THE UNIVERSITY of EDINBURGH

Edinburgh Research Explorer

On Choosing Between Privacy Preservation Mechanisms for Mobile Trajectory Data Sharing

Citation for published version:
Singh, R, Theodorakopoulos, G, Marina, MK & Arapinis, M 2018, On Choosing Between Privacy Preservation Mechanisms for Mobile Trajectory Data Sharing. in *2018 IEEE Conference on Communications and Network Security (CNS).* Institute of Electrical and Electronics Engineers (IEEE), Beijing, China, 6th Annual IEEE Conference on Communications and Network Security, Beijing, China, 30/05/18. https://doi.org/10.1109/CNS.2018.8433178

Digital Object Identifier (DOI):
10.1109/CNS.2018.8433178

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
2018 IEEE Conference on Communications and Network Security (CNS)

General rights
Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.

OPEN ACCESS

Download date: 02. Mar. 2021

On Choosing Between Privacy Preservation Mechanisms for Mobile Trajectory Data Sharing

Rajkarn Singh, George Theodorakopoulos[†], Mahesh K. Marina and Myrto Arapinis
The University of Edinburgh, [†]Cardiff University
Email: r.singh@ed.ac.uk, TheodorakopoulosG@cardiff.ac.uk, mahesh@ed.ac.uk, marapini@inf.ed.ac.uk

***Abstract*—Various notions of privacy preservation have been proposed for mobile trajectory data sharing/publication. The privacy guarantees provided by these approaches are theoretically very different and cannot be directly compared against each other. They are motivated by different adversary models, making varying assumptions about adversary's background knowledge and intention. A clear comparison between existing mechanisms is missing, making it difficult when a data aggregator/owner needs to pick a mechanism for a given application scenario. We seek to fill this gap by proposing a measure called STRAP that allows comparison of different trajectory privacy mechanisms on a common scale. We also study the trade-off between privacy and utility i.e., how different mechanisms perform when utility constraints are imposed over them. Using STRAP over two real mobile trajectory datasets, we compare state of the art mechanisms for trajectory data privacy and demonstrate the value of the proposed measure.**

I. Introduction

The widespread adoption of smart mobile devices equipped with a multitude of sensors offers a rich source of data on user patterns for mobile app and network service providers. Sharing of mobile app data to third parties (e.g., advertisers) is attractive for developers to monetize their apps [1]. Equally, sharing of mobile subscriber data by network providers offers them revenue generation opportunities and enables a variety of use cases [2]. Moreover, open publication of such datasets can be highly valuable for research purposes. Unscrupulous sharing or publication of such datasets risks violating the privacy of individual mobile users whose data figures in those datasets. In this paper, we focus our attention on the issue of privacy preserving sharing/publication of mobile trajectory data. This is due to two reasons. First, mobile user location data over time (i.e., trajectories) is perhaps the most easily obtainable user contextual data for app/service providers. Second, mobile trajectory data when aggregated across users and mined offers rich insight on collective spatio-temporal patterns of mobile users in a region.

High level of uniqueness inherent to human mobility traces [3] makes them vulnerable to re-identification and harder to ensure user privacy. In recent years, a number of privacy preservation mechanisms have been proposed for trajectory data sharing. Approaches that rely on some form of anonymization (e.g., [1], [4], [5]) have been found to be insufficient for this purpose with or without utility constraints. This in turn has led to more advanced mechanisms employing approaches like differential privacy (DP) (e.g., [6]), k-Anonymity (e.g., [2], [7]) and Plausible Deniability [8], [9]. However, the theoretical guarantees provided by these different privacy mechanisms are very different and not directly comparable against each other. For each mechanism, the notion of privacy is based on what characteristics of the available data they preserve, and use this characteristic as the criteria for their privacy definition. Moreover, different mechanisms differ in their adversary and attack models, making them very different from each other. This makes it really difficult for a data aggregator/owner (e.g., app/service provider) to choose between different mechanisms towards privacy preserving trajectory data sharing/publication.

In this paper, we aim to quantify privacy offered by different trajectory privacy preservation mechanisms (TPPMs) on a common scale, for the first time, so as to enable comparison across them. Specifically, we propose *STRAP (Scale for TRAjectory Privacy)*, a novel metric that can be used to assess the relative privacy guarantees provided by different TPPMs. We also use STRAP to study how constraining utility affects the privacy achieved by different mechanisms. The key idea underlying STRAP is the observation that even though different TPPMs have different theoretical privacy models, all of them perform obfuscation in a way that tries to reduce similarity between the original trajectories and their replications in the output database while also keeping in mind the uniqueness of the users. We use this common ground as a means to compare different mechanisms by having the STRAP measure consider both *similarity/distance* (between input and output trajectory datasets) and *uniqueness* of the users in the original dataset.

The contributions of this paper can be summarized as follows:

- We propose a new privacy metric for assessing trajectory privacy, STRAP, that can be used as a common measure to evaluate and cross-compare privacy imparted by different TPPMs. We base STRAP on distance and uniqueness factors of the database, as well as incorporate the background knowledge of adversary.
- We develop a novel way of computing trajectory distance by leveraging the idea of trajectory segmentation. This is based on the intuition that not all parts of a user's trajectory are equally vulnerable to privacy threats. Hence, we divide a trajectory into different segments and evaluate the privacy of each segment separately.
- We also study the privacy-utility tradeoff of different TPPMs i.e., for a fixed utility level to be achieved, how the different approaches compare against each other in terms of privacy.

The rest of the paper is organized as follows. In Section II, we present relevant background and discuss related work. In Section III, we present the system model. In Section IV, we give an overview of the proposed STRAP metric along with its design rationale. Sections V, VI and VII focus on different components that make up STRAP. Evaluation results focusing on using STRAP for privacy comparison between different representative TPPMs are presented in Section VIII. Section IX concludes the paper.

II. Background and Related Work

A. Trajectory Privacy Preservation Mechanisms (TPPMs)

Location and trajectory privacy has received a fair amount of attention in the literature with various different forms of solution approaches (see [10], for example). From a privacy preserving trajectory data sharing standpoint, which is the focus of this paper, the state of the art approaches fall under one of the three categories outlined below.

1) Differential Privacy (DP) based Mechanisms: DP aims at hiding the presence/absence of a user in the database. Although classical DP restricts to only publishing query results and not database itself, few recent works have used DP for publishing sanitized database by generating a model of the given data by asking relevant queries [11]. Chen et al. [12] adopt a prefix tree approach and fit a noisy Markov model based on DP to generate synthetic output trajectories. Developing this idea further, He et al. [6] propose a hierarchical reference system called DPT to increase the scope of generated synthetic trajectories incorporating varying user speeds and large spatial domain.

