An Online Framework for Publishing Dynamic Privacy-Sensitive Location Traces

University of Michigan Technical Report # CSE-TR-554-09

July 12, 2009

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 considers the problem of protecting individual anonymity when continuously publishing a stream of location trace information collected from a population of users. Fundamentally, the key challenge that arises in this setting is the presence of evolving data, and in particular, data that evolves in semi-predictable ways.

The main contribution of this paper is the first comprehensive formal framework for reasoning about privacy in this setting. Through careful analysis of the expected threat, we articulate a new privacy principle called *temporal unlinkability*. Then, by incorporating a model of user motion, we are able to quantify the risk of privacy violations probabilistically. Within this framework, we develop a simple initial set of algorithms for continuous publishing, and we demonstrate the feasibility of the approach using both real and synthetic location data.

1. INTRODUCTION

Streaming location data from sensors and GPS devices is driving a broad new class of applications. In this paper, we consider an organization that collects and distributes (online and in real-time) a stream of location trace information from a population of users. For example, cellular phone providers and car companies can track users' locations using the GPS devices attached to modern phones and cars. Often, there is a compelling reason for the collecting organization to share or sell these location traces to a third party. For example, GPS traces provide valuable real-time traffic information, and they can be used to price outdoor advertisements (e.g., billboards) based on viewership. At the same time, there are also concerns for the privacy of the users carrying the GPS devices. For example, a recent study tracking the locations of cell phone users sparked heated controversy [30].

The majority of past work in location-based privacy and anonymity has focused on one-time snapshots, applying techniques like spatial k-anonymity (see [31, 32]) to mask the locations of users at a single point in time [12, 16, 22, 29]. However, we are interested in protecting privacy across time, which poses a distinct challenge because the locations of a particular user at various points in time are highly-correlated with one another. Thus, it is often possible to infer the location of an individual by analyzing historical snapshots, a problem we will refer to as *motion prediction inference*. This is best illustrated with an example. (The full threat model is presented in Section 2.1.)

EXAMPLE 1.1. *Consider the scenario shown in Figure 1. Suppose that three users (Alice, Bob, and Charlie) live in the same neighborhood, and that their locations are being tracked via cell phone GPS devices. At 7:45 AM, all three are at their homes, but in the interest of privacy, the phone company releases a snapshot of their locations containing the following* anonymization group*:* $(\{Alice, Bob, Charlie\}, \{1, 2, 3\})$*, indicating that Alice, Bob, and Charlie are at locations 1, 2, and 3, but eliminating the association between user and location. However, through access to* auxiliary information*, for example the telephone book, an adversary can determine the locations of some of the individuals. For example, location 1 is Alice's home.*

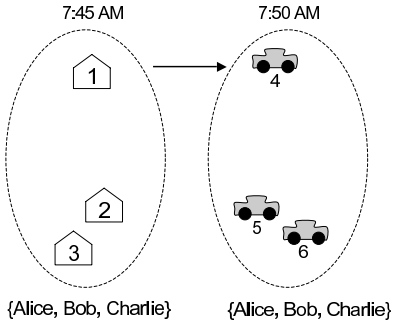


Figure 1: Example of De-Identified Location Data

Suppose now that the users leave their homes for work, and at 7:50 AM, the phone company releases another snapshot containing $(\{Alice, Bob, Charlie\}, \{4, 5, 6\})$*. By examining locations 4, 5, and 6, the adversary discovers based on his knowledge of motion patterns (e.g., speed limits and traffic) that locations 5 and 6 are too far away from location 1 to have been reached in the intervening 5 minutes. Thus, he can infer that Alice is at location 4.*

1.1 Contributions

In this paper, we describe the first formal framework for reasoning about privacy in online location trace publishing. The framework is highlighted by the following key features:

1. We carefully articulate a threat model, and use this to develop a novel high-level privacy principle (*temporal unlinkability*) that is well-suited to this domain. (Section 2.1)
2. To address the problem of data updates (specifically, the challenge of motion prediction inference), our framework includes support for an extensible ("plug-and-play") motion model that probabilistically describes the movements of users. (Section 3)
3. Using the motion model, we can compute a *breach probability*, which measures the certainty with which an adversary

can violate temporal unlinkability in the presence of motion prediction. (Section 3)

4. Finally, we propose two initial protocols for continuously publishing location traces without causing privacy breaches. (Section 4)

An 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, and are thus unsuitable for online location trace publishing. However, the experiments also indicate that our protocols can be used effectively to mitigate this problem.

2. PRELIMINARIES

2.1 Threat Model & Privacy Principle

Though widely considered important, privacy is a somewhat nebulous concept. So far, there is no single technical definition that is universally accepted for all applications and all kinds of data. However, by carefully examining the application at hand, as well as the potential threats, we arrived at a simple privacy principle, which captures many of the necessary requirements.

When publishing location trace data, we expect to encounter one or more adversaries who have access to some source of external information associating individual users with particular locations at specific points in time. Examples of this kind of *auxiliary information* include the Yellow Pages, employee directories, work schedules, etc. More generally, this auxiliary information may associate users with spatio-temporal paths (e.g., a user's route home from work). However, in our setting, it is unreasonable to assume that the organization collecting and distributing traces has access to all of this auxiliary information.

Throughout this paper, we will consider a finite population of n users, each with a unique identifier $(u_1, ..., u_n)$. For the sake of illustration, consider a strawman protocol in which unique identifiers are replaced with unique pseudonyms $p_1, ..., p_n$. For example, Alice's name might be replaced with a unique hash value. Unfortunately, this simple protocol is insecure. In particular, using the available auxiliary information, an adversary may learn associations between certain pseudonyms and identifiers. Worse, because these associations are consistent across time, the adversary is then able to learn the locations of these users at all other points in time.

To overcome the shortcomings of the strawman, one might consider shifting the pseudonyms across time. For simplicity, we assume that the users' locations are reported to the central repository in discrete time steps, $t_0, t_1, ...$ In this second approach, at each time t_i we would replace user u_j with a unique pseudonym p_j^i, such that the pseudonyms p_j^i for $i = 0, ...$ bear no discernible relationship to one another. Unfortunately, this approach is also ineffective. Using multi-target tracking tools, which implicitly model user motion, it is often still possible to track a particular user across time [17, 23].

Based on these observations, we arrived at the following guiding principle for location trace privacy.[1] Of course, we will not be able to prevent an attacker from determining the location of a particular individual when this information is already available as part of the auxiliary information. However, he should not be able to use this information to identify the user at other points in time.

