Location disclosure risks of releasing trajectory distances

Emre Kaplan[1], Mehmet Emre Gursoy[2], Mehmet Ercan Nergiz[3], Yucel Saygin[1]

Abstract

Location tracking devices enable trajectories to be collected for new services and applications such as vehicle tracking and fleet management. While trajectory data is a lucrative source for data analytics, it also contains sensitive and commercially critical information. This has led to the development of systems that enable privacy-preserving computation over trajectory databases, but many of such systems in fact (directly or indirectly) allow an adversary to compute the distance (or similarity) between two trajectories. We show that the use of such systems raises privacy concerns when the adversary has a set of known trajectories. Specifically, given a set of known trajectories and their distances to a private, unknown trajectory, we devise an attack that yields the locations which the private trajectory has visited, with high confidence. The attack can be used to disclose both positive results (i.e., the victim has visited a certain location) and negative results (i.e., the victim has not visited a certain location). Experiments on real and synthetic datasets demonstrate the accuracy of our attack.

Keywords: Privacy, spatio-temporal data, trajectory data, data mining.

1. Introduction

Location tracking devices such as GPS-equipped vehicles, smartphones and location-based applications have greatly eased the collection of spatio-temporal movement patterns and trajectories. The analysis of this data can help the society, e.g., via traffic management in metropolitan areas, discovery of traffic and passenger flows [1][2], road condition sensing [3] and fleet management. While sharing and mining trajectory data is beneficial for the society, the sensitive nature of location data raises privacy concerns. This has led to substantial research in location privacy [4][5] and privacy-preserving trajectory data management [6][7][8].

Work in the latter can be roughly divided into two camps. In the first camp, data is de-identified and anonymized (e.g., by removing explicit identifiers and

[1]Faculty of Engineering and Natural Sciences, Sabanci University, Istanbul, Turkey.
[2]College of Computing, Georgia Institute of Technology, Atlanta, GA.
[3]Acadsoft Research, Gaziantep, Turkey.

sensitive locations from the trajectories, applying extensions of k-anonymity). Or, the data is transformed using a privacy-preserving manner (e.g., noise addition, distance-preserving data transformation). On the other hand, research in the second camp is concerned with secure computation and query evaluation on trajectory databases. For example, the protocol in [9] allows evaluation of range, distance and k-NN queries. [8] allows computing the distance between two encrypted trajectories.

Although these methods are undeniably useful for privacy protection, they are not necessarily robust against known-sample attacks. A known-sample attack is where the attacker (i.e., the adversary) has a set of objects that he knows will be in (or can be inserted into) the private database [10]. For example, the adversary can be the owner/driver of a car fleet whose trajectories are collected. Or, some trajectories in the database may correspond to his own movement patterns, or the patterns of his close friends and relatives. Or, the adversary can physically track a limited number of vehicles in a fleet and would like to infer about others.

As a motivating scenario, consider that the adversary has access to a "secure" trajectory database querying service, from which he can obtain statistical information (e.g., distance, nearest neighbors). Let X denote the victim's true trajectory that the adversary wishes to infer, and KT denote the adversary's set of known trajectories. (We often use the term *target trajectory* to refer to the victim's trajectory X.) For each trajectory T in KT, the adversary issues a query of the form: "Return the distance between X and the trajectory corresponding to T" or "Return the top-k most similar trajectories and their distances to T". A querying service, such as [8] or [11], is able to answer such queries. Using the answer, the adversary obtains the distance between X and T. The same process is repeated for as many T in KT as the querying service allows, and at the end the adversary has a set of known trajectories KT and the pairwise distances between trajectories in KT and his target trajectory X.

The question we ask in this paper is: Given a set of known trajectories KT and their pairwise distances to a target trajectory X, how much can an adversary learn about X? It turns out that a lot can be learned about X using a novel attack we devised, which yields the locations that the target trajectory visited with significant confidence. The adversary can also infer, with even higher confidence, the locations that the target trajectory has not visited. We therefore show that we can achieve privacy violations by disclosing the whereabouts of a victim using only pairwise distances between trajectories, which are seemingly harmless information and easily obtainable from existing mechanisms [8, 11]. If such mechanisms are blindly assumed safe, it becomes increasingly possible for an adversary to run a known-sample attack.

The sketch of the attack is as follows: Given $|KT|$ many known trajectories, we build a system of equations that yield $\lfloor |KT|/2 \rfloor$ locations when solved, that are possibly in the target trajectory. The remaining entries are interpolated using the previously found locations. We call the resulting trajectory a candidate trajectory. We repeat this process many times to obtain a set of candidate trajectories. We decide that there is a location disclosure based on the set

of candidate trajectories. That is, if most candidates indicate that the target trajectory visited a certain location p, then the attack declares that the target visited p. Similarly, if no candidates indicate that the target trajectory visited p, the attack can be used to declare that the target has not visited p. An interesting property of the attack is its noise resilience: Even if random noise was added to the adversary's known trajectories KT or the distances between KT and the target trajectory, in expectation the adversary would build the same set of equations when launching the attack. Thus, the attack also works with a known probability distribution of trajectories or distances (rather than exact trajectories or distances).

The rest of this paper is organized as follows: In Section 2, we give an overview of related work in location privacy, privacy-preserving computation over trajectory databases and trajectory publication, and known-sample attacks. In Section 3, we introduce our notation and describe the key concepts and give mathematical definitions. We present our attack in detail in Section 4. We provide an experimental analysis by quantifying the level/confidence of location disclosure in Section 5. Finally, in Section 6 we conclude the paper.