2) K-Anonymity based Mechanisms: This class of approaches inherently follow the data publishing model. While optimal solution for k-Anonymity is known to be NP-hard [13], approximate solutions based on this approach do exist. For example, Gramaglia et al. [7] propose an approximate solution called GLOVE for full length user trajectories where the aim is to modify user location information in such a way that atleast k people in the database have similar trajectories; however, this solution is expensive and is based on a heuristic measure of anonymizability. Follow-on work by the same authors [2] provides an optimal solution to a constrained variant of k-Anonymity where they restrict the background knowledge of the attacker to a fixed time interval.

3) Plausible Deniability (PD) [8]: This is the most recently proposed approach for privacy preserving data/trajectory sharing. The idea here is to ensure that the output synthetic trajectory derived from a given original user trajectory could also have been derived from at least $k-1$ other input trajectories [9], thus reducing the probability of linking back to the original user trajectory.

B. Evaluation and Comparison of TPPMs

Shokri et al. [14] propose a framework to quantify location privacy by mounting different attacks, and applying statistical inference tools to re-establish the user identity. However the focus of their work is limited to simple cloaking and generalization based privacy preservation mechanisms.

Cormode et al. [15] propose the notion of empirical privacy which they quantify as the fraction of database tuples for which sensitive information cannot be predicted correctly. Their privacy is based on the output of a fixed set of queries that captures the similarity between input and output database records. Since their privacy is query dependent, its value may change if different queries are used for evaluating privacy.

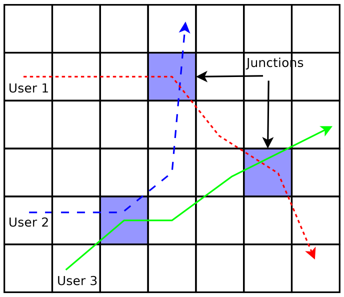


Fig. 1: Region divided into cells, some of which are junctions (colored cells in the figure).

Almasi et al. [16] evaluate the disclosure risk in terms of uniqueness of the original record, and also study the privacy-utility tradeoff. They quantify privacy only in terms of uniqueness of record in the original database, and do not capture the record's similarity to records in the output sanitized database. Note that the work by Shokri et al. [14] focuses on trajectory data, while the work of Cormode et al. [15] and Almasi et al. [16] is applicable for any/generic database.

Although existing works try to quantify privacy, none of them rigorously compare different state of the art TPPMs. Moreover, for assessing privacy they either make use of similarity (between input and output databases) or uniqueness. As we argue in section IV, both these aspects need to be taken into account and this in turn motivates the design of our STRAP metric that captures the combined effect of these aspects.

III. System Model

Consider a database D_{in} consisting of m original user trajectories. This database is passed as input to a TPPM that obfuscates trajectories and produces an output mobility database D_{out} of the same or different number of users. D_{out} can be represented as a function of input database, D_{in}, and the privacy mechanism, $\mathcal{M}$, used with privacy parameter α, i.e.

$$D_{out} = \mathcal{F}_1(D_{in}, \mathcal{M}_\alpha) \quad (1)$$

Each record (row) of input database ($a_i \in D_{in}$) or output database ($o_i \in D_{out}$) represents i_{th} user trajectory i.e., the path traversed by user i over time. Mathematically, $a_i = ((l_1, t_1), (l_2, t_2), \ldots, (l_n, t_n))$ where $l_k = (x_k, y_k) \in \mathbb{R}^2$ is the location in two dimensional space, $t_k \in \mathbb{N}\ \forall k \in [1 \ldots n]$ is the time instance when location l_k is observed, and n is the length of the trajectory a_i.

A. Division of Region into Cells

We divide entire region, $\mathcal{R}$, into smaller sub-regions called cells ($\mathcal{R} = \{c_1 \cup c_2 \cup \ldots\}$). We call a subset of these cells as *junctions* ($\mathcal{J} = \{j_1, j_2, \ldots\} \subseteq \mathcal{R}$), where multiple trajectories intersect in space i.e., multiple user paths converge into or diverge out of these junctions. Fig. 1 shows an example scenario where three user trajectories are intersecting to form three junctions. As a special case, we also consider the cells that contain start and end locations of trajectories as junctions.

Many TPPMs use interchanging or swapping of trajectories at cells with high user density (i.e., junctions) to impart privacy as this preserves overall mobility statistics. When such mechanisms are included in the comparative assessment, using the notion of junctions is helpful in properly quantifying trajectory distance. Use of junctions in our framework has additional benefits like performance enhancement as the computation is based on a subset of all cells in the region. Moreover, since junctions are used for trajectory segmentation (explained in Section VI-C), the path traversed between two junctions can also provide the knowledge of direction of user movement.

B. Adversary Model

Adversary's model is generally quantified by two main characteristics: a) adversary background knowledge, and b) adversary attack or intentions. We model adversary's background knowledge as coming from two different sources. First, partial knowledge of user trajectories. We assume that adversary knows a subset of locations of each user trajectory. This knowledge is represented as the adversary database (D_{adv}). Second, adversary also has access to the entire output database, D_{out}, published after sanitization. We additionally assume that junctions are a common knowledge among database users as well as the adversary and can be obtained by inspecting the region of interest.

Regarding the adversary attack, we assume that adversary is interested in reconstructing the trajectory of a user by looking at the obfuscated set of trajectories in the sanitized (output) dataset. This is called *reconstruction attack* [14]. The adversary makes use of D_{adv}, and D_{out} for performing this attack.

C. Characteristics of Privacy Metric

The metric that we propose compares original database against the database generated by using TPPM and aims to identify the similarity between their records. We assign a privacy value to each input record, and then compute the final privacy measure called STRAP.

To avoid any bias towards a particular TPPM, it is essential for a truly independent metric to be oblivious to the privacy mechanism used. Hence, our metric design does not rely on the knowledge of mechanism used for privacy preservation. The privacy level, Δ, measured with STRAP is only a function of the original database, output database and the adversary's knowledge, i.e.,

$$PrivacyLevel,\ \Delta = \mathcal{F}_2(D_{in}, D_{out}, D_{adv}) \tag{2}$$

D. Utility Constraint Based Comparison

Different TPPMs can be judged independent of each other by varying their privacy parameters. However to understand how they compare against each other, we fix a desired utility level to be achieved and use this utility level as a constraint to be satisfied by each mechanism.