DEFINITION 1 (PRINCIPLE OF TEMPORAL UNLINKABILITY). *Consider an adversary who is able to correctly associate a user u_j with m sequential pseudonyms, $p_j^i, ..., p_j^{i+m}$. Under reasonable assumptions of inference, the adversary should not be able, with high confidence, to identify the pseudonym p_j^h corresponding to u_j at some other point in time, $h \notin \{i, .., i+m\}$.*

[1]The underlying intuition is similar to that motivating the development of mix-zones [4], which seek to maintain similar properties in a less formal way.

2.2 Cloaking Mechanism

When thinking about privacy-protection protocols, as in security, it is often useful to draw a distinction between the privacy *policy* (i.e., set of formal guarantees) and the *mechanism* used to enforce the policy. The primary focus of this paper is the former; however, out of necessity, we chose to work with a particular mechanism. Numerous cloaking, masking, and sampling mechanisms have been proposed for location-based data. We selected one particular mechanism, which appears to generalize several other proposals. Thus, in this section, we will first describe the mechanism that we selected, and then we will describe why our formal framework for reasoning about privacy still applies in related settings. Extending of our policy and formal framework to other mechanisms is an interesting topic for future research.

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. We assume that the users are traveling in a metric space, and the system publishes location-based data in discrete *epochs*, labeled $t_0, t_1, ...$ During each epoch, a *location snapshot* associates each user with a particular 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 publishing modified location snapshots. Specifically, a *release candidate* is modeled as a set of anonymization groups, each of which contains a non-overlapping set of pseudonyms and a multiset of locations.

DEFINITION 3 (RELEASE CANDIDATE). *A release candidate $D^*(t_j)$ for $D(t_j)$ 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 chose to abstract the idea of release candidates this way for maximum flexibility, and we are careful to note the relationship between this approach and other proposals:

- **Spatial Cloaking** Spatial cloaking techniques have been proposed in which the precise locations of individuals are replaced with regions. We are careful to note that such cloaked representations (e.g., minimum bounding rectangles [12]) can be computed from the anonymization groups in our release candidates, and thus reveal no more information. Further, such cloaking techniques may still be vulnerable to attacks based on motion models. For example, 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 (Sampling)** An alternative approach would eliminate the use of consistent pseudonyms entirely. This can be modeled as a special case of our mechanism, where each $D^*(T_j)$ contains just one anonymization group.

	Domain	Variables	Instances
Pseudonym	$\{p_1, ..., p_n\}$	$P, P_1, ...$	$p_1, p_2, ...$
Epochs	$\{t_0, t_1, ...\}$	$T_0, T_1, ...$	$t_0, t_1, ...$
Locations	some metric space	$L, L_0, ...$	$l_0, l_1, ...$

Table 1: Summary of notation

- **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 time T_j, can be computed from a release candidate $D^*(T_j)$ that contains just one anonymization group.

2.3 Data Quality

When using data cloaking or anonymization, there is often a (formal or informal) tradeoff to be made between the privacy provided by the mechanism and the quality of the resulting data. In the pathological case, we could achieve perfect privacy by releasing no data, but this provides none of the benefits of data sharing.

In the case of our cloaking mechanism (Section 2.2), there are intuitively 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 maximize *publication frequency* by publishing during as many epochs as possible. The relative importance of these two dimensions varies based on the application. While the data quality issue is largely separate from the formal framework we will describe for reasoning about privacy, we will revisit data quality when describing our 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 main components: a *motion model*, and a *breach probability function*.

The motion model is a probabilistic model, used by both the central repository and the adversary to predict the location of a user given her location during adjacent epochs. The motion model is defined as a strictly independent (and replaceable) component of the framework. While it is considered non-standard in security to assume that we know the adversary's motion model, in practice we expect that the publishing organization would develop, tune, and continuously evaluate a motion model that is appropriate for the application domain.

Using the motion model as a black box, we formally characterize the notion of a privacy breach, based on the unlinkability principle described in the last section.

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 time 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, given the location of the user at the preceding h epochs (*forward motion model*), or the following h epochs (*backward motion model*).[2]

For clarity, in the following definitions we use capital letters to denote variables (locations, epochs, and users), and we use lowercase letters to denote instances. The notation is in Table 1.

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:

$$\begin{aligned}\Pr[\text{Loc}(P, T_j) = L_j | \text{Loc}(P, T_{j-1}) = L_{j-1}, ..., \\ \text{Loc}(P, T_{j-h}) = L_{j-h}] \\ = \Pr[\text{Loc}(P, T_j) = L_j | \text{Loc}(P, T_{j-1}) = L_{j-1}, ..., \\ \text{Loc}(P, T_0) = L_0].\end{aligned}$$

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* $1 \leq h \leq j$*:*

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

Finally, if a motion model satisfies the symmetry property, then it can be read forwards and backwards.[3]

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

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

3.2 Sample Linear Motion Model

The motion model is an independent and replaceable component of our framework. Nevertheless, for concreteness, we will describe a simple (forward and backward) linear motion model, which instantiates the more general template for $h = 1$.[4]

The sample motion model is based on simple velocity distribution assumptions. We assume 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 following description uses polar co-ordinates.)

Given that P's location at epoch T_1 is L_1, the possible locations for P at T_2 are illustrated in Figure 2(a) as a sector with angles $[\theta_1, \theta_2]$ and distances $[r_1, r_2]$ where $r_1 = v_1 \cdot (T_2 - T_1), r_2 = v_2 \cdot (T_2 - T_1)$.

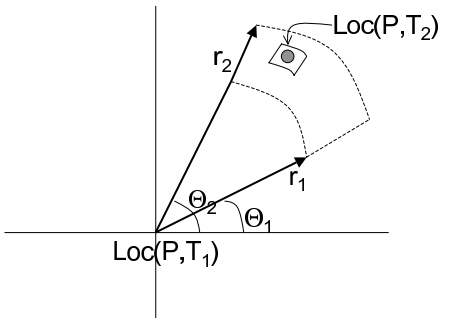
To view this distribution in terms of discrete locations, we impose a polar "grid" on this sector. $\Pr[Loc(P, T_2) = L_2]$ is assigned a probability mass based on the density of the cell containing L_2. If r and θ are independent, with pdfs $f(r)$ and $f(\theta)$, and the boundaries of the cell are given by $r \in [\epsilon_1, \epsilon_2]$ and $\theta \in [\beta_1, \beta_2]$, then the probability mass is obtained by integrating over this region:

$$\int_{\beta_1}^{\beta_2} \int_{\epsilon_1}^{\epsilon_2} f(r) f(\theta) dr d\theta$$

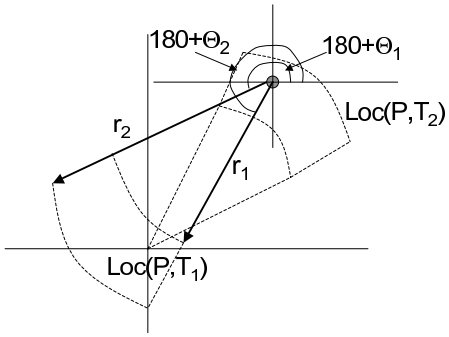
[2]Note that this motion model assumes that the movements of specific users are independent of one another.

