2011 IEEE Symposium on Security and Privacy

Quantifying Location Privacy

Reza Shokri, George Theodorakopoulos, Jean-Yves Le Boudec, and Jean-Pierre Hubaux
LCA, EPFL, Lausanne, Switzerland
firstname.lastname@epfl.ch

***Abstract*—It is a well-known fact that the progress of personal communication devices leads to serious concerns about privacy in general, and location privacy in particular. As a response to these issues, a number of Location-Privacy Protection Mechanisms (LPPMs) have been proposed during the last decade. However, their assessment and comparison remains problematic because of the absence of a systematic method to quantify them. In particular, the assumptions about the attacker's model tend to be incomplete, with the risk of a possibly wrong estimation of the users' location privacy.**

In this paper, we address these issues by providing a formal framework for the analysis of LPPMs; it captures, in particular, the prior information that might be available to the attacker, and various attacks that he can perform. The privacy of users and the success of the adversary in his location-inference attacks are two sides of the same coin. We revise location privacy by giving a simple, yet comprehensive, model to formulate all types of location-information disclosure attacks. Thus, by formalizing the adversary's performance, we propose and justify the right metric to quantify location privacy. We clarify the difference between three aspects of the adversary's inference attacks, namely their *accuracy*, *certainty*, and *correctness*. We show that correctness determines the privacy of users. In other words, the expected estimation error of the adversary is the metric of users' location privacy. We rely on well-established statistical methods to formalize and implement the attacks in a tool: the *Location-Privacy Meter* that measures the location privacy of mobile users, given various LPPMs. In addition to evaluating some example LPPMs, by using our tool, we assess the appropriateness of some popular metrics for location privacy: entropy and k-anonymity. The results show a lack of satisfactory correlation between these two metrics and the success of the adversary in inferring the users' actual locations.

***Keywords*-Location Privacy; Evaluation Framework; Location Traces; Quantifying Metric; Location-Privacy Meter**

I. Introduction

Most people are now equipped with smart phones with many sophisticated sensors and actuators closely related to their activities. Each of these devices is usually equipped with high-precision localization capabilities, based for example on a GPS receiver or on triangulation with nearby base stations or access points. In addition, the environment is more and more populated by sensors and smart devices, with which smart phones interact.

The usage of these personal communication devices, although providing convenience to their owners, leaves an almost indelible digital trace of their whereabouts. A trace is not only a set of positions on a map. The contextual information attached to a trace tells much about the individuals' habits, interests, activities, and relationships. It can also reveal their personal or corporate secrets. It can expose the users to unwanted advertisement and location-based spams/scams, cause social reputation or economic damage, and make them victims of blackmail or even physical violence. Additionally, information disclosure breaks the balance of power between the informed entity and the entity about which this information is disclosed.

In the meantime, the tools required to analyze such traces have made tremendous progress: sophisticated data mining algorithms can leverage on fast growing storage and processing power, facilitating, for example, the analysis of multiple databases in parallel. This means that the negative side-effects of insufficient location privacy are becoming more and more threatening.

Users should have the right to control the amount of information (about themselves) that is disclosed and shared with others. This can be achieved in several ways. Users can share a minimum amount of information, or share it only with few trusted entities. Privacy policies can be put in place to force organizations to protect their users' privacy. Finally, systems can be designed in a privacy-conscious manner, so they do not leak information to untrusted entities.

This paper refers to the last ambition. However, our goal here is not to design yet another location privacy protection mechanism (LPPM), but rather to try to make progress on the **quantification** of the performance of an LPPM. This is an important topic, because (i) human beings are notoriously bad estimators of risks (including privacy risks), (ii) it is the only way to make meaningful comparisons between different LPPMs and (iii) the research literature is not yet mature enough on the topic.

Let us develop this last reason. In specific areas, several contributions have been made to quantify privacy, be it for databases [8], for anonymity protocols [3], for anonymization networks [24], or for RFID privacy [25]. Yet, in the field of location privacy, notwithstanding many contributions from different disciplines (such as databases, mobile networks, and ubiquitous computing) for protecting location privacy, the lack of a unified and generic formal framework for specifying protection mechanisms and also for evaluating location privacy is evident. This has led to the divergence of (nevertheless interesting) contributions and, hence, has caused confusion about which mechanisms are

1081-6011/11 $26.00 © 2011 IEEE
DOI 10.1109/SP.2011.18

IEEE computer society

more effective. The adversary model is often not appropriately addressed and formalized, and a good model for the knowledge of the adversary and his possible inference attacks is missing. This can lead to a wrong estimation of the location privacy of mobile users. There is also often confusion between the different dimensions of the adversary's performance in his attacks, notably the accuracy, certainty and correctness of his estimation of the users' traces.

In this paper, leveraging on previous contributions in the field of (location) privacy, we propose a generic theoretical framework for modeling and evaluating location privacy. We make the following contributions.

- We provide a generic model that formalizes the adversary's attacks against private location-information of mobile users. In particular, we rigorously define *tracking* and *localization* attacks on anonymous traces as statistical inference problems.
- We rely on well-established statistical methods to evaluate the performance of such inference attacks. We formalize the adversary's success and we clarify, explain and justify the right metric to quantify location privacy: The adversary's *expected estimation error*.
- We provide a tool: the *Location-Privacy Meter* is developed based on our formal framework and is designed for evaluating the effectiveness of various location-privacy preserving mechanisms.
- We show the inappropriateness of some existing metrics, notably entropy and k-anonymity, for quantifying location privacy.

The paper is organized as follows. In Section II, we provide a detailed description of the framework we propose for the quantification of LPPMs and show how location-privacy threats can be defined and evaluated correctly. In Section III, we introduce an instantiation of the framework into an operational tool: *Location-Privacy Meter*. In Section IV, we show the usage of the tool on evaluating LPPMs and assessing existing location-privacy metrics. We discuss the related work in Section V and conclude in Section VI.

II. The Framework

In this section, we present our framework for location privacy. This allows us to precisely define location privacy and specify its relevant components and entities in various settings and also to evaluate the effectiveness of various location-privacy preserving mechanisms with respect to different attacks. We define a location-privacy framework (system) as a tuple of the following inseparable elements: $\langle \mathcal{U}, \mathcal{A}, \mathrm{LPPM}, \mathcal{O}, \mathrm{ADV}, \mathrm{METRIC} \rangle$, where $\mathcal{U}$ is the set of mobile users, $\mathcal{A}$ represents the set of possible *actual* traces for the users, and LPPM stands for the location-privacy preserving mechanism that acts on the actual traces a (a member of $\mathcal{A}$) and produces the observed traces o (a member of $\mathcal{O}$, which is the set of *observable* traces to an adversary ADV). The adversary ADV is an entity who implements some inference (reconstruction) attacks to infer some information about a having observed o and by relying on his knowledge of the LPPM and of the users' mobility model. The performance of the adversary and his success in recovering the desired information about a is captured by an evaluation metric METRIC. The success of the adversary and the location-privacy of users are two sides of the same coin, which are coupled together using METRIC.

$\mathcal{U}$	Set of mobile users
$\mathcal{R}$	Set of regions that partition the whole area
$\mathcal{T}$	Time period under consideration
$\mathcal{A}$	Set of all possible traces
$\mathcal{O}$	Set of all observable traces
$\mathcal{U}'$	Set of user pseudonyms
$\mathcal{R}'$	Set of location pseudonyms
N	Number of users
M	Number of regions
T	Number of considered time instants
N'	Number of user pseudonyms
M'	Number of location pseudonyms
f	Obfuscation function
g	Anonymization function
a_u	Actual trace of user u
o_u	Obfuscated trace of user u
o_i	Observed trace of a user with pseudonym i
$\mathcal{A}_u$	Set of all possible (actual) traces of user u
$\mathcal{O}_u$	Set of all possible obfuscated traces of user u
$\mathcal{O}_{\sigma(u)}$	Set of all observable traces of user u
P^u	Profile of user u
$\phi(.)$	Attacker's objective
$\mathcal{X}$	Set of values that $\phi(.)$ can take

Table I
NOTATIONS

In the following subsections, we present and specify all the entities and components of our framework and illustrate their inter-relationship. The tool that we have developed according to the framework, *Location-Privacy Meter*, and the theoretical details of the implemented methods will be explained in Section III.

The summary of the notations is presented in Table I. The framework is shown in Figure 1. Throughout the paper, we use bold capital letters to denote random variables, lower case letters to denote realizations of random variables, and script letters to denote sets within which the random variables take values. For example, a random variable $\boldsymbol{X}$ takes values x in $\mathcal{X}$. At times, the members of a set are also sets, but the distinction will be clear from the context.

A. Mobile Users

We consider $\mathcal{U} = \{u_1, u_2, ..., u_N\}$ a set of N mobile users who move within an area that is partitioned into M distinct regions (locations) $\mathcal{R} = \{r_1, r_2, ..., r_M\}$. See Figure 2 for an example of partitioning an area into regions. Time is discrete, and the set of time instants when the users may be observed is $\mathcal{T} = \{1, ..., T\}$. The level of space and time granularity depends on the precision that we want, on the size of the area, and on the total length of the observation

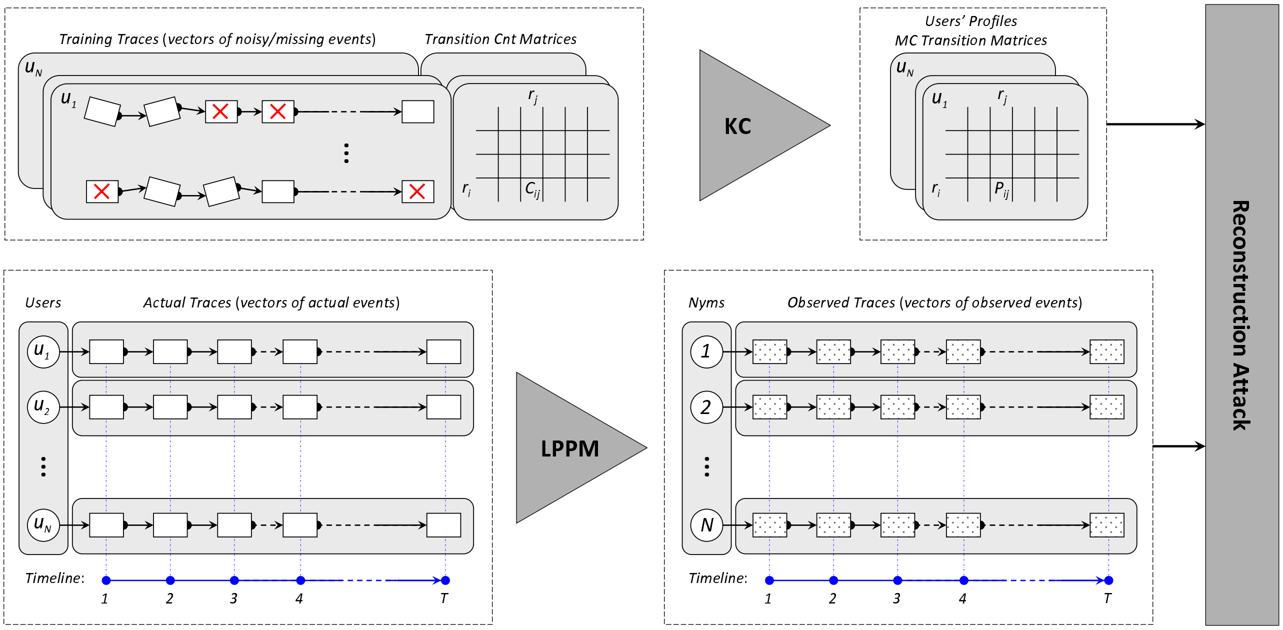


Figure 1. Elements of the proposed location-privacy framework. The users produce actual traces, which are then anonymized and obfuscated by the LPPM to produce anonymous observed traces. The attacker uses a set of training traces to create, via the knowledge construction (KC) mechanism, a mobility profile for each user in the form of a Markov Chain transition probability matrix. Having the user mobility profiles and the observed traces, the adversary tries to reconstruct (infer) the actual traces. The only element of the framework not shown here is the metric that evaluates the success of the adversary's reconstruction attack by comparing the results of the attack with the users' actual traces.

period. For example, regions can be of a city/block size, and two successive time instants can be a day/hour apart.

