personalized approach. In *ICDCS*, 2005.

[11] G. Ghinita, M. Damiani, C. Silvestri, and E. Bertino. Preventing velocity-based linkage attacks in location-aware applications. In *ACM GIS*, 2009.

[12] G. Ghinita, P. Kalnis, and S. Skiadopoulis. Prive: Anonymous location-based queries in distributed mobile systems. In *WWW*, 2007.

[13] A. Gkoulalas-Divanis, V. Verykios, and M. Mokbel. Identifying unsafe routes for network-based trajectory privacy. In *SIAM Data Mining*, 2009.

[14] M. Gruteser and D. Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *Conference on Mobile Systems, Applications and Services*, 2003.

[15] M. Gruteser and B. Hoh. On the anonymity of periodic location samples. In *Proceedings of the Second International Conference on Security in Pervasive Computing*, 2005.

[16] R. H. Güting and M. Schneider. *Moving Objects Databases*. Morgan Kaufmann, San Francisco, 2005.

[17] B. Hoh, M. Gruteser, H. Xiong, and A. Alrabady. Preserving privacy in GPS traces via uncertainty-aware path cloaking. In *CCS*, 2007.

[18] H. Jeung, Q. Liu, H. T. Shen, and X. F. Zhou. A hybrid prediction model for moving objects. In *ICDE*, 2008.

[19] W. Jin, K. LeFevre, and J. Patel. An online framework for publishing dynamic privacy-sensitive location traces. University of Michigan Tech. Report CSE-TR-553-09, 2009.

[20] P. Kalnis, G. Ghinita, K. Mouratidis, and D. Papadias. Preventing location-based identity inference on anonymous spatial queries. *IEEE Transactions on Knowledge and Data Engineering*, 19(12), 2007.

[21] J. Krumm. Inference attacks on location tracks. In *Pervasive*, 2007.

[22] K. LeFevre, D. DeWitt, and R. Ramakrishnan. Mondrian multidimensional k-anoymity. In *ICDE*, 2006.

[23] A. Machanvajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. In *ICDE*, 2006.

[24] M. Mokbel, C. Chow, and W. Aref. The new casper: Query processing for location services without compromising privacy. In *VLDB*, 2006.

[25] M. E. Nergiz, M. Atzori, and Y. Saygin. Toward trajectory anonymization: A generalization-based approach. In *2nd SIGSPATIAL ACM GIS International Workshop on Security and Privacy in GIS and LBS*, 2008.

[26] P. Samarati. Protecting respondants' identities in microdata release. *IEEE Transactions on Knowledge and Data Engineering*, 13(6), 2001.

[27] L. Sweeney. K-anonymity: A model for protecting privacy. *International Journal on Uncertainty, Fuzziness, and Knowledge-Based Systems*, 10(5), 2002.

[28] Y.F. Tao, C. Faloutsos, D. Papadias, and B. Liu. Prediction and indexing of moving objects with unknown motion patterns. In *SIGMOD*, 2004.

[29] M. Terrovitis and N. Mamoulis. Privacy preservation in the publication of trajectories. In *Proceedings of the International Conference on Mobile Data Management*, 2008.

[30] T. Wang and L. Liu. Privacy-aware mobile services over road networks. In *VLDB*, 2009.

[31] X. Xiao and Y. Tao. Anatomy: Simple and effective privacy preservation. In *VLDB*, 2006.

[32] X. Xiao and Y. Tao. M-invariance: Towards privacy preserving re-publication of dynamic datasets. In *SIGMOD*, 2007.

[33] R. Yarovoy, F. Bonchi, L. Lakshmanan, and W. Wang. Anonymizing moving objects: How to hide a mob in a crowd? In *EDBT*, 2009.