[3]Of course, symmetry is a property of the probabilistic *motion model*; it does not imply that any particular user actually repeats his movement backwards.

[4]Hoh et al. implicitly model motion using one prior time step [19]. However, by making the motion model explicit and replaceable, we develop a more general and extensible framework.



(a) Forward



(b) Backward

Figure 2: Sample linear motion model

However, if we assume that the speed and angle are uniformly distributed, then we have

$$\int_{\beta_1}^{\beta_2}\int_{\epsilon_1}^{\epsilon_2}\frac{1}{r_2-r_1}dr\frac{1}{\theta_2-\theta_1}d\theta = \frac{\epsilon_2-\epsilon_1}{r_2-r_1}\cdot\frac{\beta_2-\beta_1}{\theta_2-\theta_1}.$$

These probabilities are, of course, computed under the condition that L_2 is within the ranges $[r_1, r_2]$ and $[\theta_1, \theta_2]$, as shown in Figure 2(a). Otherwise, the probability is 0.

For the reverse motion model, we maintain the same velocity distribution assumptions. Given that P's location at T_2 is L_1, the possible locations for P at T_1 are illustrated in Figure 2(b) as a sector with angles (expressed in degrees) $[\theta_1 + 180, \theta_2 + 180]$ and distances $[r_1, r_2]$. If we compute the probability mass as before, we obtain $\Pr[Loc(P,T_1) = L_1 | Loc(P,T_2) = L_2] = \Pr[Loc(P,T_2) = L_2 | Loc(P,T_1) = L_1]$; thus the simple linear motion model is symmetric.

Of course, there are many alternatives to the linear motion model (e.g., [18, 33, 21]). Many of these models rely on using the previous locations of an object to predict future locations, and generally fit within our framework.

3.3 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*

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

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 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 $C^{(j)}_P$ and $L^{(j)}_P$ denote the sets of pseudonyms and locations, respectively, contained in this anonymization group. Let $M : C^{(j)}_P \rightarrow L^{(j)}_P$ be a one-to-one function mapping pseudonyms to locations. Notice that there are $g = |L^{(j)}_P|!$ 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.

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

EXAMPLE 3.1. *To illustrate, consider the simple example in Figure 3, 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*

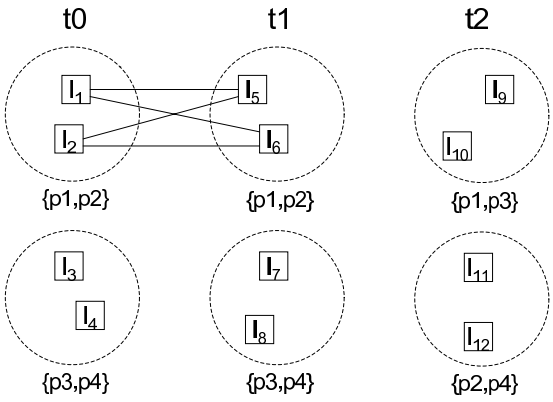


Figure 3: Example of breach probability computation

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

EXAMPLE 3.2. *Consider again the example in Figure 3, and suppose $m = h = 1$. Suppose that the true snapshot at epoch t_0 is $D(t_1) = \{(p_1, l_5), (p_2, l_6), (p_3, l_7), (p_4, l_8)\}$.*

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

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

Then, we can compute the breach probabilities.

$\Pr[Loc(p_1, t_0) = l_1 | D(t_1), D^*(t_0)] = \frac{e_{15}e_{26}}{e_{15}e_{26}+e_{25}e_{16}}$,
$\Pr[Loc(p_2, t_0) = l_2 | D(t_1), D^*(t_0)] = \frac{e_{15}e_{26}}{e_{15}e_{26}+e_{25}e_{16}}$,
$\Pr[Loc(p_1, t_0) = l_2 | D(t_1), D^*(t_0)] = \frac{e_{25}e_{16}}{e_{15}e_{26}+e_{25}e_{16}}$,
$\Pr[Loc(p_2, t_0) = l_1 | D(t_1), D^*(t_0)] = \frac{e_{25}e_{16}}{e_{15}e_{26}+e_{25}e_{16}}$.

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:*[5]

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

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, we turn our attention to developing a protocol for continuously publishing privacy-preserving location traces.

4.1 Checking for Breaches: Brute-Force

The discussion in Section 3.3 suggests an algorithm that can be used to check a release candidate $D^*(T_j)$ for forward privacy breaches, given snapshots $D(T_{j-h}), ..., D(T_{j-1})$. Similarly, there is a related algorithm that checks for backward breaches, given $D(T_{j+1}), ..., D(T_{j+h})$.

Algorithm 1 shows how to check for forward breaches 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. Of course, k is often a small constant, in which case the complexity is $O(n)$. (In the event that this is not the case, we provide some additional optimizations in Section 4.2.)

Algorithm 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:       end for
9:       denom += Pr[M]
10:   end for
11:   for each p ∈ C do
12:      for each ℓ ∈ L do
13:         BP = numer[p][ℓ] / denom
14:         if BP > T then
15:            return true
16:         end if
17:      end for
18:   end for
19: end for
20: 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. While this is acceptable for small fixed k, in this section we describe a fast pruning algorithm. In many cases, the pruning algorithm is 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 necessary computation.

Recall the formula for computing the breach probability in Equation 1. 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 $PR_i = \{\Pr[(c_1, l_i)], \ldots, \Pr[(c_k, l_i)]\}$

[5]This definition is easily adapted to use a different threshold for each user. We omit the details in the interest of simplicity.

	l_1	l_2	l_3
c_1	0.5 (P_1)	0.31 (p_2)	0.19 (p_3)
c_2	0.35 (p_1)	0.45 (P_2)	0.2
c_3	0.4	0.35	0.25 (P_3)

Table 2: Sample PR[(c,l)]

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. Table 2 shows the set of k^2 probabilities generated in Step 1. In Step 2, the maximum and minimum values per location are labeled $P_1, ..., P_3$ and $p_1, ..., p_3$, respectively.*

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 1, 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.