We use a workload of range count queries as a measure of TPPM utility. Let Q be the set of queries. Each range count query ($q \in Q$) returns the number of footfalls or the number of timestamped locations found within a specified latitude and longitude range, over a given time period.

To measure utility, we use the widely adopted method used in earlier works [17], [18]. We measure utility of each query ($q \in Q$) as the relative error in its answer when applied to the sanitized database (D_{out}) with respect to the original database (D_{in}), i.e.,

$$error(q) = \frac{|q(D_{out}) - q(D_{in})|}{q(D_{in})} \tag{3}$$

For the entire query workload (Q), we use average relative error over all queries in the workload, i.e.,

$$error(Q) = \frac{\sum_{q \in Q} error(q)}{|Q|} \tag{4}$$

IV. STRAP Design Rationale and Overview

A. Relation between Uniqueness and Privacy

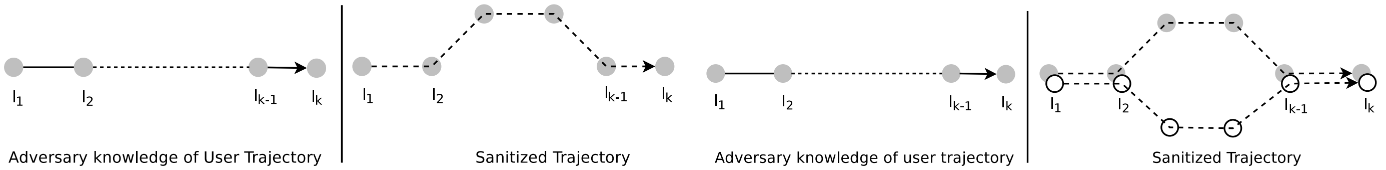
For a given trajectory, its uniqueness is defined as the number of trajectories in its close vicinity. Uniqueness is a characteristic of the database that captures how much vulnerable database records are to a re-identification attack. This idea is strongly conveyed in [3] in which authors show that human mobility traces are highly unique that even four different location points can distinguish an individual's trajectory from others. The solution approach of k-Anonymity [19] focuses to reduce this uniqueness in user attributes via generalization and suppression. Hence we base our mechanism on the uniqueness of records in the original database. Note that uniqueness in output database may not matter as the unique output trajectories may not be linked back to any of the original trajectories, hence not affecting database privacy.

To back our argument of using uniqueness as a factor in STRAP, we examine the scenario of Fig. 2. Considering the adversary model as presented in Section III-B, let us assume that adversary knows locations l_1, l_2, l_{k-1} and l_k of a user u and wants to find its remaining locations. Let us first take the case when no privacy preservation is used and the database is published in its original form. If user u is unique, the adversary will be able to match its known locations to the exact locations in the output, thereby re-constructing the entire user trajectory (Fig. 2(a)), implying that high uniqueness results in reduced privacy. Now consider the scenario when user u is not unique (again without any sanitization), and another user u' in D_{in} has the same four locations l_1, l_2, l_{k-1} and l_k (Fig. 2(b)). In this case, the adversary will be confused between two locations for l_{k-2} not being sure which one belongs to user u. Thus, as uniqueness reduces, the chances of user re-identification also reduces, implying higher privacy.

B. Relation between Trajectory Distance and Privacy

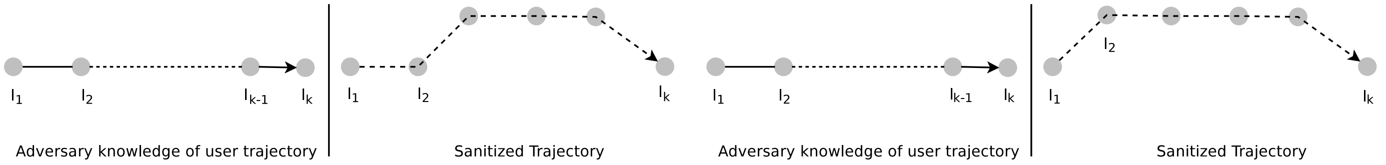
The desirable property that privacy preserving mechanisms try to achieve is that an adversary is unable to link a published sanitized trajectory to an original user trajectory. To achieve this, TPPMs perform obfuscation by adding noise thereby changing the true values of database attributes. We capture this obfuscation by measuring geographical distance between the actual trajectory of a user and corresponding output trajectories, thus also base our privacy metric on their distance. This distance provides an index of similarity [15], [16] and helps in determining the confidence level with which original attribute values can be inferred from the output.

We argue for the use of distance in the same way as for uniqueness. After trajectory obfuscation, output database will



(a) Unique trajectory $\implies$ lower privacy (b) Non-unique trajectory $\implies$ higher privacy

Fig. 2: Scenario depicting how trajectory uniqueness affects privacy


l1
l2
lk-1
lk
Adversary knowledge of user trajectory
l1
l2
lk
Sanitized Trajectory
l1
l2
lk-1
lk
Adversary knowledge of user trajectory
l2
l1
lk
Sanitized Trajectory

(a) Shorter distance $\implies$ lower privacy (b) Longer distance $\implies$ higher privacy

Fig. 3: Scenario depicting how distance between input and output trajectories affects privacy

contain noisy locations which may be different from original locations. The number of matched locations will depend upon the level of noise added. For a smaller value of noise being added, distance from original trajectory will be shorter (Fig. 3(a)). Here, locations l_1, l_2 and l_k match providing adversary with higher confidence about user's remaining locations, implying lower privacy. On the other hand, a higher distance from original trajectory will result in a small number of matched locations (for example in Fig. 3(b) only two locations l_1 and l_k match), implying higher privacy.

C. Solution Overview

In view of the above discussion, we base STRAP on two different factors: (i) *expected distance* ($\mathbb{ED}$) of an input trajectory from the output database trajectories – this captures the similarity of an original record against output records (Section VI) and (ii) *uniqueness* ($\mathcal{U}$) of the input trajectories – this captures the degree of identifiability of a user in a given set of original records (Section VII).

Note that higher expected distance of a trajectory means that it is difficult to reconstruct the input trajectory given the sanitized output, hence higher privacy. Higher uniqueness, on the other hand, can lead to easy identifiability of a record, implying lower privacy. Based on these two factors, STRAP computes the privacy level (Δ_u) for each user trajectory, a_u, as follows:

$$\Delta_u = \frac{\mathbb{ED}(a_u)}{\mathcal{U}_u} \tag{5}$$

Higher the value of STRAP (Δ), more private is the data record. It is also important to emphasize that our method does not quantify the absolute privacy achieved by a TPPM (i.e., whether the adversary has been successful in its attack) but it quantifies the relative privacy levels of different TPPMs.