The spatiotemporal position of users is modeled through events and traces. An *event* is defined as a triplet $\langle u, r, t \rangle$, where $u \in \mathcal{U}$, $r \in \mathcal{R}$, $t \in \mathcal{T}$. A *trace* of user u is a T-size vector of events $a_u = (a_u(1), a_u(2), ..., a_u(T))$. The set of all traces that may belong to user u is denoted by $\mathcal{A}_u$. Notice that, of all the traces in $\mathcal{A}_u$, exactly one is the true trace that user u created in the time period of interest ($t = 1...T$); this one is called the *actual trace* of user u, and its events are called the *actual events* of user u. The set of all possible traces of all users is denoted by $\mathcal{A} = \mathcal{A}_{u_1} \times \mathcal{A}_{u_2} \times \ldots \times \mathcal{A}_{u_N}$; the member of $\mathcal{A}$ that was actually created by the N users is denoted by a, so it is also the set of actual traces.

B. Location-Privacy Preserving Mechanisms

Mobile users share their location with possibly untrusted entities in various location-based services, or may unwillingly expose their location to curious eavesdropping entities through the wireless channel. In addition to these types of sharing, their location traces can be made public for research purposes. In all these scenarios, an adversarial entity can track the users over the observation period, unless their actual traces are properly modified and distorted before being exposed to others, i.e., before becoming observable. The mechanism that performs this modification in order to protect the users' location-privacy is called a Location-Privacy Preserving Mechanism (LPPM).

LPPMs can be implemented in different manners and architectures: *online* vs. *offline*, and *centralized* vs. *distributed*. In the offline manner, all the traces are available to the LPPM, for example in a database, whereas in the online manner, the modification is performed on-the-fly while users visit new regions as time progresses. The modification can be performed in the centralized architecture by a trusted third party (mostly known as the central anonymity server or privacy proxy) as opposed to being done by the users or on their mobile devices in a distributed architecture, where modifications are (mostly) done independently from each other. Next, we abstract away these details and provide a generic model for LPPMs.

A location-privacy preserving mechanism LPPM receives a set of N actual traces, one for each user, and modifies them in two steps. In the obfuscation process, the location of each event is obfuscated, i.e., replaced by a *location pseudonym* in the set $\mathcal{R}' = \{r'_1, ..., r'_{M'}\}$. In the anonymization process, the traces are anonymized, i.e., the user part of each trace is replaced by a *user pseudonym* in the set $\mathcal{U}' = \{u'_1, ..., u'_{N'}\}$. Notice that each region may be obfuscated to a different location pseudonym each time it is encountered, whereas each user is always obfuscated to the same user pseudonym (as in this paper we focus on evaluating users' location-privacy from their location traces). Also, note that the information used by an LPPM to obfuscate an event varies

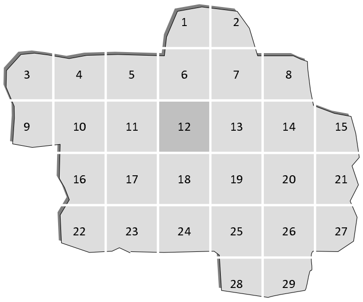


Figure 2. Example of locations and obfuscation. The area within which users move is divided into $M = 29$ regions. Consider user u whose actual location is region r_{12} at a given time t. Different obfuscation methods will replace r_{12} with a different location pseudonym $r' \in \mathcal{R}'$: $r' = \{14\}$ in the *perturbation* method, $r' = \{12, 15, 26\}$ in the *adding dummy regions* method, $r' = \{9, 10, 11, 12, 13, 14, 15\}$ in the *reducing precision* method, and $r' = \emptyset$ in the *location hiding* method.

depending on its type and architecture. For example, an online mechanism in the distributed architecture only looks at the current event for obfuscation, whereas an online mechanism in the centralized architecture can consider all so-far generated events from all of the users at the time of obfuscating the current event.

Formally, an *obfuscated event* is a triplet $\langle u, r', t\rangle$, where $u \in \mathcal{U}$, $r' \in \mathcal{R}'$, and $t \in \mathcal{T}$. As before, an *obfuscated trace* of user u is a T-size vector of obfuscated events $o_u = (o_u(1), o_u(2), ..., o_u(T))$. The set of all possible obfuscated traces of user u is denoted by $\mathcal{O}_u$.

An *obfuscation mechanism* is a function that maps a trace $a_u \in \mathcal{A}_u$ to a random variable $\boldsymbol{O}_u$ that takes values in the set $\mathcal{O}_u$. The probability density function of the output is f:

$$f_{a_u}(o_u) = \Pr\{\boldsymbol{O}_u = o_u | \boldsymbol{A}_u = a_u\}. \tag{1}$$

For the obfuscation, the LPPM covers various methods that reduce the accuracy and/or precision of the events' spatiotemporal information:

- *perturbation (adding noise)*
- *adding dummy regions*
- *reducing precision (merging regions)*
- *location hiding*

These methods probabilistically map a region in an event to a location pseudonym in $\mathcal{R}'$. For these methods, it suffices that the set $\mathcal{R}'$ be the power set of $\mathcal{R}$, i.e., $\mathcal{R}' \equiv \mathcal{P}(\mathcal{R})$. Figure 2 illustrates different obfuscation functions.

An *anonymization mechanism* is a function $\boldsymbol{\Sigma}$ chosen randomly among the functions that map $\mathcal{U}$ to $\mathcal{U}'$. The random function $\boldsymbol{\Sigma}$ is drawn according to a probability density function g:

$$g(\sigma) = \Pr\{\boldsymbol{\Sigma} = \sigma\}. \tag{2}$$

In this paper, we will consider only one anonymization mechanism: random permutation. That is, the set $\mathcal{U}'$ is $\{1, 2, \ldots, N\}$, a permutation of the users is chosen uniformly at random among all $N!$ permutations and each user's pseudonym is his position in the permutation.

A location-privacy preserving mechanism LPPM is a pair (f, g). Given a set of actual traces $\{a_{u_1}, ..., a_{u_N}\}$, the mechanism LPPM applies f to obfuscate each trace, thus generating a set of obfuscated traces $\{o_{u_1}, ..., o_{u_N}\}$, which are instantiations of the random variables $\{\boldsymbol{O}_{u_1}, ..., \boldsymbol{O}_{u_N}\}$. It then applies g on that set, thus generating a set of obfuscated and anonymized traces $\{o_{\sigma(u_1)}, o_{\sigma(u_2)}, ..., o_{\sigma(u_N)}\}$, where $\sigma(\cdot)$ is an instantiation of the random function $\boldsymbol{\Sigma}$.

Now, we can summarize the operation of the LPPM with the following probability distribution function that gives the probability of mapping a set of actual traces $a \in \mathcal{A}$ to a set of observed traces $o \in \mathcal{O} = \mathcal{O}_1 \times \mathcal{O}_2 \times \ldots \times \mathcal{O}_N$:

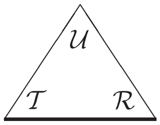
$$\mathrm{LPPM}_a(o) = \Pr\left\{\cap_{i=1}^{N} \boldsymbol{O}_{\boldsymbol{\Sigma}(u_i)} = o_{\sigma(u_i)} | \cap_{i=1}^{N} \boldsymbol{A}_{u_i} = a_{u_i}\right\} \tag{3}$$

Broadly speaking, the aim of the adversary is to invert this mapping: Given o, he tries to reconstruct a.

C. Adversary

In order to evaluate an LPPM accurately, we must model the adversary against whom the protection is placed. Hence, the adversary model is certainly an important, if not the most important, element of a location-privacy framework. An adversary is characterized by his *knowledge* and *attack(s)*. A framework should specify how the adversary obtains and constructs his knowledge, how to model his knowledge and what attacks he performs in order to reconstruct users' location-information.

The adversary is assumed to know the anonymization and obfuscation probability distribution functions f and g. The adversary may also have access to some training traces (possibly noisy or incomplete) of users, and other public information about locations visited by each user, such as their home and workplace. From this information, the adversary constructs a *mobility profile* P^u for each user $u \in \mathcal{U}$. In Section III-B, one way of constructing the adversary's knowledge is explained in detail as part of the *location-privacy meter* tool.



Given the employed LPPM (i.e., f and g), the users' profiles $\{(u, P^u)\}_{u\in\mathcal{U}}$, and the set of observed traces $\{o_1, o_2, ..., o_N\}$ that are produced by the LPPM, the attacker runs an inference (reconstruction) attack and formulates his objective as a question of the $\mathcal{U}-\mathcal{R}-\mathcal{T}$ type. Schematically, in such a question, the adversary specifies a subset of users, a subset of regions and a subset of time instants, and asks for information related to these subsets. If the adversary's objective is to find out the whole sequence (or a partial subsequence) of the events in a user's trace, the attack is called a *tracking* attack. The attacks that target a single event (at a given time instant) in a user's trace, are called *localization* attacks. These two categories of attacks are examples of *presence/absence disclosure* attacks [21]: they infer the relation between users and regions over time. In contrast, if the physical proximity between users is of the adversary's interest, we call the attack a *meeting disclosure* attack (i.e., who meets whom possibly at a given place/time).