In order to improve these bounds, we make the following observation: In Equation 1, 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.[6] 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 the values $max[1], ..., max[x]$ and $min[1], ..., min[x]$. There is a polynomial-time algorithm for finding these values, but we postpone the full description of the algorithm to Appendix B.
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 example in Table 2, 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

Generally-speaking, the cloaking mechanism described in Section 2.2 gives us two tools to work with in order to guarantee that a published stream of location trace data does not result in a privacy breach. 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. In this section, we explore the space, and with an eye toward practicality, we suggest two promising initial approaches.

4.3.1 General Case

First, consider the general case in which we have the flexibility to vary the size and composition of anonymization groups and to limit the frequency of publication. 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 must have future snapshots $D(T_{j+1}), ..., D(T_{j+h})$ in hand. A simple solution is to delay publishing for h subsequent epochs, after which $D^*(T_j)$ is easily checked for (forward and backward) breaches.

In the general case, we also have the flexibility to choose a new set of anonymization groups at each epoch T_j. One might view this as an optimization problem: Given an objective function, find the *best* release candidate that does not cause a breach. However, like many related problems, this problem is apparently computationally difficult. For the sake of illustration, consider a simple objective function that minimizes the maximum radius of any anonymization group in $D^*(T_j)$.[7] In this case, the 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*

[6] Notice that this is still a conservative (loose) estimate because we do not enforce the requirement that the mapping of objects to locations be a function. (I.e., in the resulting estimate, a single user can be assigned to multiple locations.) For example, in Table 2, c_1 is assigned to locations l_2 and l_3 in the minimum estimate $p_1 \cdot p_2 \cdot p_3$.

[7] Related objective functions, based on area or volume of resulting clusters, have been used in prior work [2, 3, 10, 13, 20].

1. *$D^*(T_j)$ does not cause a (forward or backward) privacy breach, and*
2. *The objective function $\max_{i=1..B} R(C_i)$ is minimized, where $R(C_i)$ is the radius of C_i.*

THEOREM 1. *Problem 1 is NP-hard.*

PROOF. The proof is provided in Appendix C. □

In light of the apparent complexity of the optimization problem, and the combinatorial nature of the checking algorithms described in Sections 4.1 and 4.2, it does not seem that solving Problem 1 is a reasonable goal.

On the upside, however, there is a practical compromise solution that simply leverages an existing (heuristic or approximation) algorithm for k-anonymity (e.g., [2, 3, 12, 24]) to generate a release candidate $D^*(T_j)$. If $D^*(T_j)$ does not cause a breach, it is published. Otherwise, we simply do not publish during epoch T_j.

4.3.2 Durable Anonymization Groups

While the general case allows us to vary the composition of anonymization groups and the frequency of publication, in this section, we consider a restricted protocol 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 $C_i(t_i) = ... = C_i(t_j)$.

Publication protocols involving only durable clusters have several appealing properties. In particular, while the general publishing approach requires that we check for forward and backward breaches, this is not necessary if we require durable groups and the motion model is symmetric. Not checking for backward breaches has several advantages: (1) It reduces the checking time by half, and (2) There is no need to delay publishing for h epochs as in the general case.[8]

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.*

PROOF. The proof is provided in Appendix C. □

EXAMPLE 4.3. *Again, consider the example in Figure 3, 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, an initial burn-in"period can be used to discover *flocks* of users with similar motion patterns.[9] Also, note that these anonymization groups do *not* need to be durable in perpetuity. It is possible to occasionally re-cluster the users, temporarily reverting to the general case (forward and backward checks) when this happens.

[8]The other more subtle advantage of the durable clusters approach is that it allows us to use a variable epoch numbering system, in which we only assign an epoch number at points in time when a release candidate is actually published. On the other hand, in the non-durable case, when we also check for backward breaches, it is assumed that some release candidate will be published in future epochs $T_{j+1}, \ldots, T_{j+h}$, and that these future times will each receive an epoch number.

[9]We could use an existing trajectory clustering algorithm (e.g., [34]) to find the flocks.

5. EXPERIMENTAL RESULTS

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

- We use our framework to analyze the occurrence of the motion prediction inference problem. Much prior work on location privacy has focused on applying k-anonymous cloaking to protect the locations of users at a single point in time [12, 13, 16, 22, 29]. However, to the best of our knowledge, all of these tools are vulnerable to motion prediction inference (Section 1). 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

As experimental data we used one synthetic and one real dataset. These datasets are described below.

■ **Synthetic data** The Network-based Generator of Moving Objects (NG-MO) [6] simulates points moving in a road network. We use this generator to produce points moving on the road network of the San Francisco (SF) Bay Area (1735800×1372400 ($unit^2$)). In the simulation, mobile nodes pick their speed from the range [1419, 5207570] (speed units), and pick their angles from the range [0, 180] degrees. We generated a dataset with 1200 trajectories.

■ **Real data** We also use real GPS traces from a study conducted by a Transportation Research Institute in our university. (This data is not publicly available yet and we omit the details of the organization to adhere to the rules of double-blind reviewing.) Actual GPS units were mounted on volunteers' cars, and data was collected from the cars as the drivers went about their daily activities. 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 need the range of speed and angles. These measures were computed from the trajectory dataset. The speeds ranged from 0 to 170 km/hr, and the angles ranged from 0 to 180 degrees. For the motion model, we assume a uniform distribution over both of these ranges, as described in Section 3.2.

5.2 Implementation and Experimental Setup

To carry out the evaluation we implemented the following two protocols for data publication:

1. **Durable Clusters** In the first protocol, the data is initially clustered into anonymization groups at epoch 1 using the clustering method proposed 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)
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.

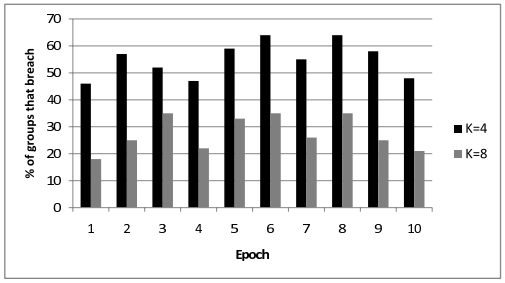


Figure 4: % of groups that exceed the breach threshold T, k-Condense method, SF Data, k=4 and k=8, T=25%

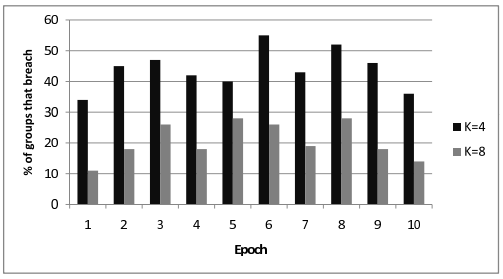


Figure 5: % of groups that exceed the breach threshold T, r-Gather method, SF Data, k=4 and k=8, T=25%