Our solution is divided into three main steps as described below:

1) Develop a common mobility model that encodes the adversary's knowledge of user mobility:
   1.1 Build a Markov model from adversary's knowledge of input trajectories.
   1.2 Build a Markov model of output database.
   1.3 Combine these two models into one model.
2) Compute segmentation based trajectory distance using the mobility model developed in the first step:
   2.1 Segment each input and output user trajectory based on junctions.
   2.2 Calculate expected distance of all input trajectories by computing segment distances.
3) Compute trajectory uniqueness and the final metric value:
   3.1 Find uniqueness of each user trajectory relative to other users in the input dataset.
   3.2 Use uniqueness as the weight to compute final privacy metric, STRAP, for the database.

Section V describes the first step i.e., building the adversary's knowledge model. Trajectory distance measure is explained in Section VI, while trajectory uniqueness and final metric computation are presented in Section VII.

V. Adversary Knowledge and User Profile

As mentioned earlier, an adversary has two sources of knowledge 1) partial user trajectories (D_{adv}) and 2) complete published database (D_{out}). It first builds the input trajectory based profile (P^u) for each user u from D_{adv}. Adversary also develops a common profile of transition probabilities (P^{out}) for the output database, and then combines these two to build the final user mobility model.

A. Input Database Based User Profiles

Consider that each user location l_i is known to adversary with a probability p. This provides adversary with partial trajectories of each user (D_{adv}). Using D_{adv}, we construct a matrix of transition probabilities, P^u, for each user u following the first order Markov model. This matrix contains the probability of transitioning from one junction to another, where each junction is a state in our Markov model and the missing states are determined using Gibbs Sampling [20].

B. Knowledge gain from Output Database

A second order Markov model based transition probability matrix, P^{out}, is constructed from the output database (D_{out}). We use second order Markov model as it better captures the direction of movement of the user. Using a second order Markov model for D_{adv} does not work well, as adversary only knows a limited number of user locations (each with probability p) making the second order model very sparse.

Moreover, we compute transitions only against junctions $\mathcal{J}$ (and not all cells) since junctions are the regions where multiple users meet and a synthetic user trajectory path can be swapped at junctions to differ from an original trajectory. Also, transitions across junctions suffice to capture the direction of user movement using a second order Markov model. The size of profile matrix for each user, P^u, is $(|\mathcal{J}| \times |\mathcal{J}|)$, and the size of output database matrix, P^{out}, is $(|\mathcal{J}|^2 \times |\mathcal{J}|)$, where $(|\cdot|)$ represents cardinality of the set.

C. Overall User Transition Probabilities

Since user profile encodes incomplete information of user movements, an adversary is never completely certain about actual user trajectory. We capture this uncertainty in the adversary's belief of P^u by means of Shannon's Entropy [21]. Specifically, for each user u, we calculate normalized entropy $(H^u_{j_i})$ of transition probabilities for each junction, $j_i \in \mathcal{J}$ i.e., each row of P^u matrix as,

$$H^u_{j_i} = -\frac{\sum_{j_k \in \mathcal{J}} P^u_{j_i,j_k} \log_2(P^u_{j_i,j_k})}{\log_2(|\mathcal{J}|)} \tag{6}$$

where $P^u_{j_i,j_k}$ is the transition probability of user u from junction j_i to junction j_k, $|\mathcal{J}|$ represents the number of transition states. Dividing by $\log_2(|\mathcal{J}|)$ normalizes the value of entropy between 0 and 1.

Now, we factor in the knowledge obtained from output database (P^{out}) depending upon the degree of uncertainty of user profile, P^u. A higher value of $H^u_{j_i}$ (≈ 1) indicates more uncertainty in P^u i.e., adversary's confidence on P^u is lower. Hence give more weightage to P^{out} transition matrix. We use entropy to provide appropriate weights to P^u and P^{out} and compute their weighted average as,

$$W^u_{j_i} = (1 - H^u_{j_i})P^u_{j_i} + (H^u_{j_i})P^{out}_{j_i}, \forall j_i \in \mathcal{J} \tag{7}$$

where $P^u_{j_i}$ (and $P^{out}_{j_i}$) is j_i row of P^u (and P^{out}), W^u is the final transition probability knowledge matrix of user u that adversary uses to compute expected trajectory distance.

It is worth mentioning here that since we use second order Markov model for D_{out}, this can be an issue when the dataset is sparse. In that case, the Markov model built will not be based on a rich set of transitions and hence not very reliable. For such databases, we can revert back to using first order Markov model for P^{out}. Though we will have a disadvantage on directionality, but it can get compensated by using the richer Markov model with fewer transitions.

VI. Trajectory Distance Measure

In this section, we aim to develop a means to compute similarity between D_{in} and D_{out}. To do this, we propose a novel way of computing trajectory distances by exploiting trajectory segmentation. First we describe why simple distance measure between two full trajectories does not work. Then we describe our method of trajectory segmentation.

A. Geographic Distance Between Trajectories

To capture geographic similarity between two trajectories (or trajectory segments), we use Dynamic Time Warping (DTW) [22]. This distance measure is widely used by data mining community for a variety of time series related tasks [23]. DTW finds a mapping between two series such that total distance between the points on the mapped path is minimized. Considering that timestamped location points form a time series, we use DTW to find distance between them. We chose DTW over other measures because TPPMs also modify the timestamp of locations along with modifying the location data, and DTW captures this time difference very nicely by detecting phase shifts in time series. Other measures like Euclidean distance are not able to capture obfuscation along time dimension. For example, two exactly same paths traversed at different times of the day can have zero or non-zero distance depending upon the time lag window chosen by DTW.

Consider two trajectories X and Y consisting of location-time tuples. The normalized DTW distance between X and Y is given as,

$$dist(X,Y) = DTW(X,Y)/\max(|X|,|Y|) \tag{8}$$

where $|X|$ (and $|Y|$) is the number of timestamped locations in trajectory X (and Y). Note that $dist(X,Y)$ is non-negative, a higher value of $dist(X,Y)$ represents lower similarity between trajectories X and Y. $DTW(X,Y)$ between two trajectories, X and Y, is defined based on [23] as,

$$DTW = \begin{cases} 0, & \text{if } |X| = |Y| = 0 \\ \infty, & \text{if } |X| = 0 \text{ or } |Y| = 0 \\ d_{eu}(x_1,y_1) + \\ min\{DTW(Rest(X), \\ Rest(Y)), \\ DTW(Rest(X),Y), \\ DTW(X,Rest(Y))\}, & \text{otherwise} \end{cases}$$

Here $d_{eu}(x_i,y_i)$ is the Euclidean distance between location points x_i and y_i. Most modern implementations of DTW have the time complexity of $O(\max(|X|,|Y|))$.