An example of a very general tracking attack is the one that aims to recover the actual trace of each user. That is, it targets the whole set of users and the whole set of time instants, and it asks for the most likely trace of each user, or even for the whole probability distribution of traces for each user. More specific objectives can be defined, which lead to all sorts of *presence/absence/meeting disclosure* attacks: Specify a user and a time, and ask for the region where the user was at the specified time; specify a user and a region, and ask for the times when the user was there; specify a subset of regions, and ask for the (number of) users who visited these regions at any time.

In this paper, we provide an algorithm that implements the most general tracking attack; with the results of this attack at hand, many other objectives can be achieved. For some specific types of objectives we design attacks that are much faster and less computationally intensive than the general attack. The details will be explained in Section III-D.

D. Evaluation

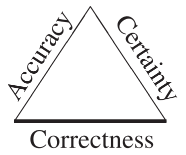
At a high level, the adversary obtains some obfuscated traces o, and, knowing the LPPM and the mobility profiles of the users, he tries to infer some information of interest about the actual traces a. As we have mentioned, the possible objectives of the adversary range from the very general (the traces a themselves) to the specific (the location of a user at a specific time, the number of users at a particular location at a specific time, etc.).

Nevertheless, usually, neither the general nor the specific objectives have a single deterministic answer. The actual traces are generated probabilistically from the mobility profiles, and the observed traces are generated probabilistically by the LPPM. So, there are many traces a that might have produced the observed traces o. The same goes for the more specific objectives: There are many regions where a user might have been at a particular time. The output of the attack can be a probability distribution on the possible outcomes (traces, regions, number of users), the most probable outcome, the expected outcome under the distribution on outcomes (the average number of users), or any function of the actual trace. We call $\phi(\cdot)$ the function that describes the attacker's objective. If its argument is the actual trace a, then its value $\phi(a)$ is the correct answer to the attack. $\mathcal{X}$ is the set of values that $\phi(\cdot)$ can take for a given attack (M regions, N users, M^T traces of one user, etc.).

The probabilistic nature of the attacker's task implies that he cannot obtain the exact value of $\phi(a)$, even if he has an infinite amount of resources. The best he can hope for is to extract all the information about $\phi(a)$ that is contained in the observed traces. The extracted information is in the form of the posterior distribution $\Pr(x|o), x \in \mathcal{X}$, of the possible values of $\phi(a)$ given the observed traces o. We call *uncertainty* the ambiguity of this posterior distribution with respect to finding a unique answer – that unique answer need not be the correct one; see the discussion on correctness later. The uncertainty is maximum, for example, if the output of a localization attack is a uniform distribution on the locations. On the contrary, the uncertainty is zero if the output is a Dirac distribution on one location.

Of course, the attacker does not have infinite resources. Consequently, the result of the attack is only an estimate $\widehat{\Pr}(x|o)$ of the posterior distribution $\Pr(x|o)$. We call *inaccuracy* the discrepancy between the distributions $\widehat{\Pr}(x|o)$ and $\Pr(x|o)$.

Neither the uncertainty metric nor the inaccuracy metric, however, quantify the privacy of the users. What matters for a user is whether the attacker finds the correct answer to his attack, or, alternatively, how close the attacker's output is to the correct answer. Knowing the correct answer, an evaluator of the LPPM calculates a distance (or expected distance) between the output of the attack and the true answer. The choice of distance depends on the attack; we give examples in Section IV. We call this distance the *correctness* of the attack, and we claim that this is the appropriate way to quantify the success of an attack.



It is important that the accuracy and the certainty not be mistaken to be equivalent to the correctness of the attack. Even an attacker with infinite resources will not necessarily find the true answer, as he might have observed only an insufficient number of traces. But he will extract all the information that is contained in the traces, so the accuracy will be maximum. If the accuracy is maximum, and simultaneously the observed traces point to a unique answer – so the certainty is also maximum – the correctness still need not be high. It is possible, for instance, that the user did something out of the ordinary on the day the traces were collected; what he did is still consistent with the observed trace, but as it is not typical for the user it is assigned a low probability/weight in the attack output.

1) Accuracy: We compute the accuracy of each element of the distribution $\widehat{\Pr}(x|o), x \in \mathcal{X}$, separately. That is, we

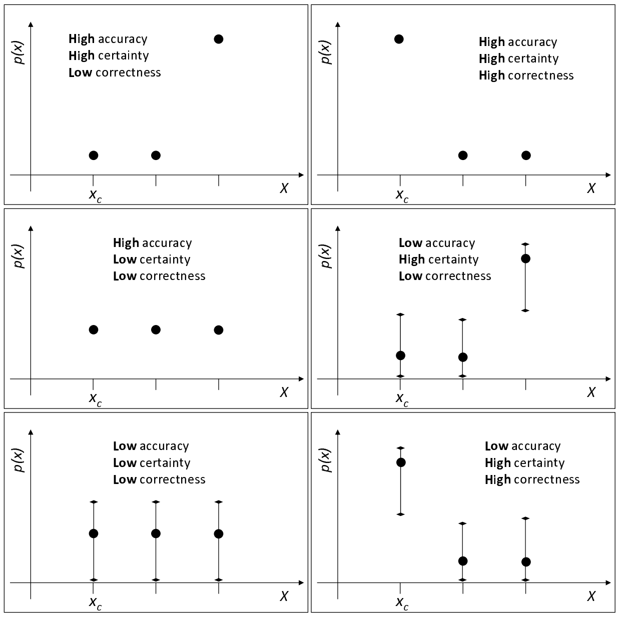


Figure 3. Accuracy, Certainty, and Correctness of the adversary. The adversary is estimating $\widehat{\Pr}(x|o)$ where the true value for x (correct guess) is x_c. In this example, x can get three discrete values. The black dot shows the estimate $\widehat{\Pr}(x|o)$ for different x and the lines show the confidence interval for a given confidence level chosen by the adversary. As it is shown in the figures, the accuracy of the estimation is independent of its certainty and correctness. Moreover, the level of correctness does not convey anything about the level of certainty, and high certainty does not mean high correctness. The only correlation between certainty and correctness is that low certainty *usually* (depending on the size of $\mathcal{X}$ and the distance between its members) implies low correctness.

estimate the posterior probability $\Pr(x|o)$ for each possible value x of $\phi(a)$. We quantify the accuracy with a confidence interval and a confidence level. By definition, the probability that the accurate value of $\Pr(x|o)$ is within the confidence interval is equal to the confidence level.

The extreme case is that the interval is of zero length (i.e., a point) and the confidence level is 1 (i.e., the attacker is absolutely confident that the point estimate is accurate). An attacker using more and more accurate estimation tools could achieve this extreme case, thus making $\widehat{\Pr}(x|o)$ converge to $\Pr(x|o)$. However, achieving such ultimate accuracy might be prohibitively costly. So, the adversary will in general be satisfied with some high enough level of accuracy (i.e., large enough confidence level, and small enough confidence interval). When the accuracy reaches the desired level, or the resources of the adversary are exhausted, the probability $\widehat{\Pr}(x|o)$ with some confidence interval is the estimate of the adversary.

2) Certainty: We quantify the certainty with the entropy of the distribution $\widehat{\Pr}(x|o)$. The entropy shows how uniform vs. concentrated the estimated distribution is and, in consequence, how easy it is to pinpoint a single outcome x out of $\mathcal{X}$. The higher the entropy is, the lower the adversary's certainty is.

$$\hat{H}(x) = \sum_{x} \widehat{\Pr}(x|o) \log \frac{1}{\widehat{\Pr}(x|o)} \qquad (4)$$

3) Correctness: The *correctness* of the attack is quantified using the expected distance between the true outcome $x_c \in \mathcal{X}$ and the estimate based on the $\widehat{\Pr}(x|o)$. In general, if there is a distance $\|\cdot\|$ defined between the members of $\mathcal{X}$, the expected distance can be computed as the following sum, which is the adversary's *expected estimation error*:

$$\sum_{x} \widehat{\Pr}(x|o) \|x - x_c\| \qquad (5)$$

As an example, if the distance is defined to be equal to 0 if and only if $x = x_c$ and to be equal to 1 otherwise, then the incorrectness can be calculated to be $1 - \widehat{\Pr}(x_c|o)$, which is the *probability of error* of the adversary.

The value x_c is what the users want to hide from the adversary. The higher the adversary's correctness is, the lower the privacy of the targeted user(s) is. Hence, ***correctness is the metric that determines the privacy of users***.

In summary, the adversary achieves the maximum accuracy for his estimates $\widehat{\Pr}(x|o)$ that is possible under his resource constraints. He can measure the success of the attack by computing the certainty over the results. However, to measure users' privacy, the evaluator of an LPPM must consider the true value x_c and measure the adversary's correctness. Notice that the adversary does not know the value of x_c, hence he cannot evaluate this aspect of his performance. Figure 3 illustrates through some examples the independence of these three aspects (of the adversary's performance) from each other.

III. Location-Privacy Meter: Implementation of our Framework as a Tool

In this section, we present *Location-Privacy Meter*, a realization of our framework as a tool to measure location privacy. We have developed a modular tool based on the framework presented in Figure 1 and multiple reconstruction (inference) attacks are designed to evaluate the effectiveness of LPPMs with respect to different adversaries. The tool, available online [1], is developed in the C++ language, so it is fast and it can be ported to various platforms. As will be explained further, designers of new LPPMs can easily specify various LPPM functions in our tool in order to compare the users' location privacy in different schemes.

In the following subsections, we explain in detail the specifications of different modules of the tool and also the algorithms that we use in *Location-Privacy Meter*. The evaluation of some LPPMs will be presented in Section IV.

A. Location-Privacy Preserving Mechanisms

In the current implementation of the tool, we have developed two main LPPM obfuscation mechanisms that appear frequently in the literature: *precision reducing (merging regions)* and *location hiding*. The anonymization mechanism is the random permutation.

The *precision reducing* obfuscation mechanism reduces the precision of a region by dropping the low-order bits of the region identifier. If, as in our case, the whole area is divided into a grid pattern of regions, the x and y coordinates of the region can be obfuscated separately. The number of dropped bits determines the level of obfuscation. Let μ_x and μ_y be the number of dropped bits in the x and y coordinates, respectively. This is a deterministic obfuscation in which, for example, $\mu_x = 1$ will map regions r_{12} and r_{13} (in Figure 2) to the same location pseudonym, as they are on the 4th and 5th column of the same row.

In the *location hiding* mechanism, every event is independently eliminated (i.e., its location is replaced by $\emptyset$) with probability λ_h: location hiding level.

An LPPM designer can easily import her LPPM into our tool by specifying the probability density function LPPM (see (3)), or, equivalently, by specifying an anonymization function and an obfuscation function.

B. Knowledge of the Adversary

In this section, we provide a model for constructing the a priori knowledge of the adversary to be used in the various reconstruction attacks. The schema of the knowledge construction (KC) module is illustrated in Figure 1.