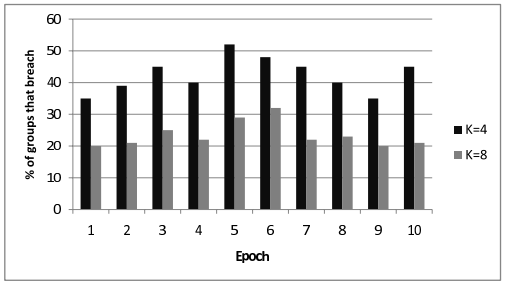


Figure 6: % of groups that exceed the breach threshold T, k-Condense method, GPS Data, k=4 and k=8, T=25%

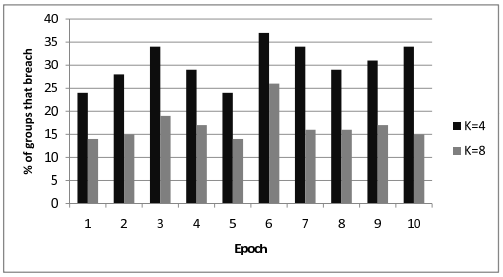


Figure 7: % of groups that exceed the breach threshold T, r-Gather method, GPS Data, k=4 and k=8, T=25%

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, as described in Section 3.2. We vary the value of k from 4 to 12 and the breach probability threshold T from 25% to 75%. For the pruning experiments, we use the improved pruning algorithm described in Section 4.2.2 with x set to k. In the following, we only report selected results for $k = 4, 8, 12$ and $T = 25\%, 50\%, 75\%$, and we report results for the first ten epochs of the evolving datasets.

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.

In the experiments described in this section, 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.)

5.3.1 Static k-Anonymization on SF Dataset

The results for the NG-MO San Francisco dataset are presented in Figure 4, 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 the threshold is exceeded in all cases!

By analyzing Figure 4, we also observe that the number of 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 5). 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.

5.3.2 Static k-Anonymization on GPS Dataset

In addition to the synthetic SF dataset, we performed the same experiment using the real GPS data. These results are shown in Figures 6 and 7. Like the results for the SF data, the breach probability exceeds the threshold at each epoch.

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 for the GPS data with $k = 4, 8$ and $T = 25\%$ and 75% are shown in Figures 8 through 11. In all cases, the time to check the breach probabilities is significantly smaller with the durable clusters than with the non-durable clusters, as expected.

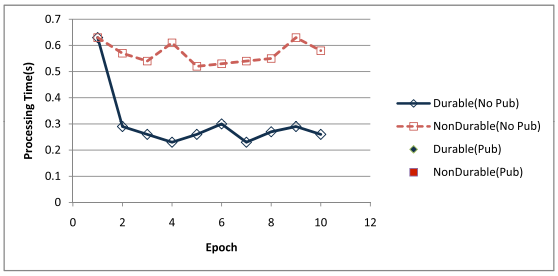


Figure 8: Durability Test, GPS Data, k=4, T=25%

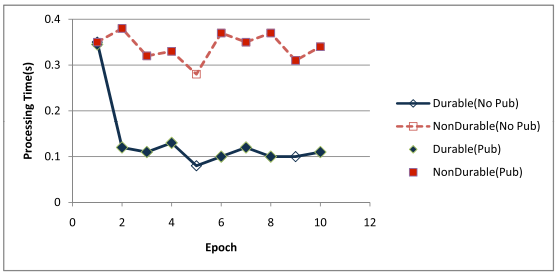


Figure 9: Durability Test, GPS Data, k=4, T=75%

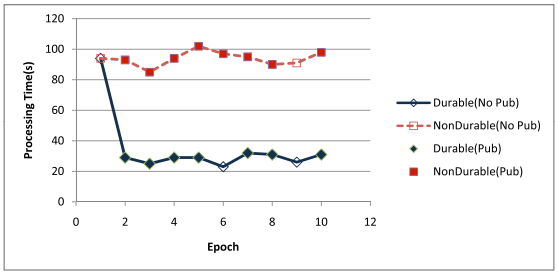


Figure 10: Durability Test, GPS Data, k=12, T=25%

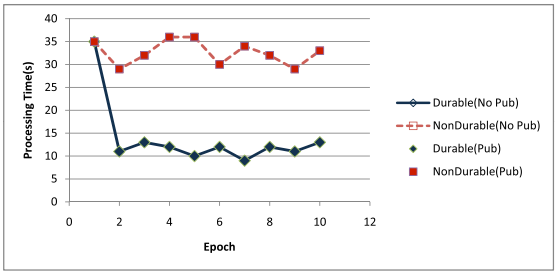


Figure 11: Durability Test, GPS Data, k=12, T=75%

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.

From Figures 8 and 9, we observe that we can't publish any release candidate for $T = 25\%$ and $k = 4$. However, with $T = 75\%$ we can publish a release candidate at nearly every epoch. Also, notice that with a larger threshold T the computation time decreases; this is because the pruning process is more effective with a larger threshold.

Next, we examine the effects of increasing k. Figures 10 and 11 show the results with $k = 12$. Notice that increasing k allows more release candidates to be published, but 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.

We observed similar results using the SF data set, as shown in Figures 12- 15.

5.5 Efficiency and Effectiveness of Pruning

In the final set of experiments, we evaluate the effectiveness of the pruning method described in Section 4.2. In the interest of conciseness, we will only present results for $k = 8$ and $T = 50\%$.

The results are shown in Figure 16 and Figure 17 for the SF and GPS datasets, respectively. From these figures, 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 16 and Figure 17, 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 27 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.

We also ran the same tests for $T = 25\%$ and $T = 75\%$ respectively for $k = 8$. For conciseness, we omit the figures and only briefly describe these results. We found that for $T = 25\%$, both durable and non-durable clustering with pruning consumes more processing time than $T = 50\%$ (mainly since pruning is less effective for lower T). In addition, for $T = 25\%$ the publishing frequency is smaller compared to the case of $T = 50\%$. Compared to the case when $T = 25\%$, when $T = 75\%$ less time was spent on processing (due to more effective pruning), and the publishing frequency is higher because of a more relaxed threshold.

We also explored the effect of pruning when changing the group size k. We set $k = 4$ and $k = 12$, with $T = 50\%$. Again, we only summarize these results. These results again indicate that smaller k reduces the processing much faster with pruning.

From these experiments, and those described in the previous section, we also observed that an increase in the value of threshold T is more likely to lead to successful publication than an increase in k. For example, the clusters generated with $k = 8$, $T = 75\%$ can be published more frequently than the clusters generated with $k = 12, T = 50\%$. Also, in Figures 9 and 10, observe that clusters generated with $k = 4, T = 75\%$ have more publications than those generated with $k = 12, T = 25\%$.