B. Why Full Trajectory Comparison Does Not Work

Here we argue that the intuitive way of judging trajectory similarity i.e., by calculating trajectory distance of full lengths is insufficient. Let us assume that the closest match of a trajectory $a_u \in D_{in}$, that has the minimum distance from a_u, is $o_u \in D_{out}$, i.e.,

$$o_u = \underset{o_k \in D_{out}}{\arg\min}\ dist(a_u,o_k), \quad a_u \in D_{in} \tag{9}$$

This way of obtaining the closest trajectory will only be effective when each input trajectory is obfuscated independently. For the advanced mechanisms involving synthetic trajectory generation based on common database characteristics, this will not work. Similarly, this measure will not be effective for swapping or intersection based techniques. To clarify this further, consider the scenario of Fig. 4, where two trajectories, u_1 and u_2, start from cells (or junctions) j_1 and j_5 respectively. After the application of privacy mechanism, their start locations remain the same however the final locations get interchanged.

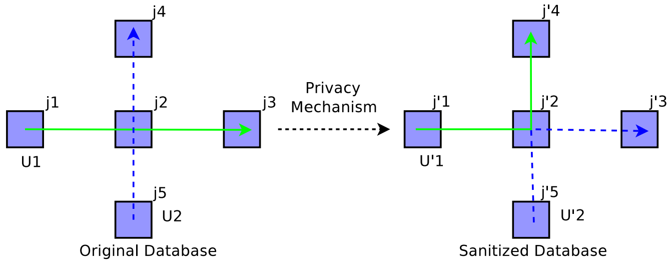


Fig. 4: Scenario showing full trajectory comparison is ineffective.

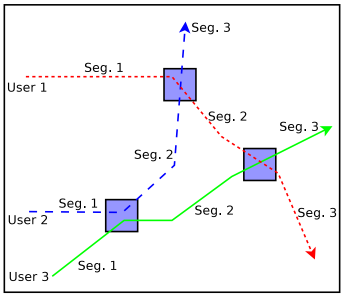


Fig. 5: Illustration of dividing trajectories into segments at junctions.

A simple distance calculation of u_1 and u_2 (using eq. 8), from either of the output trajectories, u'_1 and u'_2, returns a non-zero distance value. However, we can see that each user's partial trajectory segment is replicated in the output database thereby revealing sensitive user locations. Hence, distance metric should return a zero value in this case. This suggests that full length distance calculation will not be able to capture such scenarios.

C. *Trajectory Segmentation*

Use of segments in trajectory distance computation is beneficial in 1) detecting trajectory overlap based on junctions, and 2) comparing a long input trajectory with a relatively shorter output trajectory. Second point is significant because we observed in our experiments that some privacy mechanisms produce very small length output trajectories to privatize original trajectories. Computing full length trajectory distance in such cases produces misleading results.

We partition each user trajectory into smaller segments based on junctions $\mathcal{J}$. First we identify the locations in user trajectories that pass through junctions. Then we partition each user trajectory a_u into segments from the locations where they pass through junctions. Let S^u be the set of trajectory segments of a_u i.e., $S^u = \{s^u_1, s^u_2, \ldots\}$. Fig. 5 shows three trajectories, each divided into three segments since they pass through two junctions. Such segmented trajectories are created for all users in D_{in} and D_{out}.

Once trajectories in D_{in} and D_{out} are segmented, we compute expected distance of each input trajectory segment (Section VI-D). We then perform normalized addition of the expected distances of all trajectory segments of a user to obtain its distance from the closest matched output trajectories.

D. *Expected Segment and Trajectory Distance*

Selecting each segment from input trajectory and finding its distance from the closest segment in the output database is not sufficient. To explain this further, again consider the scenario of Fig. 4. Distance of trajectory segment $s_1 = (j_1, j_2)$ in the input database from the segment $s'_1 = (j'_1, j'_2)$ in the output database will be zero. Similarly, distance of input trajectory segment $s_2 = (j_2, j_3)$ from $s'_2 = (j'_2, j'_3)$ will also be zero. This way, total distance of trajectory u_1, which is equal to the sum of its segment distances, will be zero. This zero distance implies low or no privacy i.e., adversary can easily find correct user locations. However, this is not true as the segment s_2 of the input trajectory u_1 is clearly being compared with an entirely different user trajectory's (i.e., u_2) segment.

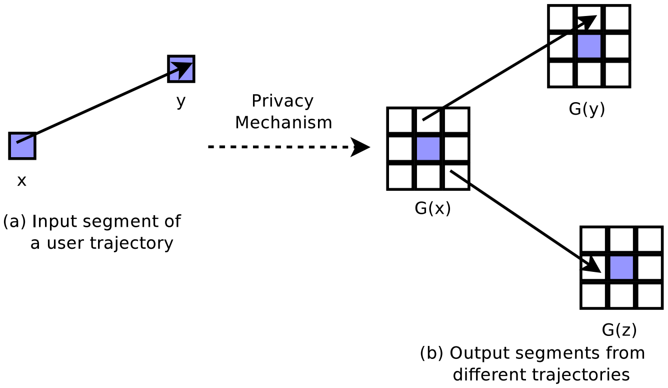
To address this issue, we use knowledge matrix of attacker (W^u, eq. 7) to find the expected distance of each input trajectory from the output database in a probabilistic manner. Consider a user trajectory a_u divided into k segments, $S^u = \{s^u_1, s^u_2, \ldots s^u_k\}$. Let the i_{th} segment s_i of trajectory a_u starts from junction x and ends at junction y (i.e., $s^u_i = s^u_{x,y}$). Let $G(x)$ be the set of *cells* in the immediate vicinity of junction x including x itself. We call $G(x)$ the neighborhood of x. $G(x)$ represents a coarser region around the junction x as shown in Fig. 6. We define *expected distance* ($\mathbb{ED}$) of input segment s^u_i starting at junction x as the weighted average distance of s^u_i from all output segments starting at junctions $G(x)$. Weights for averaging are derived from adversary knowledge matrix W^u. Mathematically,

$$\mathbb{ED}(s^u_{x,y}) = \sum_{(x',z') \in S_{G(x),G(z)}; z \in \mathcal{J}} w^u_{x,z'} \cdot dist(s^u_{x,y}, s_{x',z'})$$

where $w^u_{x,z'}$ is the transition probability of user u from junction x to z'. $S_{G(x),G(z)}$ is the set of all user segments that start from neighborhood of x i.e., $G(x)$ and end at neighborhood of z i.e., $G(z)$ in the output database. Also, note that $s^u_{x,y}$ is the user segment of D_{in}, while $s_{x',y'}$ is from D_{out}.