The adversary collects various pieces of information about the mobility of the users. In general, such information can be translated to events; perhaps the events can be linked into transitions, i.e., two events of the same user with successive timestamps; perhaps they can be further linked into a partial trace or even a full trace. The quality of these events to the adversary might be varied, e.g., they might contain noise. It is conceivable that the adversary obtains information, such as a user's home address, that is not obviously translatable to an event. Then the adversary can create typical events (or traces) that encode that information, i.e., successive appearances of a user at his home location between the evening and the morning hours.

All this prior mobility information on each user is encoded in one of two ways: Either in the form of some traces, or as a *matrix of transition counts* TC_u. The traces can be noisy or they might be missing some events. The TC_u matrix is of dimension $M \times M$ and its ij entry contains the number of i to j region transitions that u has created and have not been encoded as traces. Any knowledge of the general movement within the regions, i.e., how a typical user moves, that cannot be attributed to a particular user can be incorporated in the TC matrices. In addition to this mobility information on the users, the adversary also considers the general *mobility constraints* of users within regions. For example, it might not be possible to move between two far-away regions in one time instant, or cross a border between two regions because of some physical obstacles.

The adversary makes the assumption that user mobility can be modeled as a Markov Chain on the set of regions $\mathcal{R}$. So, the mobility profile P^u of a user is a transition matrix for that user's Markov Chain. The entry $P^u_{ij}, i, j = 1..M$ of P^u is the probability that u will move to region r_j in the next time slot, given that he is now in region r_i. The objective of the adversary is to construct P^u starting with the prior mobility information (traces and TC_u). The construction is done with Gibbs sampling [20] to find the conditional probability distribution of the entries of the MC matrix, given the prior information. Then, one MC matrix is created out of the distribution, for instance by averaging.

How restrictive is the Markovian assumption on user mobility? For example, if $\mathcal{T}$ represents one full day, users will have different mobility patterns depending on the time of day. A Markov Chain can still model this situation with arbitrary precision at the cost of increasing the number of states. There will be two (or three, or more) interconnected Markov Chains, corresponding to different time periods of the day: morning and evening, or morning, afternoon and evening, or even more fine-grained. Each MC is defined on the set of regions $\mathcal{R}$, so it still has M states, but each has different transition probabilities. The M states of each MC are labeled not only by a region, but also by the period of the day that they correspond to. Finally, there are appropriate transitions from the morning states to the afternoon states, from the afternoon states to the evening states, and so on. So, the model is extensible to more general mobility models, but to keep the presentation simple we assume that $\mathcal{T}$ represents one single time period.

Hereafter, we explain how to create the profile P^u of user u from a training trace TT_u with missing data, and a transition count matrix TC_u. Note that the method that we have implemented considers multiple training traces per user. However, to simplify the presentation we consider only one trace. Moreover, as we are talking about profiling each user separately, we omit the subscript/superscript u.

The ultimate goal is to estimate the parameters of the underlying Markov Chain (i.e., the matrix P). As the training trace TT is incomplete (i.e., we do not have the location of the user at all time instants), we also need to fill in the missing data at the same time. Let ET be an estimated completion for TT. Formally, we estimate the profile P of the user with the expectation $E[P|TT, TC]$. To compute this expectation we will sample from the distribution

$$\Pr(P|TT, TC) = \sum_{ET} \Pr(P, ET|TT, TC). \qquad (6)$$

However, sampling directly from $\Pr(P, ET|TT, TC)$ is not straightforward; it involves computing the sum of terms whose number grows exponentially with the length of the trace. Hence, we use Gibbs sampling, a

Monte Carlo method, as it only takes polynomial time to produce a sample from the conditional distributions $\Pr(P|ET, TT, TC)$ and $\Pr(ET|P, TT, TC)$. In order to sample from $\Pr(P, ET|TT, TC)$, we create a homogeneous Markov Chain on the state space of P and ET in an iterative procedure. Starting from an initial value for $ET^{\{0\}}$, Gibbs sampling produces pairs $(P^{\{l\}}, ET^{\{l\}})$ as follows:

$$P^{\{l\}} \sim \Pr(P|ET^{\{l-1\}}, TT, TC) \quad (7)$$

$$ET^{\{l\}} \sim \Pr(ET|P^{\{l\}}, TT, TC) \quad (8)$$

Convergence properties of the Gibbs sampling for this problem are studied in [20]. We are interested in the sequence of the $P_{ij}^{\{l\}}$ values; it is not a Markov chain, but it is ergodic and converges at geometric rate to a stationary distribution, which is the desired $\Pr(P|TT, TC)$. We compute P_{ij} for every i, j as the average of $P_{ij}^{\{l\}}$ over all samples l.

Now, the only remaining question is how to sample from the distributions (7) and (8). In order to sample a $P^{\{l\}}$ from (7), we assume that the rows of the transition matrix P are independent, and we produce samples for each row separately. We also consider a Dirichlet prior for each row P_i. Hence, the l^{th} sample for row P_i comes from the following distribution:

$$Dirichlet\left(\{TC_{ij} + Cnt_{ij}(ET^{\{l-1\}}) + \epsilon_{ij}\}_{j=1..M}\right) \quad (9)$$

where $Cnt_{ij}(\cdot)$ is the number of transitions from region r_i to r_j in a trace, and ϵ_{ij} is a very small positive number if, according to the *mobility constraints*, it is possible to move from r_i to r_j in one time instant (otherwise ϵ_{ij} is zero).

To sample an $ET^{\{l\}}$ from (8), we follow the simplification proposed in [20] and iteratively construct $ET^{\{l\}}$ by performing T successive samplings, for $t = 1, \ldots, T$, from

$$\frac{P^{\{l\}}_{ET(t-1)ET(t)}\, b(TT(t)|ET(t))\, P^{\{l\}}_{ET(t)ET(t+1)}}{\sum_{r\in\mathcal{R}} P^{\{l\}}_{ET(t-1)r}\, b(TT(t)|r)\, P^{\{l\}}_{rET(t+1)}}. \quad (10)$$

The values $P^{\{l\}}_{ET(0)ET(1)}$ and $P^{\{l\}}_{ET(T)ET(T+1)}$ are defined to be 1. The function $b(r|ET(t)), r \in TT$ is equal to 0 if $r \neq \emptyset$ and $r \neq ET(t)$. Otherwise, it is equal to 1. Note that the function $b(r_i|r_j)$ can also represent the noise function if the training trace is noisy: $b(r_i|r_j)$ is the probability that r_j is reported as r_i.

C. Tracking Attacks

We now describe two tracking attacks and their implementations. Recall from Section II-C that in a tracking attack the adversary is interested in reconstructing complete or partial actual traces, i.e., in *sequences* of events, rather than just isolated events.

1) Maximum Likelihood Tracking Attack: The objective of this attack is to find the jointly most likely traces for all users, given the observed traces. Formally, the objective is to find

$$\arg\max_{\sigma, A} \Pr(\sigma, A|O). \quad (11)$$

Notice that the maximization above is done in a space with $N!M^T$ elements, so a brute force solution is impractical.

We proceed by running this attack in two phases: first deanonymization and then deobfuscation. The deanonymization phase finds the most likely assignment of users to obfuscated traces. Notice that it is not correct to simply assign to each user the trace that she is most likely to have created, because more than one user might be assigned to the same trace. The most likely assignment is a *joint* assignment; it maximizes the product $\prod_{u\in\mathcal{U}} P(o_{\sigma(u)}|P^u)$ over all $N!$ user-to-trace assignments.

The most likely assignment is found as follows. First, the likelihood $P(o_x|P^u), x \in \mathcal{U}', u \in \mathcal{U}$ is computed for all $O(N^2)$ trace-user pairs (o_x, u). For the case when the obfuscation function operates on each region separately, we compute the likelihood for each pair with the *Forward-Backward algorithm* [18]. With this algorithm, each likelihood computation takes time $O(TM^2)$ by taking advantage of the recursive nature of the likelihood that we want to compute. In particular, we define the *forward* variable $\alpha_t(r), t \in \mathcal{T}, r \in \mathcal{R}$ as

$$\alpha_t(r) = \Pr\{o_x(1), o_x(2), \ldots, o_x(t), a_x(t) = r|P^u\}, \quad (12)$$

which is the joint probability of the observed trace o_x up to time t and that the actual location of the user with pseudonym x is r at time t, given that the pseudonym x is associated with user u. Notice that, if we can compute the forward variable at all regions at time T, i.e., $\alpha_T(r), r \in \mathcal{R}$, then the desired likelihood is simply

$$P(o_x|P^u) = \Pr\{o_x(1), o_x(2), \ldots, o_x(t), a_x(t) = r|P^u\} = \sum_{r=r_1}^{r_M} \alpha_T(r). \quad (13)$$

For the recursive computation of the forward variables we use the fact that

$$\alpha_{t+1}(r) = \left(\sum_{\rho=r_1}^{r_M} \alpha_t(\rho) P^u_{\rho r}\right) f_r(o_x(t+1)), \quad 1 \leq t \leq T-1, r \in \mathcal{R}. \quad (14)$$

Within the sum there is one term for each way of reaching region r at time $t+1$, i.e., having been at each of the regions $\rho \in \mathcal{R}$ at time t. After computing the sum, we only need to multiply with the probability of obfuscating region r to the location pseudonym observed at time $t+1$. The only remaining issue is the initialization of the forward variables:

$$\alpha_1(r) = \pi^u_r f_r(o_x(1)), r \in \mathcal{R}. \quad (15)$$

The vector $\pi_r^u, r \in \mathcal{R}$ is the steady state probability vector for the mobility profile of u.

For the computation of the likelihood we do not need the backward variables (which is where the rest of the algorithm's name comes from). We will, however, define and use them in Section III-D on Localization attacks.

The whole likelihood computation for one trace-user pair can be done in $M(M+1)(T-1)+M$ multiplications and $M(M-1)(T-1)$ additions. If the obfuscation function operates on the whole trace simultaneously, rather than on each region individually, the worst case computation will take time $O(TM^T)$.

Having computed the likelihoods for all trace-user pairs, we complete the deanonymization phase of the attack by assigning exactly one trace to each user. To this end, we create an edge-weighted bipartite graph of traces and users, where the weight of the edge between user u and trace o_x is the likelihood $P(o_x|P^u)$. Then, we find the *Maximum Weight Assignment* (MWA) in this graph. We use the Hungarian algorithm, which has time complexity of order $O(N^4)$. Faster algorithms exist, but the Hungarian algorithm is simple, and the MWA only needs to be computed once in this attack; the MWA is also an instance of a linear program, so linear program solvers can be used. The outcome is a matching of users and traces, such that the product $\prod_{u \in \mathcal{U}} P(o_{\sigma(u)}|P^u)$ is maximized over all $N!$ user-to-trace assignments.