6. RELATED WORK

Problems of privacy and anonymity in location-aware applications have drawn considerable recent interest. It comes as no surprise that several recent studies involving real GPS traces have revealed that simply removing individual identifiers (e.g., names) is not sufficient to guarantee anonymity [17, 23], particularly due to the threat of motion prediction and location-tracking. In this section, we give a brief overview of the past research that is most related to the tools proposed in this paper.

Mix-Zones: The principle of temporal unlinkability (Section 2.1) is similar in spirit to the motivation for *mix-zones*, which were proposed by Beresford and Stajano [4]. Mix-zones sought, informally, to prevent an adversary from following the trajectory of any user for extended periods of time. This was accomplished by suppressing trace information while users passed through crowded areas (mix-zones), the intuition being that this confuses an attacker attempting to track any particular user. While the intuition is similar, this work provided no formal guarantees about an attacker's ability to track

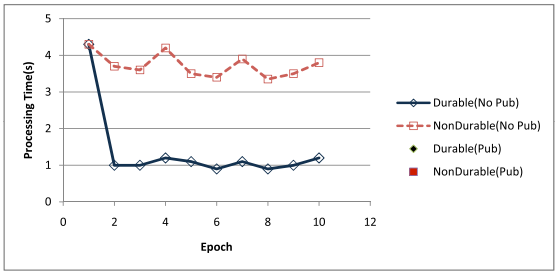


Figure 12: Durability Test, SF Data, k=4, T=25%

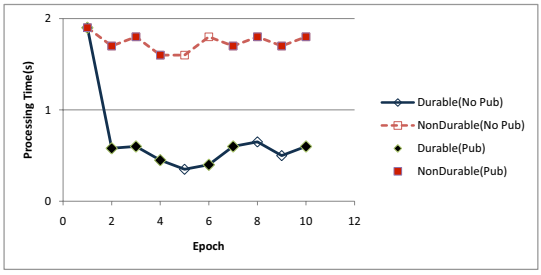


Figure 13: Durability Test, SF Data, k=4, T=75%

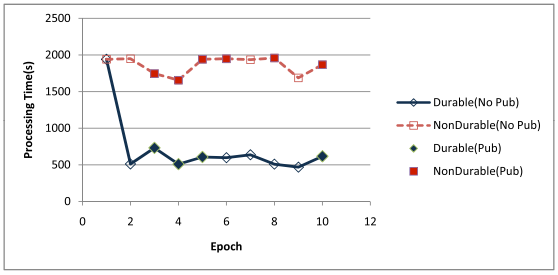


Figure 14: Durability Test, SF Data, k=12, T=25%

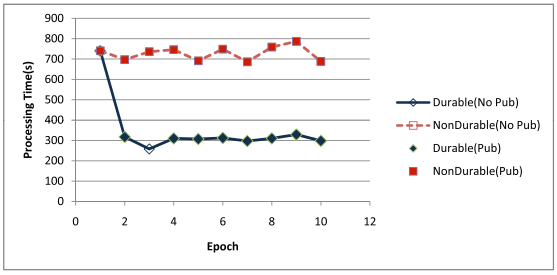


Figure 15: Durability Test, SF Data, k=12, T=75%

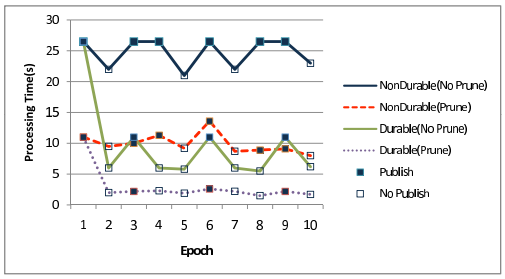


Figure 16: Time Comparisons, SF Data, k=8, T=50%

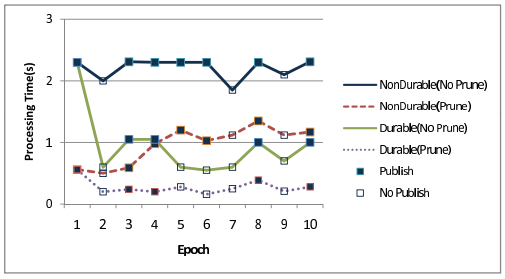


Figure 17: Time Comparisons, GPS Data, k=8, T=50%

individuals.

Uncertainty-Aware Path Cloaking: The uncertainty-aware path cloaking approach proposed by Hoh et al. [19] is also similar in spirit to our work. Mechanistically, they propose to limit the frequency with which individual users' locations are published. This publication approach is used as a way to prevent an attacker from correctly following a user across time, as described by the ideas of *tracking uncertainty* and *time to confusion*.

Our work improves upon this work in three key ways: First, the model developed by Hoh et al. [19] implicitly predicts user motion based on speed and one step of history. By making the motion model explicit and replaceable, we provide a more general and extensible framework, which is, for example, able to capture directionality, in addition to speed. Second, our framework admits privacy-preserving publication mechanisms based on both limiting publication frequency, as well as clustering / cloaking. Third, and most importantly, we observe a flaw in their formulation that can lead to unanticipated privacy breaches. In particular, their definition of tracking uncertainty treats all users independently, which is incorrect for finite user populations (e.g., if we are collecting GPS traces from a fixed set of 200 cars). An extended description of this observation is provided in Appendix A.

Spatial k-Anonymity for LBS: Many have suggested applying spatial variations of k-anonymity to protect private location data. This has been used most often in location-based services (LSB), where users request services based on their locations (e.g., find the nearest gas station), but they do not necessarily want to disclose their precise locations (or queries) to the service provider. A common solution is to replace the user's precise location with a spatial *cloaking region* containing at least $k-1$ other users [12, 13, 16, 22, 29]. Little of the work in location-based services has considered the possibility of tracking users across multiple requests. Notable exceptions include Bettini et al. [5] and Chow et al. [9]. However, neither of these works provide formal privacy guarantees against motion prediction inference.

Recently, Gkoulalas-Divanis et al. proposed anonymizing LBS requests using frequent trajectories [15]. 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 to the problem of online trajectory publishing, where location updates are published frequently and in real-time.

k-Anonymous Trajectories: Recent work has also developed variations of k-anonymity that are used *offline* to anonymize databases of *fully-specified* trajectories [1, 34]. However, after publishing an anonymized database of trajectories using the proposed techniques, it is unclear whether we would be able to publish future location information for the same users without causing a privacy breach. Thus, this work does not naturally extend to the the online

publishing problem considered in this paper. Xu and Cai also seek to anonymize fully-specified trajectories using historical trajectory information [37], but this approach and threat model essentially eliminate time as an identifiable attribute.