The reason we use generalized neighborhood $G(\cdot)$ of junctions is that when privacy mechanisms obfuscate trajectories, junctions may not appear at exactly the same cells but in nearby cells. Therefore, a user segment in input may not pass through exactly same junctions in output, but from the cells that are in the neighborhood of those junctions. By exploring all segments starting from neighboring cells, we make sure none of those cases are left unnoticed.

Now we compute the final expected distance for a user's full trajectory a_u as the normalized sum of expected distances



(a) Input segment of a user trajectory

(b) Output segments from different trajectories

Fig. 6: (a) Segment in D_{in} passing through junctions (x, y), (b) Segments in D_{out} passing through neighborhood $(G(x), G(y))$ of (x,y) and neighborhood $(G(x), G(z))$ of (x,z).

of corresponding trajectory segments. Expected distance can be zero for some of the trajectory segments, either because an exact sequence of locations in the output segment is found or because no segment originated from the neighborhood $G(x)$ in D_{out}. The latter type of segments imply higher privacy since no output segment matched with them. We identify such segments (S^u_{zero}) and subtract their count from total number of segments in the trajectory ($|S^u|$). We use this difference as the normalization factor for our final distance value computation. Thus the total expected distance of trajectory a_u is computed as follows,

$$\mathbb{ED}(a_u) = \frac{\sum_{s_i \in S^u} \mathbb{ED}(s^u_i)}{|S^u| - |S^u_{zero}|} \tag{10}$$

VII. Trajectory Uniqueness and The Final Privacy Metric

A. Trajectory Uniqueness Measure

To illustrate why including uniqueness in privacy design metric is necessary, consider two trajectories a_1 and a_2 in Fig. 7 where each trajectory is represented as a dot in n-dimensional space. Here a_1 has no trajectory in its vicinity while there are three other trajectories in the neighborhood of a_2, hence a_1 is more unique than a_2. Consider that both a_1 and a_2 are obfuscated by adding the equal amount of noise to their respective locations using the same TPPM. Since a_1 is unique, with some background knowledge an adversary can easily construct the original user trajectory from the obfuscated one. However, adversary will get confused among multiple trajectories that are similar to a_2 while reconstructing its trajectory since it is not unique.

To compute each trajectory uniqueness, we find the number of other trajectory records present within a certain distance from the given trajectory. Let C_u be the number of neighboring trajectories of a_u that are less than d_{th} distance apart from a_u. Let C_{total} be the count of neighboring trajectories of all users in the database, i.e.,

$$C_{total} = \sum_{u \in D_{in}} C_u \tag{11}$$

We compute uniqueness ($\mathcal{U}$) of user u as reciprocal of the count of its neighboring trajectories, normalized by total user count, i.e.,

$$\mathcal{U}_u = \frac{1}{C_u / C_{total}} = \frac{C_{total}}{C_u} \tag{12}$$

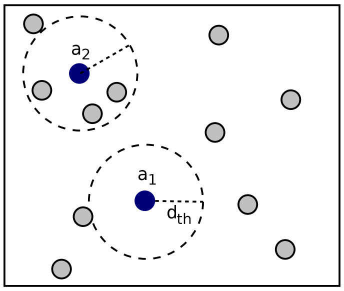


Fig. 7: User trajectories each represented as a dot in n-dimensional space.

Note that the uniqueness value will depend upon d_{th} level chosen. However, as uniqueness is a characteristic of only D_{in}, d_{th} will affect the final STRAP value for all TPPMs in the same relative manner. Hence, any sensible value of d_{th} can be chosen.

Also, one can argue that since uniqueness depends upon D_{in} only and is same across different mechanisms, why is distance not sufficient for STRAP calculation. While this is true when the privacy of a single trajectory is being compared, when multiple trajectories are present in the database, distance only cannot assess privacy of overall database.

B. Final STRAP Metric

Given the expected trajectory distance and uniqueness measure, the final value of privacy metric, STRAP, that represents the relative privacy level of each user u, as given in equation (5), will be,

$$\Delta_u = \frac{\mathbb{ED}(a_u)}{\mathcal{U}_u} = \frac{\sum_{s_i \in S^u} \mathbb{ED}(s^u_i)}{|S^u| - |S^u_{zero}|} \Big/ \frac{C_{total}}{C_u} \tag{13}$$

STRAP value for the whole database can be computed as,

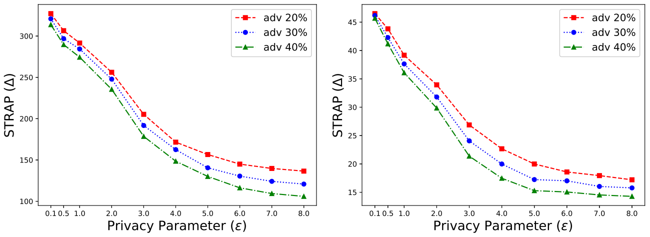
$$\Delta_{D_{in}} = \sum_{u \in D_{in}} \Delta_u \tag{14}$$

In the same way, we can use the STRAP metric to compute the relative privacy achieved by top most 5% or 10% vulnerable trajectories in the database.

If m is the number of trajectories in each of the input and output databases, n is the average length of each trajectory and $\mathcal{J}$ is the number of junctions in the entire region, then runtime complexity for creating Markov models will be $O(mn)$ as this can be obtained by a single parsing of the trajectories. To compute the DTW distance between an input-output pair of trajectories, we need to compute all pairwise DTW distances between each input segment (length $n/|\mathcal{J}|$) and each output segment (length $n/|\mathcal{J}|$). There are maximum $|\mathcal{J}|^2$ such segment pairs, and computing the distance for each pair takes $O(n/|\mathcal{J}|)$ time, so the runtime for distance between the input-output pair of trajectories is $O(n|\mathcal{J}|)$. There are m such trajectories, so the total computation time for expected distance of all trajectories will be $O(mn|\mathcal{J}|)$. Finally, runtime complexity to compute uniqueness of all trajectories is $O(m^2n)$.