Given the maximum weight assignment, we proceed to the second phase of the attack: We find the most likely deobfuscation for the trace assigned to each user. We use the Viterbi algorithm [18] to do that. More formally, the most likely deobfuscation is

$$\arg \max_{a_u \in \mathcal{A}_u} \Pr\{a_u(t), t=1,\ldots,T | o_u(t), t=1,\ldots,T\}. \tag{16}$$

The Viterbi algorithm is a dynamic programming algorithm. We define $\delta_t(r)$ as

$$\delta_t(r) = \max_{a_u(s)_{s=1,\ldots,t-1}} \Pr\{\ a_u(s)_{s=1,\ldots,t-1}, a_u(t)=r, \\ o_u(s)_{s=1,\ldots,t-1}|P^u\}\,, \tag{17}$$

which is the joint probability of the most likely trace $a_u(\cdot)_1^{t-1}$ that at time t is at region r, and the trace observed up to time t. Maximizing this quantity is equivalent to maximizing (16). Then, similarly as before, we recursively compute the values at time T, i.e., $\delta_T(r)$.

$$\delta_t(r) = \max_{\rho \in \mathcal{R}} \left(\delta_{t-1}(\rho) P_{\rho r}^u\right) f_r(o_u(t)), \\ 2 \le t \le T, r \in \mathcal{R}. \tag{18}$$

The initialization in this case is

$$\delta_1(r) = \pi_r f_r(o_u(1)), r \in \mathcal{R}. \tag{19}$$

From the values $\delta_T(r)$, we compute the joint probability of the most likely trace and the observations by computing

$$\max_{r \in R} \delta_T(r). \tag{20}$$

Of course, we are interested in the most likely trace itself, not only in its probability. The most likely trace is computed by keeping track, at each time $2 \le t \le T$, of the argument (region ρ) that maximizes (18) and, for $t=T$, the one that maximizes (20). Then, we can backtrack from time T back to time 1 and reconstruct the trace.

Parenthetically, notice that finding the most likely trace is exactly equivalent to finding the shortest path in an edge-weighted directed graph. The graph's MT vertices are labeled with elements of the set $\mathcal{R} \times \mathcal{T}$, i.e., for each time t there are M vertices corresponding to each of the M regions. There are edges only from vertices labeled with time t to vertices labeled $t+1, 1 \le t \le T-1$. The weight of an edge $(t,r) \to (t+1,\rho)$ is equal to $-\log P_{r\rho}^u f_\rho(o_u(t+1))$. Indeed, minimizing the sum of negative logarithmic terms is equivalent to maximizing the product of the original probabilities.

Having completed the two phases of the attack, we observe that the trace computed is not necessarily a maximum for (11). Indeed from (11), it follows that:

$$\arg\max_{\sigma,a} \Pr(\sigma, a|\boldsymbol{O}) = \arg\max_{\sigma,a} \Pr(a|\sigma,\boldsymbol{O}) \Pr(\sigma|\boldsymbol{O}) \\ = \arg\max_{\sigma,a} \prod_i \Pr(\boldsymbol{A}_u = a_{u_i}|\boldsymbol{O}_{\sigma(u_i)}) \Pr(\sigma|\boldsymbol{O}).$$

Indeed, MWA does maximize the second term (actually, it maximizes $\Pr(\boldsymbol{O}|\sigma)$ over all σ, which is equivalent to maximizing $\Pr(\sigma|\boldsymbol{O})$) and Viterbi does maximize the first (i.e., $\Pr(a|\sigma,\boldsymbol{O})$). But, it is possible that an assignment σ^* and a set of traces a^* that jointly maximize the total likelihood ($\Pr(\sigma,a|\boldsymbol{O})$) are different from the results obtained from the MWA and Viterbi algorithms separately.

However, we consider such cases as pathological: In the MWA, a user u is mapped to an obfuscated trace o_u that he has high likelihood of producing. That is, u is likely to produce unobfuscated traces that are, in turn, likely to be obfuscated to o_u. In other words, the unobfuscated traces that are typical for u are likely to be obfuscated to o_u. There might be a nontypical outlier (a^*) that is more likely than the most likely typical trace, but that optimal combination would be isolated in the $\mathcal{A}$ space. As such, choosing the outlier would not be *robust* to small changes in, for example, the mobility model.

2) Distribution Tracking Attack: We now consider the most general type of tracking attack, one which computes the *distribution* of traces for each user, rather than just the most likely trace:

$$\Pr\{\cap_{i=1}^N \boldsymbol{A}_{u_i} = a_{u_i}, \boldsymbol{\Sigma} = \sigma | o_1, o_2, \ldots, o_N\} \tag{21}$$

The implementation of this attack uses the Metropolis-Hastings (MH) algorithm on the product of the space $\mathcal{A}$ with the space of all possible permutations σ. The purpose of the MH algorithm is to draw independent samples (from the space $\mathcal{A} \times \Sigma$) that are identically distributed according to the desired distribution (21). The algorithm makes use of the fact that the desired distribution, briefly written as $\Pr\{a, \sigma|o\}$, is equivalently:

$$\Pr\{a, \sigma|o\} = \frac{\Pr\{o|a, \sigma\} \Pr\{\sigma|a\} \Pr\{a\}}{\Pr\{o\}} \qquad (22)$$

The denominator is a normalizing factor that is hard to compute, but it does not depend on a. The algorithm allows us to sample from the distribution $\Pr\{a, \sigma|o\}$ without computing the denominator $\Pr\{o\}$. However, the numerator needs to be easy to compute, which is true in our case: We compute the probability $\Pr\{o|a, \sigma\}$ using (1); the probability $\Pr\{\sigma|a\}$ is constant and equal to $\frac{1}{N!}$, as we use random permutation as the anonymization function; and the probability $\Pr\{a\}$ is computed from the users' profiles.

At a high level, the MH algorithm performs a random walk on the space of possible values for (a, σ). The transition probabilities of the random walk are chosen so that its stationary distribution is the distribution from which we want to sample.

First of all, we need to find a feasible initial point for the walk (i.e., a point that does not violate the mobility profile of any user; it is not a trivial matter to find such a point). We use the output of the maximum likelihood tracking attack.

We then need to define a neighborhood for each point (a, σ). We define two points (a, σ) and (a', σ') to be neighbors if and only if exactly one of the three following conditions holds:

- The components σ and σ' differ in exactly two places. That is, $N - 2$ out of the N traces are assigned to the same users in both σ and σ', and the assignment of the remaining two traces to users is switched. The components a and a' are identical.
- The components a and a' differ in exactly one place. That is, the location of exactly one user at exactly one timeslot is different. All other locations are unchanged. The components σ and σ' are identical.
- Points (a, σ) and (a', σ') are identical. That is, a point is assumed to be included in its own neighborhood.

We finally define a proposal density function that determines the candidate neighbor to move to at the next step; this function also influences the convergence speed of the algorithm. For simplicity, we use a uniform proposal density, so the candidate is selected randomly among all neighbors.

To perform the random walk, suppose that the current point is (a, σ) and the selected candidate is (a', σ'). Then, (a', σ') is accepted with probability $\min\{\frac{\Pr\{o|a', \sigma'\} \Pr\{a'\}}{\Pr\{o|a, \sigma\} \Pr\{a\}}, 1\}$. If (a', σ') is rejected, then we repeat the procedure of selecting and probabilistically accepting a neighbor. If it is accepted, it is logged as a step in the random walk. However, it is not an independent sample, as it is correlated with (a, σ). Only after making enough steps to overcome the inherent correlation among successive steps is a step stored as an independent sample. After storing enough independent samples, the algorithm stops.

How many independent samples are enough? The attacker collects as many samples as needed to satisfy his accuracy requirements. The confidence interval for the chosen confidence level must be shorter than the desired length. Suppose the attacker needs to collect n independent samples.

How many steps of the random walk must be taken between each pair of successive samples to ensure the independence of these n samples? There are standard statistical tests of independence; our choice is the *Turning Point* test. The basic idea of this test is that, among three successive independent and identically distributed samples, all $3! = 6$ possible orderings are equiprobable. Given three numerical values x_{i-1}, x_i, x_{i+1}, a *turning point* exists at i if and only if x_i is either larger than both x_{i-1}, x_{i+1} or smaller than both x_{i-1}, x_{i+1}. If the three numerical values are independent and identically distributed, then the probability of a turning point is $\frac{2}{3}$. Then, given a large enough number of values, n in our case, the number of turning points is approximately Gaussian with mean $\frac{2n-4}{3}$ and variance $\frac{16n-29}{90}$.

So, we test if the number of turning points in our sequence of n MH samples can be claimed to be Gaussian with this mean and variance. If so, we stop. Otherwise, we make more steps in the random walk and skip more and more of the logged intermediate steps before storing each sample.

It should be emphasized that ***the Distribution Tracking attack can answer all kinds of U-R-T questions***. The attacker can specify a very wide range of objectives as functions of a sample of the MH algorithm. Then, the attacker computes this function on each independent sample, and the sample average of the computed values is the estimate of the attacker's objective. In this case, the accuracy and certainty metrics would be computed on the values that the function returns, rather than directly on the MH samples.

Despite its generality, the Distribution Tracking attack is computationally intensive. So, it might make sense to use heuristics to find the distribution of traces for each user.

An important heuristic is to consider, as we have already seen, only the most likely deanonymization. Then we find the posterior distribution of nonobfuscated traces *separately* for each user-to-obfuscated-trace assignment that the deanonymization produced. Formally, the objective is to find the pdf

$$\max_{\sigma} \Pr(\sigma, a|\boldsymbol{O}). \qquad (23)$$

The implementation of this heuristic is simply to find the MWA, as explained in the Maximum Likelihood Tracking attack, and then run Metropolis-Hastings for each user-trace pair separately. That is, MH would run on each space $\mathcal{A}_u$ separately for each u, and of course the neighborhood of a point would be restricted to single location changes, as there can be no changes in the username part.

D. Localization Attacks

In localization attacks, a typical question is to find the location of a user u at some time t. The most general answer to such a question is to compute

$$\Pr\{a_u(t) = r | o_u, P^u\} \tag{24}$$

for each $r \in \mathcal{R}$. The output for the attacker is a distribution on the possible regions, from which he can select the most probable, or form an average, etc. For this attack, the attacker needs to know or estimate the observed trace that user u created, perhaps by using the Maximum Weight Assignment, which is what we have implemented.