Other Related Work: In other related work, Machanavajjhala et al. described a synthetic-data approach for masking commuter (origin, destination) data [27]. This approach provides strong privacy guarantees based on the idea of differential privacy [11], but it is designed for one-time publishing of location data, rather than for an application tracking mobile users.

Privacy and anonymity have also been studied extensively for publishing generic non-aggregate personal data. This work has included approaches to probabilistically modeling disclosure by masked (generalized) data [8, 26, 28, 35], as well as a variety of generalization mechanisms based on spatial partitioning, spatial indexing, and clustering [2, 3, 10, 14, 20, 24, 25]. Recently, several techniques have been proposed that aim to extend these static one-time publishing techniques to a dynamic setting, involving incrementally-updated data sets and multiple releases [7, 36].

Finally, though not the primary focus of this paper, considerable research has focused on motion modeling, trajectory prediction, and tracking for mobile users [33, 21, 18].

7. CONCLUSION

In this paper, we developed a novel formal framework for reasoning about anonymity in the context of continuously publishing location traces. Our framework uses a pluggable motion model to predict the movements of a population of users. Based on this motion model, we provide a formal characterization of privacy breach. Our definition is inspired by a common threat, where an adversary has access to auxiliary information associating certain users with particular locations and times. Given that the adversary already knows the location of a user over a series of time steps, we limit the certainty with which he can identify this user at surrounding points in time.

We developed several simple and effective protocols for continuously publishing location traces. Our experimental results on real and synthetic data indicate the feasibility of the approach. The results also confirm the intuition that anonymizing static location snapshots can lead to inference when continuously publishing the locations of mobile users.

There are several interesting opportunities for future work. While the motion model is a fully independent and replaceable component of our framework, in the future we plan to conduct a more extensive evaluation and comparison of various motion models. In addition, for cases where reliable motion models are unavailable, we plan to search for classes of “conservative" motion models, which reliably do not underestimate (but perhaps overestimate) the breach probabilities.

More broadly, we have every reason to believe that the problems of data evolution and prediction are not limited to GPS data. For example, consider a longitudinal social science study that tracks a set of study participants over a period of years, and in which it is important to publish new findings every year. Just as the location of an individual at 7 AM is highly correlated with that individual’s location at 7:05, certain personal attributes (e.g., age, residence, personal habits) tend to vary in predictable ways over time. For example, age progresses linearly, and residence changes probabilistically, based on the observation that most people do not move far from one year to the next. Future work should investigate applying the same principles (temporal unlinkability and socio-economic “motion" prediction) to other domains, such as longitudinal data, that require online publication of evolving data.

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] B. Chen, K. LeFevre, and R. Ramakrishnan. Privacyskyline: Privacy with multidimensional adversarial knowledge. In *VLDB*, 2007.

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

[10] J. Domingo-Ferrer and J.M. Mateo-Sanz. Practical data-oriented microaggregation for statistical disclosure control. *IEEE Transactions on Knowledge and Data Engineering*, 4(1), 2002.

[11] C. Dwork. Differential privacy. In *ICALP*, 2006.

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

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

[14] G. Ghinita, P. Karras, P. Kalnis, and N. Mamoulis. Fast data anonymization with low information loss. In *VLDB*, 2007.

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

[16] 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.

[17] 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.

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

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

[20] T. Iwuchukwu and J. Naughton. K-anonymization as spatial indexing: Toward scalable and incremental anonymization. In *VLDB*, 2007.

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

[22] 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.

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

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

[25] K. LeFevre, D. DeWitt, and R. Ramakrishnan. Workload-aware anonymization. In *SIGKDD*, 2006.

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

[27] A. Machanvajjhala, D. Kifer, J. Abowd, J. Gehrke, and L. Vilhuber. Privacy: Theory meets practice on the map. In *Proceedings of the IEEE International Conference on Data Engineering*, 2008.

[28] D. Martin, D. Kifer, A. Machanvajjhala, J. Gehrke, and J. Halpern. Worst-case background knowledge in privacy. In *ICDE*, 2007.

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

[30] Associated Press. Study secretly tracks cell phone users outside US. June 2008.

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

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

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

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

[35] X. Xiao and Y. Tao. Personalized privacy preservation. In *SIGMOD*, 2006.

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

[37] T. Xu and Y. Cai. Exploring historical location data for anonymity preservation in location-based services. In *INFOCOM*, 2008.

APPENDIX

A. FLAW IN UNCERTAINTY-BASED APPROACH

Recent work by Hoh et al. [19] also considered incrementally publishing location samples produced by a population of users carrying GPS devices. At an intuitive level, the goals of this work are similar to ours. They propose limiting the frequency with which user locations are published ("sampled") in order to prevent an attacker from correctly tracking a particular user across time. (Their mechanism is slightly different from ours. They consider sampling on a per-user basis, and they do not consider the possibility of clustering users into anonymization groups.)

Though stated informally, the privacy goals of Hoh et al. [19] are nevertheless similar to temporal unlinkability. However, they formulate a different privacy requirement using *mean time to confusion (MTTC)* as the privacy metric. In the following, we observe that their formulation contains at least one important flaw. Specifically, because the population of users is typically finite (e.g., GPS devices attached to a fleet of 200 cars), it is a mistake to view the users independently, and doing so may actually lead to unanticipated breaches of privacy (under both our definition and theirs).

First, we give a brief overview of the privacy definition of Hoh et al. [19]. For ease of exposition, we will use notation consistent with our previous definitions. For simplicity, we will only consider the case in which all locations are published in a series of discrete epochs.

Hoh et al. informally constructed a forward motion model, implicitly based on speed and one step of history. (Empirically, they constructed this model based on a fitted exponential distribution.) It is reasonable to view the constructed motion model as follows:

$$\Pr[Loc(P,T) = L_j | D(T_{prev})]$$

Let $\{L_1, ..., L_n\}$ be the set of location samples at time T. For each user P_i and location L_j, they computed:

$$p(P_i, L_j) = \frac{\Pr[Loc(P,T) = L_j | D(T_{prev})]}{\sum_{j=1}^{n} \Pr[Loc(P,T) = L_j | D(T_{prev})]}$$

Then, for each user, they defined the *tracking uncertainty*:

$$H(P_i) = -\sum_{i=1}^{n} p(P_i, L_j) \log p(P_i, L_j)$$

Finally, they defined the *mean time to confusion* as the mean tracking time during which the uncertainty remains below a confusion threshold T.