VIII. Evaluation

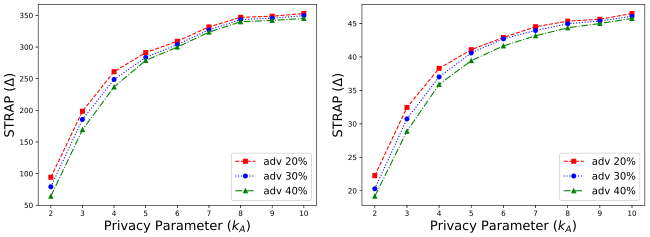
For evaluation purpose, we consider mechanisms based on data publishing model, since it does not restrict the use of data to limited applications. From each of the privacy notions discussed in Section II-A, we select the following state-of-the-art mechanisms. For DP, we use the most recent work that publishes trajectories derived from a differentially private model [6]. Although k-Anonymity is a well established concept, there are not many works that implement k-Anonymity on full location trajectories. We use GLOVE [7] as it is the latest and complete implementation of k-Anonymity on trajectory data. For Plausible Deniability, we use [8] which is an implementation specific to location trajectories. Privacy parameters for DP, k-Anonymity and PD are represented as ϵ, k_A and k_{PD}, respectively.



(a) MDC

(b) EPFL

Fig. 8: Variation of STRAP with change in DP parameter (ϵ)



(a) MDC

(b) EPFL

Fig. 9: Variation of STRAP with change in k-Anonymity (k_A)

A. Datasets

To study the impact of database characteristics on privacy and utility, we use two different mobility datasets described below.

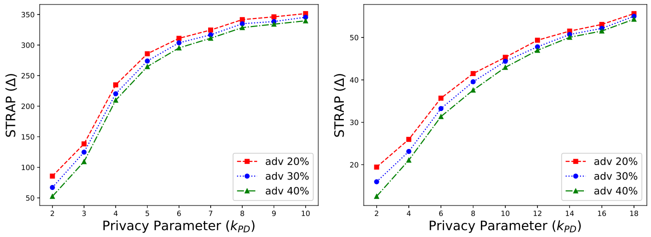
1) Mobile Data Challenge (MDC) Dataset: MDC dataset [24] consists of walking traces of 185 users over the area of Lake Geneva in Switzerland. The location information is provided in the form of GPS coordinates along with timestamp consisting of around 12 million user locations. Using this dataset has a challenge that it is very sparse with user trajectories spanning over a vast region of approximately 512km × 223km. Therefore, we selected 70 users with highest frequency of occurrence in the dataset over a duration of 8 hours. We divide total time into bins of 5 minutes each.

2) EPFL Mobility Dataset: This dataset consists of taxi mobility traces in San Francisco and is openly available from CRAWDAD [25]. The average time interval between two consecutive location updates is less than 10 seconds making the dataset quite fine-grained. We selected top 400 users for the duration of 8 hours timestamped every 5 minutes.

B. Results

1) Varying Privacy Parameter: We plot the variation of STRAP against privacy parameters of DP (ϵ), k-Anonymity (k_A) and PD (k_{PD}) in Fig. 8, 9 and 10 respectively. These figures show a monotonic increase in privacy (STRAP) with an increase in k_A and k_{PD}, and a decrease in ϵ, thus validating our privacy metric is in coherence with standard privacy definitions.

As seen in Fig. 8, lower ϵ values provide higher privacy, indicated by a higher value of STRAP. Similarly, higher ϵ values provide lower privacy, however the fall in privacy reduces as ϵ increases. In Fig. 9 and 10, similar behavior is observed for k-Anonymity (k_A) and Plausible Deniability (k_{PD}), where higher k indicates higher privacy and vice-versa.



(a) MDC

(b) EPFL

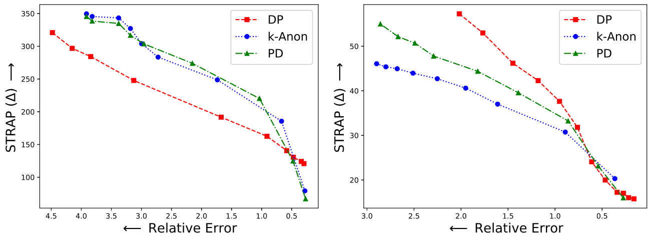
Fig. 10: Variation of STRAP with change in PD parameter (k_{PD})

Moreover, we observe that there is a noticeable affect on privacy with change in adversary knowledge. We varied adversary knowledge percentage (p) as 20%, 30% and 40% of the user trajectory points, and noticed that as adversary knowledge increases, the privacy reduces.

Also, notice that STRAP values are different across different datasets even when the same TPPM is applied (e.g. Fig. 8(a) and 8(b)). Here, lesser value of STRAP for EPFL dataset compared to MDC dataset does not mean that EPFL dataset is less private than MDC. We cannot draw such a comparison using STRAP because its values are calculated based on actual values of database attributes, hence changing the database will return a different metric value, even if the same privacy parameter is used.

2) Cross-Comparison Under Utility Constraints: To be able to cross-compare different mechanisms, we fix a desired utility level to be achieved and find the STRAP value that provided desired utility. We notice that different TPPMs provide very different privacy when we put a utility constraint over them. Fig. 11 shows this privacy-utility trade-off by plotting STRAP along Y-axis against utility along X-axis. We plot relative error (eq. (4)) in reverse on X-axis to better visualize the trade-off (lesser the relative error, higher the utility). Ideally, one would want privacy mechanisms to fall in the top right corner exhibiting both high privacy and utility. However, in reality, as privacy increases, noise in data increases resulting in reduced utility, hence the resulting curves exhibit different behaviors.

We observe when privacy requirements are not stringent, these mechanisms achieve almost similar and higher utility values (bottom right corner in Fig. 11). In other words, if utility of resulting database is the primary focus, all TPPMs exhibit similar performance and impart lower privacy, hence any of these can be chosen. However, as the utility requirements start relaxing, their performance starts differing from each other in terms of the privacy levels achieved (as we move towards top left corner in Fig. 11).



(a) MDC

(b) EPFL

Fig. 11: Privacy Vs Utility Tradeoff - Range Count Queries

In Fig. 11(a) for MDC dataset, we notice that inspite of being known as a strong privacy notion, DP performs worse than k-Anonymity and PD. However, for EPFL dataset, in Fig. 11(b), DP provides higher privacy compared to both k-Anonymity and PD at lower utility constraints. This contrast in DPT behavior is seen because of the difference in characteristics of two datasets used: MDC dataset is very sparse, while EPFL dataset is highly dense. The performance of DPT algorithm depends upon the number of trajectory records, it builds a poor mobility model thereby giving a reduced utility when the number of users is lesser [6]. Among k-Anonymity and PD, there is no obvious winner for MDC dataset, but PD provides slightly higher privacy for most of the utility values for EPFL dataset. Thus, when sparsity is not an issue, PD is preferable over k-Anonymity.