Of course, he can perform the attack for each of the observed traces, as it is not very computationally intensive. In particular, these probabilities can be easily computed with the Forward-Backward algorithm. In the section on the Maximum Likelihood Tracking attack, we described the computation of the forward variables

$$\alpha_t(r) = \Pr\{o_x(1), o_x(2), \ldots, o_x(t), a_x(t) = r | P^u\}. \tag{25}$$

The backward variables are defined to be

$$\beta_t(r) = \Pr\{o_x(t+1), o_x(t+2), \ldots, o_x(T) | a_x(t) = r, P^u\}, \tag{26}$$

that is, $\beta_t(r)$ is the probability of the partial trace from time $t+1$ to the end, given that the region at time t is r and given that user u created the trace.

Again, we can recursively compute the backward variables using the fact that

$$\beta_t(r) = \sum_{\rho=r_1}^{r_M} P^u_{r\rho} f_\rho(o_x(t+1)) \beta_{t+1}(\rho),$$
$$t = T-1, T-2, \ldots, 1, r \in \mathcal{R}. \tag{27}$$

Notice that the computation takes place backwards in time. The initialization (at time T) of the backward variables is arbitrary:

$$\beta_T(r) = 1, r \in \mathcal{R}. \tag{28}$$

Having computed the backward variables, the probability $\Pr\{a_u(t) = r | o_u\}$ is then equal to

$$\Pr\{a_u(t) = r | o_u, P^u\} = \frac{\alpha_t(r)\beta_t(r)}{\Pr(o_u | P^u)}. \tag{29}$$

The variable $\alpha_t(r)$ accounts for the observations up to time t and region r at time t, and $\beta_t(r)$ accounts for the remainder of the observed trace, given that the region at t is r. The term $\Pr(o_u|P^u)$ is a normalization factor that was earlier computed as $\sum_{r=r_1}^{r_M} \alpha_T(r)$. An alternative way of computing it is as $\sum_{r=r_1}^{r_M} \alpha_t(r)\beta_t(r)$, which more directly shows its role as a normalization factor.

E. Meeting Disclosure Attacks

In a meeting disclosure attack, a typical objective specifies a pair of users u and v, a region r, and a time t, and then it asks whether this pair of users have met at that place and time. The probability of this event is computed as the product $\Pr\{a_u(t) = r | o_u, P^u\} \Pr\{a_v(t) = r | o_v, P^v\}$ by using the results of the localization attack. A more general attack would specify only a pair of users and ask for the expected number of time instants that they have met in any region. Such questions can be answered by using the results of the localization attack for each user u_i as will be explained in Section IV. Yet another question would not specify any usernames, but only a region and a time. The objective would be the expected number of present users in the region at that time. Again, a localization attack for each user would be the first step as will be explained in Section IV.

IV. Using the Tool: Evaluation of LPPMs

In this Section, we pursue two main goals:

- We show a few examples of using the *Location-Privacy Meter* to quantify the effectiveness of LPPMs against various attacks.
- We evaluate the appropriateness of two popular metrics, namely, *k-anonymity* and *entropy*, for quantifying location privacy.

In order to use the *Location-Privacy Meter*, we first need to provide and specify (i) the location traces that we obfuscate/anonymize, (ii) the LPPMs that we implement, and (iii) the attacks that we perform.

The location traces that we use belong to $N = 20$ randomly chosen mobile users (vehicles) from the epfl/mobility dataset at CRAWDAD [17]. Each trace contains the location of a user every 5min for 8hours (i.e., $T = 96$). The area within which users move (the San Francisco bay area) is divided into $M = 40$ regions forming a 5×8 grid.

We use two location-privacy preserving mechanisms that are explained in Section III-A: *precision reducing* with parameters μ_x, μ_y (the number of dropped low-order bits from the x, y coordinate of a region, respectively), and *location hiding* with parameter λ_h (the probability of hiding a region). Let $\mathrm{LPPM}(\mu_x, \mu_y, \lambda_h)$ denote an LPPM with these specific parameters. The traces are also anonymized using a random permutation function (i.e., each user is assigned a unique pseudonym from 1 to N).

In order to consider the strongest adversary, we feed the *knowledge constructor (KC)* module with the users' actual traces. We run the inference mechanisms explained in Sections III-C and III-D and obtain results for the following U-R-T attack scenarios:

- **LO-ATT**: *Localization Attack*: For a given user u and time t, what is the location of u at t? (Since the location is a random variable, the answer is the probability distribution over the regions).

- **MD-ATT**: *Meeting Disclosure Attack*: For a given pair of users u and v, what is the expected number of meetings between u and v? Put differently, at how many time instants in $\mathcal{T}$ the two users are in the same region.
- **AP-ATT**: *Aggregated Presence Attack*: For a given region r and time t, what is the expected number of users present in r at t?

The metric to evaluate location privacy of users in all three attacks is the failure of the adversary in finding the correct answer: his *incorrectness*. For **LO-ATT**, according to (5), the privacy of user u at time t is computed as

$$LP_{\textbf{LO-ATT}}(u,t) = \sum_{r \in \mathcal{R}} \hat{p}_{u,t}(r) \| r - a_u(t) \| \tag{30}$$

where $a_u(t)$ is the actual location of u at time t, and the distance $\|r - a_u(t)\|$ is equal to 0 if $r = a_u(t)$ (i.e., correct estimation by the adversary), and it is equal to 1 otherwise. Moreover, $\hat{p}_{u,t}(r) = \widehat{\Pr}\{a_u(t) = r | o_u, P^u\}$ as described in Section III-D.

For **MD-ATT**, let $Z^t_{u,v} = 1_{a_u(t)=a_v(t)}$ be the random variable that indicates whether u and v meet at time t. The adversary estimates their expected number of meetings over all time instants

$$\widehat{E}(\sum_t Z^t_{u,v}) = \sum_t \widehat{\Pr}(Z^t_{u,v} = 1) = \sum_t \sum_r \hat{p}_{u,t}(r) \hat{p}_{v,t}(r)$$

The actual number of meetings between u and v is $\sum_t 1_{a_u(t)=a_v(t)}$. Hence, according to (5), the privacy of u and v is

$$LP_{\textbf{MD-ATT}}(u,v) = \| \widehat{E}(\sum_t Z^t_{u,v}) - \sum_t 1_{a_u(t)=a_v(t)} \|, \tag{31}$$

whose values range from 0 and T.

For **AP-ATT**, let $Y^u_{r,t} = 1_{a_u(t)=r}$ be the random variable that indicates whether u is in r at t. The adversary estimates the expected value of $\sum_u Y^u_{r,t}$ which is

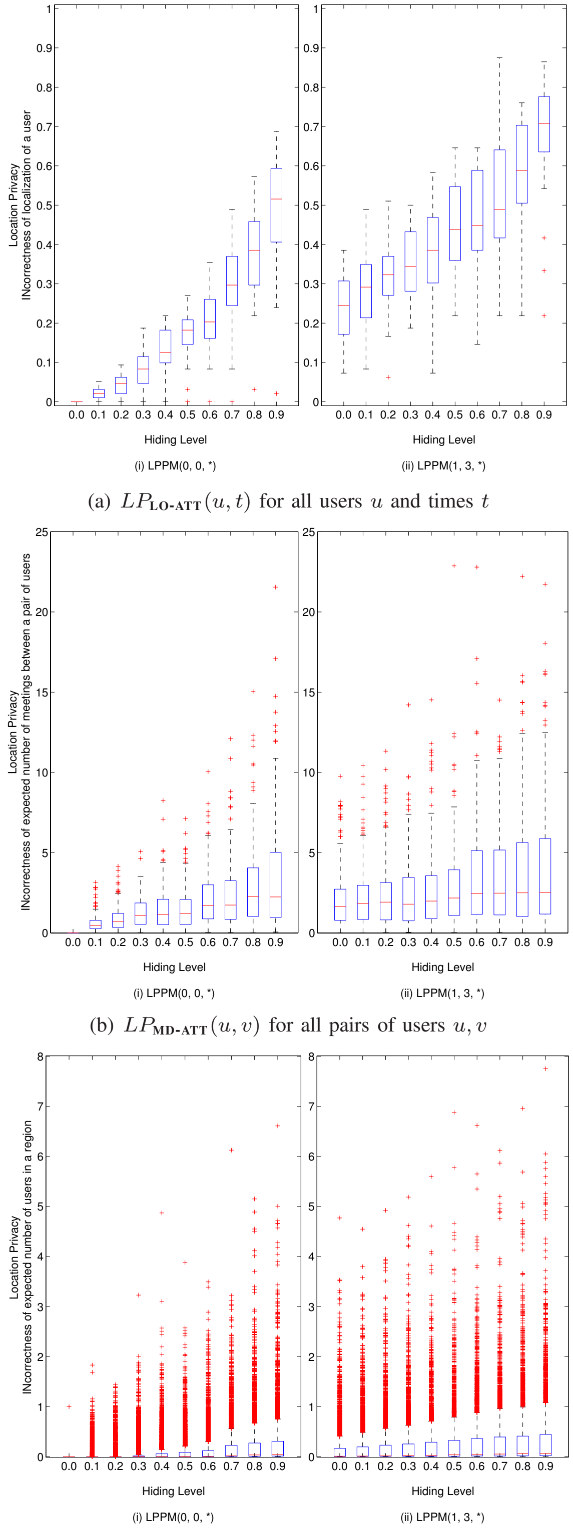
$$\widehat{E}(\sum_u Y^u_{r,t}) = \sum_u \widehat{\Pr}(Y^u_{r,t} = 1) = \sum_u \hat{p}_{u,t}(r)$$

The actual number of users in region r at t is $\sum_u 1_{a_u(t)=r}$. Hence, according to (5), the privacy of users at time t for region r is

$$LP_{\textbf{AP-ATT}}(r,t) = \| \widehat{E}(\sum_u Y^u_{r,t}) - \sum_u 1_{a_u(t)=r} \|, \tag{32}$$

and its values range from 0 to N.

Figure 4 illustrates the results that we have obtained about the effectiveness of the precision-reduction and location-hiding LPPMs against these three attacks. Each row in the figure corresponds to one attack. The left-hand column shows the results for the LPPM with parameters $(0,0,0.0), (0,0,0.1), ..., (0,0,0.9)$, and the right-hand column shows the results for the LPPM with



(a) $LP_{\textbf{LO-ATT}}(u,t)$ for all users u and times t

(b) $LP_{\textbf{MD-ATT}}(u,v)$ for all pairs of users u, v

(c) $LP_{\textbf{AP-ATT}}(r,t)$ for all regions r and times t

Figure 4. The system-level location-privacy against attacks **LO-ATT**(a), **MD-ATT**(b) and **AP-ATT**(c). Left-hand and right-hand side plots show the attack results against $\text{LPPM}(0,0,*)$ and $\text{LPPM}(1,3,*)$, respectively. The last parameter of LPPMs (hiding level λ_h) is shown on the x-axis. The boxplot shows, in particular, the median, 25^{th} and 75^{th} percentiles.

parameters $(1, 3, 0.0), (1, 3, 0.1), ..., (1, 3, 0.9)$. Recall that $\mathrm{LPPM}(\mu_x, \mu_y, \lambda_h)$ denotes the location-privacy preserving mechanism with parameters μ_x and μ_y as the number of dropped low-order bits from the x and y coordinates, respectively, and with parameter λ_h as the probability of hiding a region. Each box-and-whisker diagram (boxplot) shows the system level location-privacy of users for a specific set of LPPM parameters against a specific attack. The bottom and top of a box show the 25^{th} and 75^{th} percentiles, and the central mark shows the median value. The ends of the whiskers represent the most extreme data points not considered as outliers, and the outliers are plotted individually.