The main flaw in this approach is viewing users independently when computing $p(P_i, L_j)$. As a concrete example, suppose that we have only two users p_1 and p_2, and suppose that the locations of these users at time t are l_1 and l_2. Suppose that the above-mentioned motion model yields the following probabilities:

$\Pr[Loc(p_1, t) = l_1 | D(t-1)] = 0.2$
$\Pr[Loc(p_2, t) = l_1 | D(t-1)] = 0.8$
$\Pr[Loc(p_1, t) = l_2 | D(t-1)] = 0.8$
$\Pr[Loc(p_2, t) = l_2 | D(t-1)] = 0.2$

Hoh's methodology would conclude that
$H(p_1) = -0.2 \log 0.2 - 0.8 \log 0.8 = 0.7219$
$H(p_2) = -0.2 \log 0.2 - 0.8 \log 0.8 = 0.7219$

However, notice that this is flawed! If we know that there is a finite set of users, as is generally the case, then there are only two possibilities at time t: (1) p_1 is at l_1 and p_2 is at l_2, or (2) p_2 is at l_1 and p_1 is at l_2.

If we instead re-compute the above probabilities using this observation (and the methodology described in Section 3.3), we find

$\Pr[Loc(p_1, t) = l_1 | D(t-1), D^*(t)] = 0.0588$
$\Pr[Loc(p_2, t) = l_1 | D(t-1), D^*(t)] = 0.9412$
$\Pr[Loc(p_1, t) = l_2 | D(t-1), D^*(t)] = 0.9412$
$\Pr[Loc(p_2, t) = l_2 | D(t-1), D^*(t)] = 0.0588.$

Further, we could recompute the entropy-based uncertainty values from Hoh et al. [19], obtaining the following significantly lower (i.e., less anonymous) values:

$H(p_1) = -0.0588 \log 0.0588 - 0.9412 \log 0.9412 = 0.3226$
$H(p_2) = -0.0588 \log 0.0588 - 0.9412 \log 0.9412 = 0.3226.$

It is interesting and important to observe that computing these possible combinations (mappings from the set of users to the set of locations) is the main source of computational complexity in our privacy checking algorithms (Sections 4.1 and 4.2). While Hoh et al. present a more computationally efficient publication scheme, it neglects to capture this important case, and thus is vulnerable to unanticipated attacks.

B. FINDING MAX[X] AND MIN[X]

Here, we describe an algorithm to find $max[1], ..., max[x]$, which is critical to the optimized pruning algorithm described in Section 4.2.2. (The algorithm for finding $min[1], ..., min[x]$ is analogous.)

The algorithm maintains three main data structures: (In the following, recall that k is the size of the anonymization group.)

- Each set PR_i is sorted into descending order. In the following, we will use the notation $(j_1, ..., j_k)$ to refer to a sequence of positions in the sorted lists $PR_1, ..., PR_k$. For example, if $k = 3$, then $(2, 1, 3)$ refers to the second entry in PR_1, the first entry in PR_2, and the third entry in PR_3.
- The algorithm keeps two sets of k cursors. Initially, $curr_1 = ... = curr_k = 1$ and $next_1 = ... = next_k = 2$.
- Finally, the algorithm maintains a set of $(candidate, value)$ pairs, where *candidate* is a sequence of entry positions, and *value* is the result of multiplying the constituent probability entries. These pairs are stored in a heap, which is sorted according to $value$.

Pseudocode is given in Algorithm 2. Notice that the cursors are used to generate new candidates. When the value of some cursor $next_i$ increases, then candidates are generated by matching $next_i$ with all natural numbers $(c_1, \ldots, c_{i-1}, c_{i+1}, \ldots, c_k)$ such that $c_1 \leq curr_1, \ldots, c_{i-1} \leq curr_{i-1}, c_{i+1} \leq curr_{i+1}, \ldots, c_k \leq curr_k$.

Algorithm 2 $max[x]$ Algorithm

1: Input: $PR_1, \ldots, PR_k$
2: Output: $max[1], \ldots, max[x]$
3: Sort each list $PR_1, \ldots, PR_k$ in descending order
4: $max[1]$ = value of entry $(1, ..., 1)$
5: Initialize $curr_1 = ... = curr_k = 1$
6: Initialize $next_1 = ... = next_k = 2$
7: Generate candidates using $curr_1, \ldots, curr_k$ and $next_1, \ldots, next_k$
8: Insert new candidates into heap H
9: **while** H is not empty and fewer than x values found **do**
10: Remove the next max entry $(j_1, \ldots, j_k)$ from the heap
11: **for** $i = 1$ to k **do**
12: **if** $j_i > curr_i$ **then**
13: $curr_i = j_i$
14: $next_i = j_i + 1$
15: **end if**
16: **end for**
17: Generate new candidates using $curr_1, \ldots, curr_k$ and $next_1, \ldots, next_k$
18: Insert new candidates into H
19: **end while**
20: Return $max[1], \ldots, max[x]$

A simple complexity analysis of the worst-case maximum number of candidates picked while finding $max[x]$ ($x \leq k$) is as follows. Suppose that the positions examined are uniformly distributed in the same c lists ($c < k$), where each list contains x/c checked positions. The complexity of checking all positional combinations is $(x/c)^c$. The worst-case complexity is derived by maximizing this value for $1 \leq c \leq x - 1$. Also, notice that when $c = 1$ or $c = x - 1$, the algorithm's complexity is $O(x)$.

C. PROOFS

Proof of Theorem 1: The proof is a straightforward reduction from r-Gather, which was described in [3], and shown to be NP-hard. The problem is to cluster n points in a metric space into a set of clusters, such that each cluster has at least r points, while minimizing the maximum radius among the clusters.

For any instance of the r-Gather problem, we construct (in polynomial time) an equivalent instance of Problem 1 using an appropriately unrestrictive motion model. For example, if we consider the linear motion model described in Section 3.2 and metric space $\mathbb{R}^2$, then (1) Set the speed distribution $[v_1, v_2]$ to $[0, v_{max}]$, where v_{max} is large enough so that r_2 covers the whole data set; (2) Set angle distribution $[\theta_1, \theta_2]$ to $[0, 360]$ (degrees); and (3) Set breach threshold $T = 1/r$, where r is the parameter of r-Gather.

Conditions (1) and (2) guarantee that every point falls in the region predicted by the motion model and has equal probability. Thus, the breach probability just depends on the number of points in each cluster. Condition (3) guarantees that for any point in a cluster, its breach probability is greater than the threshold T only if the cluster contains at least $1/T$ points. Thus, this instance of Problem 1 is equivalent to the r-Gather problem.

Proof of Theorem 2: The proof is straightforward. Notice that, if all clusters are durable and the motion model is symmetric, then the set of forward breach probabilities at T_j is the same as the set of backward breach probabilities at time T_{j-h}. If the publishing protocol guarantees that there is never a forward breach (i.e., all forward breach probabilities are $\leq T$), then there will never be a backward breach.