3) Effect of Database Characteristics: Although privacy notions are database oblivious, but the existing algorithms implementing them are highly dependent on database characteristics such as number of records (database size), sparsity of the database (user density), etc. For instance, a trajectory database with dense user distribution loses little in terms of utility when k-Anonymity is applied, as observed in graphs with different datasets. Similarly, as seen in Fig. 11(a), DPT algorithm is vulnerable to this problem to a great extent.

During our experiments we also noticed that the performance of these algorithms in terms of their execution time varies greatly with the database size. We found that DPT algorithm is highly scalable and its execution time did not slow down considerably as the number of users increased. PD algorithm, however, takes a considerable amount of time to build the model of semantic clusters i.e. proportional to the square of all locations in the dataset [8]. Moreover, once trajectories are generated, PD discards most of them as they do not meet the criteria of having been derived from k_{PD} users. As k_{PD} increases, it becomes more time consuming to obtain satisfactory traces leading to prolonged execution time. GLOVE is a greedy algorithm, has quadratic runtime complexity and designed to be highly parallelizable [7], it is much faster than PD.

IX. Conclusion

We have developed a measure called STRAP to evaluate and compare the privacy provided by different TPPMs based on database uniqueness and obfuscation distance. As part of STRAP, we have also proposed a novel way of computing distance between different trajectories. Given a database and an application scenario, our STRAP metric can be used to chose among different TPPMs. Our results show that given the fixed utility constraints, the privacy achieved by different mechanisms is highly dependent on the database used. No single mechanism performed consistently better over others for all types of datasets and applications. Although we developed STRAP in the context of trajectory data, we believe it is straightforward to extend to other types of data via suitable modification of the distance measure used. We leave the detailed examination of this issue for future work.

Acknowledgment

We thank Hugh Leather for his contribution at the early stages of this work. Rajkarn Singh is supported by a PhD studentship under the EPSRC Centre for Doctoral Training in Pervasive Parallelism at the University of Edinburgh.

References

[1] G. Tsoukaneri et al., "On the inference of user paths from anonymized mobility data," in *IEEE European Sym. on Security and Privacy*, 2016.

[2] M. Gramaglia, M. Fiore, A. Tarable, and A. Banchs, "Preserving mobile subscriber privacy in open datasets of spatiotemporal trajectories," in *Proc. of Int. Conf. on Computer Communications, INFOCOM*, 2017.

[3] Y. d. Montjoye et al., "Unique in the crowd: The privacy bounds of human mobility," *Nature Scientific Reports*, 2013.

[4] H. Zang and J. Bolot, "Anonymization of location data does not work: A large-scale measurement study," in *Proceedings of the Int. Conf. on Mobile Computing and Networking (MOBICOM)*, 2011.

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

[6] X. He, G. Cormode, A. Machanavajjhala, C. M. Procopiuc, and D. Srivastava, "DPT: Differentially private trajectory synthesis using hierarchical reference systems," *Proc. of VLDB Endow.*, 2015.

[7] M. Gramaglia and M. Fiore, "Hiding mobile traffic fingerprints with GLOVE," in *Proceedings of ACM CoNEXT*, 2015.

[8] V. Bindschaedler and R. Shokri, "Synthesizing plausible privacy-preserving location traces," in *IEEE Symposium on Security and Privacy (SP)*, 2016.

[9] V. Bindschaedler, R. Shokri, and C. A. Gunter, "Plausible deniability for privacy-preserving data synthesis," *Proc. of VLDB Endowment*, 2017.

[10] C.-Y. Chow and M. F. Mokbel, "Trajectory privacy in location-based services and data publication," *SIGKDD Explor. Newsl.*, 2011.

[11] D. Mir, S. Isaacman, R. Caceres, M. Martonosi, and R. Wright, "Dp-where: Differentially private modeling of human mobility," in *Big Data, 2013 IEEE International Conference on*, 2013.

[12] R. Chen, G. Acs, and C. Castelluccia, "Differentially private sequential data publication via variable-length n-grams," in *Proceedings of the ACM Conference on Computer and Communications Security*, 2012.

[13] A. Meyerson and R. Williams, "On the complexity of optimal k-anonymity," in *Proceedings of the ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, ser. PODS '04, 2004.

[14] R. Shokri et al., "Quantifying location privacy," in *IEEE Symposium on Security and Privacy, (SP)*, 2011.

[15] G. Cormode, C. M. Procopiuc, E. Shen, D. Srivastava, and T. Yu, "Empirical privacy and empirical utility of anonymized data," in *Int. Conf. on Data Engineering Workshops (ICDEW)*, 2013.

[16] M. M. Almasi, T. R. Siddiqui, N. Mohammed, and H. Hemmati, "The risk-utility tradeoff for data privacy models," in *IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, 2016.

[17] X. Xiao, G. Bender, M. Hay, and J. Gehrke, "iReduct: Differential privacy with reduced relative errors," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2011.

[18] X. Xiao, G. Wang, and J. Gehrke, "Differential privacy via wavelet transforms," *IEEE Trans. on Knowledge and Data Engineering*, 2011.

[19] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan, "Mondrian multidimensional k-anonymity," in *Proceedings of International Conference on Data Engineering (ICDE)*, 2006.

[20] C. Robert, G. Celeux, and J. Diebolt, "Bayesian estimation of hidden markov chains: A stochastic implementation," *Statistics and Probability Letters*, 1993.

[21] C. E. Shannon, "A mathematical theory of communication," *ACM SIGMOBILE Computing and Communication Reviews*, 2001.

[22] B.-K. Yi, H. V. Jagadish, and C. Faloutsos, "Efficient retrieval of similar time sequences under time warping," in *Proceedings of International Conference on Data Engineering (ICDE)*, 1998.

[23] E. Keogh and C. A. Ratanamahatana, "Exact indexing of dynamic time warping," *Knowledge and Information Systems*, 2005.

[24] IDIAP and NRC-Lausanne. Mobile Data Challenge (MDC) Dataset., https://www.idiap.ch/dataset/mdc, 2011.

[25] M. Piorkowski, N. Sarafijanovic-Djukic, and M. Grossglauser, "CRAWDAD dataset epfl/mobility (v. 2009-02-24)," https://crawdad.org/epfl/mobility/20090224, 2009.