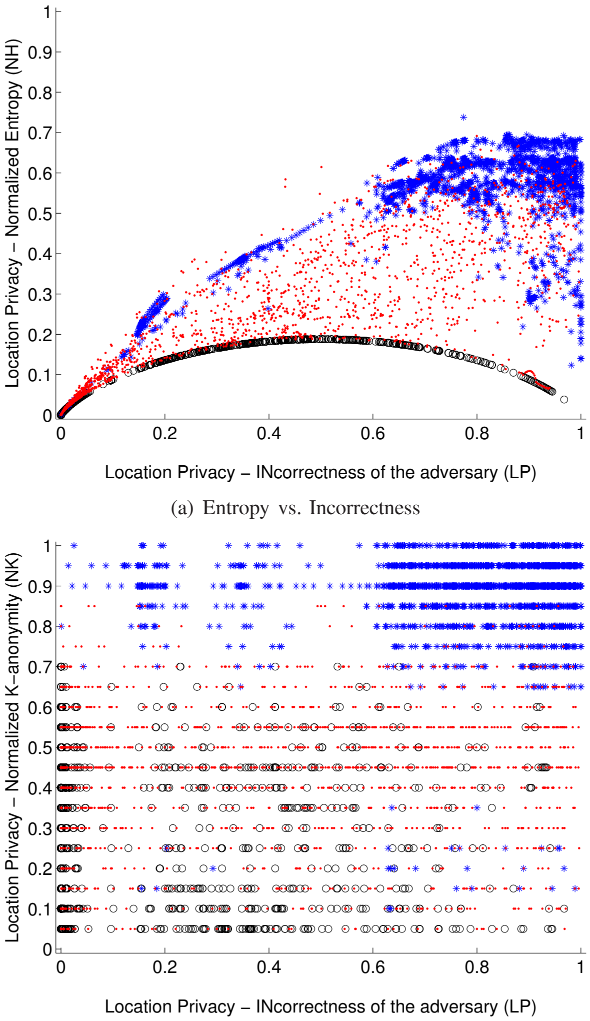
By system-level location-privacy, we collectively refer to the privacy values (expected error - incorrectness) achieved for all possible combinations of attack parameters ((u, t) for **LO-ATT**, (u, v) for **MD-ATT**, (r, t) for **AP-ATT**). The system-level location-privacy is represented by the median privacy value, shown in the boxplot as the central mark in the box. We also plot the 25^{th} and 75^{th} percentiles of the privacy value in order to show the diversity of adversary's expected error. As an example, the first boxplot in Figure 4(a).ii, which is associated with 0.0 in the x-axis, shows $LP_{\textbf{LO-ATT}}(u, t)$ for all u and t, using $\mathrm{LPPM}(1, 3, 0.0)$.

We expect to see improvement in location privacy, as we increase the level of obfuscation. We also expect to observe convergence of location privacy to its near maximum value, when we set the location-hiding level equal to 0.9 (i.e., 90% of the users' locations are hidden from the adversary). Unsurprisingly, we observe these two things in the plots: Reading a plot from left to right we see the effect of increasing the hiding level λ_h (0.0 to 0.9) for constant precision-reducing levels μ_x and μ_y. Namely, the privacy always increases, although the effect is much more pronounced in **LO-ATT**(first row). By comparing corresponding boxes of two adjacent plots, i.e., same hiding levels, we see the added value of the precision-reducing mechanism (on the left, μ_x and μ_y are both 0; on the right, μ_x is 1 and μ_y is 3). Again, the clearest improvement happens in **LO-ATT**.

An interesting conclusion is that the effect of the LPPM is most positive against **LO-ATT**, which is, in a sense, the most intrusive attack of the three: it targets the exact location of a single user at a single time. The other two attacks, especially **AP-ATT**, are more related to statistics of the user mobility, so there could even be legitimate reasons that one would want to collect that information. For instance, a researcher who studies the geographical distribution of users would be interested in the number of users in a region. We can conclude that the tested LPPMs protect users' location-privacy against malicious adversaries, but they still provide information for less harmful activities.

Now, we assess the appropriateness of two metrics, namely *k-anonymity* and *entropy*, for quantifying location privacy. Note that any other heuristic metric can be evaluated in the same way. We focus on **LO-ATT**, and we assess these metrics by testing to what extent they are correlated to the success of the adversary in correctly localizing users over time (i.e., the incorrectness metric $LP_{\textbf{LO-ATT}}(u, t)$). We choose three LPPMs: $\mathrm{LPPM}(1, 0, 0.0)$ as a weak mechanism, $\mathrm{LPPM}(1, 2, 0.5)$ as a medium mechanism, and $\mathrm{LPPM}(2, 3, 0.9)$ as a strong mechanism.

In Section II-D, we use entropy to measure the uncertainty of the adversary. Here, we assess the normalized entropy of the pdf of the location of user u at time t, as a metric for



(a) Entropy vs. Incorrectness

(b) K-anonymity vs. Incorrectness

Figure 5. Comparison of location-privacy metrics. The x-axis shows the users' location-privacy based on the incorrectness metric (30). The y-axis shows (a) the normalized entropy of the adversary's estimation, (b) the normalized k-anonymity. Each point in the plot represents the location privacy of some user at some time for two metrics (incorrectness vs entropy in (a), incorrectness vs k-anonymity in (b)). "$*$"s are the location privacy values achieved from $\mathrm{LPPM}(2, 3, 0.9)$ as a strong mechanism, "$\cdot$"s are the values for $\mathrm{LPPM}(1, 2, 0.5)$ as a medium mechanism, and "$\circ$"s are the values for $\mathrm{LPPM}(1, 0, 0.0)$ as a weak mechanism. The two metrics would be fully correlated only if all points were on the diagonal $(0, 0)$ to $(1, 1)$.

her location privacy. The normalized entropy is computed as follows:

$$NH_{\textbf{LO-ATT}}(u,t) = \frac{-\sum_{r\in\mathcal{R}} \hat{p}_{u,t}(r)\log(\hat{p}_{u,t}(r))}{\log(M)} \quad (33)$$

where $\log(M)$ is the maximum entropy over M regions.

According to the k-anonymity metric, the location-privacy of a user u at a given time t is equal to the number of users who satisfy all of the following conditions: (i) they are anonymous, (ii) they obfuscate their location by merging regions (which includes their actual location), (iii) their obfuscated location (i.e., the set of merged regions) is a superset of the obfuscated location of u at t. We divide this number of users by N, the total number of users, to have the normalized k-anonymity:

$$NK_{\textbf{LO-ATT}}(u,t) = \frac{1}{N}\sum_{v\in\mathcal{U}} 1_{a_v(t)\in o_u(t)\wedge o_u(t)\subseteq o_v(t)} \quad (34)$$

Figure 5 illustrates the relation between the incorrectness of the adversary $LP_{\textbf{LO-ATT}}(u,t)$ and the two above-mentioned metrics: normalized entropy $NH_{\textbf{LO-ATT}}(u,t)$, and normalized k-anonymity $NK_{\textbf{LO-ATT}}(u,t)$. We see that the entropy is more correlated to the adversary's incorrectness than k-anonymity is. However, both entropy and k-anonymity misestimate the true location privacy of users.

Let us focus on Figure 5(a). All but few of the points fall into the "$NH < LP$" triangle, which means that, in this setting, the entropy metric underestimates location privacy. For example, consider the "$*$"s on the $NH = 0.6$ horizontal line, all of whose entropy is 0.6. The incorrectness metric (LP) of these points ranges from 0.6 to 1. Or, consider the vertical line $LP = 1$, where there are "$*$"s corresponding to values of NH ranging from 0.2 to 0.7. In both cases, the estimation of location privacy by NH is up to 5 times less than the true location privacy of users, which makes it an unappropriate and loose lower bound for location privacy. We observe the same phenomenon in the results of the two other LPPMs (represented by "$\cdot$"s and "$\circ$"s).

The results are even worse for k-anonymity in Figure 5(b) as there is less correlation between NK and LP. In fact, k-anonymity in some cases underestimates location privacy (consider the area where $NK < 0.5$ and $LP > 0.5$) and in some other cases ($NK > 0.5$ and $LP < 0.5$) overestimates it. Hence, this is not an appropriate estimator for location privacy either.

V. Related Work

There are several papers in the field of location privacy that aim at clarifying the way to effectively protect users' location privacy by classifying the problems and studying various unaddressed issues and missing elements in this field of research. We will discuss these papers in the beginning of this section. These papers cover a range of different concerns, but highlight the following two urgent topics:

- Understanding the threats and formalizing the attacks on location privacy
- Designing a standard and appropriate evaluation metric for location privacy based on a sound theoretical model that can be used to compare various schemes

Krumm [14] studies various computational location privacy schemes: those that can be formally specified and quantitatively measured. The authors regard the *accuracy of location privacy metrics* as the key factor in the progress of computational location privacy, and emphasize the importance of finding a single (or a small set of sufficient) quantifier for location privacy.

Decker [6] gives an overview of location privacy threats and studies the effects of various countermeasures on protecting location privacy. The author also discusses which protection mechanisms (such as obfuscation, anonymization) are appropriate for different location-based services, considering the specification and requirements of those services.

Shokri *et al.* [21], [22] survey various LPPMs and also the metrics used for measuring location privacy (called uncertainty-based, error-based and k-anonymity). The authors compare various metrics *qualitatively* and show that metrics such as entropy and k-anonymity are inadequate for measuring location privacy. The authors rely on a number of common-sense examples to justify the results.

Duckham [7] proposes a few rules as the key principles of research on location privacy, which make this field of research different from other research topics in privacy. The author refers to the predictable mobility of humans, the constraints of the area within which people move, the effects of location-based applications on privacy, the effectiveness of centralized vs. distributed protection mechanisms and, last but not least, the importance of a formal definition of fundamental terms (such as the precision and accuracy of information) in the design of protection mechanisms.

All the above-mentioned papers, of course, have been a source of inspiration for our research in this paper. However, despite the fact that we share common concerns (especially the two emphasized items in the beginning of this Section) neither these papers, nor any other paper we know about, provide a framework with which LPPMs can be evaluated quantitatively. Our work is a realization of the goals and concerns of the research community and provides a modular platform every part of which can be separately analyzed and be improved upon; for example, by simulating more powerful attacks using other inference techniques.

Other papers related to our work implement particular attacks to show the *predictability* and *uniqueness* of users' location traces, and some of them evaluate the efficacy of specific protection mechanisms. Each paper uses a different model to state the problem and evaluate location privacy. In spite of this diversity, this provides us with tools that can potentially be used in a generic framework.

A prominent example of such papers is [15], in which Liao *et al.* propose a framework for recognizing mobile users' activities based on the places they visit and also the temporal patterns of their visit. The authors develop an inference technique based on Markov Chain Monte Carlo (MCMC) methods and show how users' activities are dependent on their mobility traces. The paper does not talk about the consequences of these techniques, if used by an adversary, on users' privacy. However, it shows the relation between location privacy (i.e., to what extent a user's identity is unlinkable to a location) and the general privacy of mobile users (e.g., their activities and habits). Thus, it explains the value of protecting mobile users' location-privacy for preventing the loss of their general privacy.

Other papers define the users' (location) privacy as the extent to which the users' names (real identities) can be derived from their traces. In our terms, they address "what is the likelihood that an anonymous trace belongs to a given user." In fact, the results show the uniqueness of users' mobility patterns.

Bettini *et al.* [2] state that location traces can act as quasi-identifiers of mobile users and lead to identification of anonymous traces. Hence, they propose a k-anonymity method to protect users' anonymity.

Hoh *et al.* [12] and Krumm [13] focus on finding users' identities based on their home addresses. Hence, they run some inference attacks on location traces to find the home address of the user to which the trace belongs. The effectiveness of various protection mechanisms such as spatial cloaking (hiding), noise (perturbation), and rounding (reducing precision) on foiling these attacks are also evaluated.

Mulder *et al.* [5] show that anonymous location traces, even at a low space granularity (i.e., at the level of the size of the GSM cells) and spanning a short time period (a few hours), can be re-identified, given the mobility profiles of the individuals.

Golle and Partridge [10] discuss the anonymity of home/work location pairs. The authors show that knowing home and work addresses is enough to de-anonymize the location traces of most of the users (especially in the United States, where they obtained their results). Freudiger *et al.* [9] use more advanced clustering algorithms to show mobile users' privacy-erosion over time as they make use of various types of location-based services.

In the same vein of the previous works, Ma *et al.* [16] show that published anonymous mobility traces can be identified using statistical inference methods such as maximum likelihood estimators, if the adversary has access to some samples of those traces with known user names.

Note that these papers in general only highlight the vulnerability of location traces to *de-anonymization* by an adversary with access to different types of information.

However, there are very few research contributions where the authors focus on how traceable a user is; that is, the extent to which the adversary can correctly reconstruct a complete trace from partial fragments. An example of this line of investigation is [11], in which Hoh and Gruteser propose a tracking attack based on multi-target tracking algorithms [19] (using a Kalman filter) can help the adversary to link different pieces of a user's anonymous trace. The authors propose a path confusion method in which traces of different users are perturbed to create confusion in the tracking algorithm. They also formulate an optimization problem to solve the tradeoff between location privacy and usefulness of the perturbed traces.

In our paper, as opposed to the enumerated related work, we *jointly* consider obfuscation and anonymization methods and develop generic attacks that can be used against any LPPM. The framework we propose in this paper enables us to formalize and evaluate various LPPMs. To the best of our knowledge, the *Location-Privacy Meter* is the first generic tool developed to evaluate location privacy of location traces.

Finally, we should mention that modeling and formalizing evaluation frameworks for privacy has recently been the focus of researchers in other domains. Good examples of this movement are differential privacy (for databases, typically) proposed by Dwork [8], a framework to evaluate anonymity protocols by Chatzikokolakis *et al.* [3], an evaluation framework for MIX networks by Troncoso and Danezis [4], [24], and a privacy model for RFIDs by Vaudenay [25].

For a more in-depth survey of various privacy-preserving methods, metrics and attacks in the location-privacy literature, the reader is referred to [14], [21], [23].

Acknowledgment

The authors would like to thank George Danezis, Julien Freudiger and Prateek Mittal for their insightful discussions on the earlier versions of the framework, Mathias Humbert and Mohamed Kafsi for their valuable comments on the submitted manuscript, and also Vincent Bindschaedler for helping us in the development of the Location-Privacy Meter.

VI. Conclusion

In this paper, we have raised the questions "what is location privacy?" and "how can location privacy be quantified, given an adversary model and a protection mechanism?" In order to address these questions, we have established a framework in which various entities, which are relevant to location privacy of mobile users, have been formally defined. The framework enables us to specify various LPPMs and attacks. Within this framework, we were also able to unravel various dimensions of the adversary's inference attacks. We formally justify that the incorrectness of the adversary in his inference attack (i.e., his expected estimation error) determines the location privacy of users.

We have developed an operational tool, named *Location-Privacy Meter*, as a realization of our framework. A designer of an LPPM can easily specify and integrate her algorithm in this tool for evaluation. Relying on well-established statistical methods, we have implemented a generic attack that can be used to answer all sorts of information disclosure questions. We have also developed some specific attacks, such as localization attacks, that are more targeted and hence more time-efficient.

As a follow-up to this work, we will add new modules with which we can support pseudonym changes over time for users, in order to capture all possible LPPM algorithms. We would also like to incorporate the location-based applications into the framework and analyze the effectiveness of LPPMs with respect to these applications.

References

[1] Location-Privacy Meter tool. Available online through http://people.epfl.ch/reza.shokri, 2011.

[2] C. Bettini, X. S. Wang, and S. Jajodia. Protecting privacy against location-based personal identification. In *In 2nd VLDB Workshop SDM*, pages 185–199, 2005.

[3] K. Chatzikokolakis, C. Palamidessi, and P. Panangaden. Anonymity protocols as noisy channels. In *Proceedings of the 2nd international conference on Trustworthy global computing*, TGC'06, pages 281–300, Berlin, Heidelberg, 2007. Springer-Verlag.

[4] G. Danezis and C. Troncoso. Vida: How to use bayesian inference to de-anonymize persistent communications. In *Proceedings of the 9th International Symposium on Privacy Enhancing Technologies*, PETS '09, pages 56–72, Berlin, Heidelberg, 2009. Springer-Verlag.

[5] Y. De Mulder, G. Danezis, L. Batina, and B. Preneel. Identification via location-profiling in gsm networks. In *WPES '08: Proceedings of the 7th ACM workshop on Privacy in the electronic society*, pages 23–32, New York, NY, USA, 2008.

[6] M. Decker. Location privacy-an overview. In *ICMB '08: Proceedings of the 2008 7th International Conference on Mobile Business*, pages 221–230, Washington, DC, USA, 2008. IEEE Computer Society.

[7] M. Duckham. Moving forward: location privacy and location awareness. In *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Security and Privacy in GIS and LBS*, SPRINGL '10, pages 1–3, New York, NY, USA, 2010.

[8] C. Dwork. Differential Privacy. In M. Bugliesi, B. Preneel, V. Sassone, and I. Wegener, editors, *Automata, Languages and Programming*, volume 4052, chapter 1, pages 1–12. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[9] J. Freudiger, R. Shokri, and J.-P. Hubaux. Evaluating the privacy risk of location-based services. In *Financial Cryptography and Data Security (FC)*, 2011.

[10] P. Golle and K. Partridge. On the anonymity of home/work location pairs. In *Pervasive '09: Proceedings of the 7th International Conference on Pervasive Computing*, pages 390–397, Berlin, Heidelberg, 2009. Springer-Verlag.

[11] B. Hoh and M. Gruteser. Protecting location privacy through path confusion. In *SECURECOMM '05: Proceedings of the First International Conference on Security and Privacy for Emerging Areas in Communications Networks*, pages 194–205, Washington, DC, USA, 2005. IEEE Computer Society.

[12] B. Hoh, M. Gruteser, H. Xiong, and A. Alrabady. Enhancing security and privacy in traffic-monitoring systems. *IEEE Pervasive Computing*, 5(4):38–46, 2006.

[13] J. Krumm. Inference attacks on location tracks. In *In Proceedings of the Fifth International Conference on Pervasive Computing (Pervasive), volume 4480 of LNCS*, pages 127–143. Springer-Verlag, 2007.

[14] J. Krumm. A survey of computational location privacy. *Personal Ubiquitous Comput.*, 13(6):391–399, 2009.

[15] L. Liao, D. J. Patterson, D. Fox, and H. Kautz. Learning and inferring transportation routines. *Artif. Intell.*, 171:311–331, April 2007.

[16] C. Y. Ma, D. K. Yau, N. K. Yip, and N. S. Rao. Privacy vulnerability of published anonymous mobility traces. In *Proceedings of the sixteenth annual international conference on Mobile computing and networking*, MobiCom '10, pages 185–196, New York, NY, USA, 2010. ACM.

[17] M. Piorkowski, N. Sarafijanovic-Djukic, and M. Grossglauser. CRAWDAD data set epfl/mobility (v. 2009-02-24). Downloaded from http://crawdad.cs.dartmouth.edu/epfl/mobility.

[18] L. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[19] D. Reid. An algorithm for tracking multiple targets. *IEEE Transactions on Automatic Control*, 24(6):843–854, 1979.

[20] C. Robert, G. Celeux, and J. Diebolt. Bayesian estimation of hidden Markov chains: A stochastic implementation. *Statistics & Probability Letters*, 16(1):77–83, 1993.

[21] R. Shokri, J. Freudiger, and J.-P. Hubaux. A unified framework for location privacy. In *3rd Hot Topics in Privacy Enhancing Technologies (HotPETs)*, 2010.

[22] R. Shokri, J. Freudiger, M. Jadliwala, and J.-P. Hubaux. A distortion-based metric for location privacy. In *WPES '09: Proceedings of the 8th ACM workshop on Privacy in the electronic society*, pages 21–30, New York, NY, USA, 2009.

[23] R. Shokri, C. Troncoso, C. Diaz, J. Freudiger, and J.-P. Hubaux. Unraveling an old cloak: k-anonymity for location privacy. In *Proceedings of the 9th annual ACM workshop on Privacy in the electronic society*, WPES '10, pages 115–118, New York, NY, USA, 2010. ACM.

[24] C. Troncoso and G. Danezis. The bayesian traffic analysis of mix networks. In *Proceedings of the 16th ACM conference on Computer and communications security*, CCS '09, pages 369–379, New York, NY, USA, 2009. ACM.

[25] S. Vaudenay. On privacy models for rfid. In *Proceedings of the Advances in Crypotology 13th international conference on Theory and application of cryptology and information security*, ASIACRYPT'07, pages 68–87, 2007.