2. Related Work

Location privacy has been an important problem in various fields including vehicular networks, location-based services, location and proximity-based social networks (e.g., Foursquare, Tinder) and mobile crowdsourcing. Since it is implausible to review all of these areas in detail, we present only some of the major approaches and findings.

In [12], Gruteser and Grunwald introduce the notions of spatial and temporal *cloaking* for privacy-preserving access to location-based services. These rely on perturbing the resolution of data, through e.g., k-anonymization. In contrast, *mix-zones* break the continuity of location exposure by ensuring that users' movements cannot be traced while they are inside a mix-zone. Palanisamy and Liu [13] describe the state of the art approaches in building mix-zones over road networks. We refer the reader to [14] for a comparison between mix-zones and spatial cloaking. Gedik and Liu [15] offer privacy in mobile systems via the application of *location k-anonymity*. Andres et al. [16] introduce the notion of *geo-indistinguishability*, a generalization of differential privacy for location-based services. Alternative approaches to protect location privacy include path confusion [17], data obfuscation [18] and addition of dummies [19].

There have also been efforts to unify the aforementioned approaches. Shokri et al. [20] describe a framework that captures different types of users, privacy protection mechanisms and metrics. The authors also propose a new metric to measure location privacy, based on the expected distortion in reconstructing users' trajectories. In follow-up work [4], they use their framework to quantify location privacy under various types of adversarial information and attacks. They conclude that there is a lack of correlation between previously existing privacy metrics (e.g., k-anonymity) and the adversary's ability to infer users'

location. Wernke et al. [5] offer a survey on attacks and defenses in location privacy.

The main differences between the work in our paper and the location privacy literature are as follows: The threat in location privacy is often application and domain-dependent, and in real-time, i.e., there is a need to anonymize the location of a user while she is actually using a location-based service. Also, the knowledge of the adversary is a snapshot of users' locations or proximity to certain entities (e.g., a restaurant, another user) rather than a complete trajectory. On the other hand, our work assumes that complete trajectories were collected and stored in a central, private database. We initially assume that privacy protection mechanisms such as mix-zones or cloaking are not used. Although we then show the feasibility of the attack on partial and imperfect trajectory data, modifying the attack so that it defeats a particular location privacy mechanism is not the main purpose of this paper.

In **privacy-preserving trajectory data publishing**, the data owner has a database of trajectories and aims to publish this database while preserving individuals' privacy. In this paper, we do not assume that a trajectory database must be shared with the adversary to run the attack (and hence, most work in this area is orthogonal to ours), only a distance calculation interface to a private database is sufficient. However, trajectory publishing is relevant to our work in two aspects: an adversary who receives a copy of the published trajectories can (1) calculate distances between them, and (2) add the published database to his background knowledge, i.e., his set of known trajectories.

We first study anonymization-based techniques for trajectory publishing. In [6], Terrovitis and Mamoulis show that given partial trajectory information, it is possible to identify the full trajectory of an individual in a published database. They propose a suppression-based technique to combat this problem. A similar suppresion-based approach is later taken in [21], where the authors implement the $(K, C)_L$ privacy model for trajectory anonymization. Abul et al. [22] propose (k, δ)-*anonymity*, similar to k-anonymity but with an additional δ factor to account for location imprecision. Nergiz et al. [23] introduce generalizations in the area of trajectory anonymization, and study extensions of k-anonymity as their privacy model. Domingo-Ferrer et al. [24] present a novel distance metric for trajectories that is useful for clustering, and then use this metric for anonymization via microaggregation (i.e., replacing each cluster with synthetic trajectories).

With the widespread acceptance of differential privacy, the literature in trajectory publishing has also started shifting towards this privacy model. In [25], Chen et al. model trajectories as sequential data, and devise a method to publish such sequences in a differentially private manner. The main criticism of this work is that it only allows trajectories that consist of points from a small, fixed domain (e.g., only a few subway stops). Jiang et al. [26] try to address this shortcoming by privately sampling a suitable distance and direction at each position of a trajectory to infer the next possible position. More recently, Hua et al. [7] use differentially private generalizations and merging of trajectories to

publish trajectory data. In contrast, He et al. [27] build and publish synthetic datasets using differentially private statistics obtained from a private trajectory database.

Secure computation over trajectory databases enable users to perform various computations (e.g., statistical queries, k-NN queries, similarity search) on a trajectory database securely and accurately, while the data remains at its owner (i.e., never published). As argued earlier, the advent of these methods are sometimes a benefit rather than a burden for our attack.

In [28], Gkoulalas-Divanis and Verykios propose using a secure query engine that sits between a user and a trajectory database. This engine restricts users' queries, issues them on the database and then perturbs the results (e.g., by introducing fake trajectories) to fulfill certain privacy goals (e.g., disable tracking). The authors enhance their work in [9], supporting many types of queries useful for spatio-temporal data mining, e.g., range, distance and k-NN queries. Liu et al. [8] develop a method to securely compute the distance between two encrypted trajectories, which reveals nothing about the trajectories but the final result. Most similar to this work is the work of Zhu et al. [29], where authors describe a protocol to compute the distance between two time series in a client-server setting. In [11], Gowanlock and Casanova develop a framework that efficiently computes distance and similarity search queries on in-memory trajectory datasets.

Last, we survey **known sample attacks on private databases**. In *known sample* (or *known input*) attacks, the adversary is assumed to know a sample of objects in the private database, and tries to infer the remaining objects. This is the setting we consider in this paper. Liu et al. [30] develop a known sample attack that assumes the attacker has a collection of samples chosen from the same distribution as the private data. Chen et al. [31] develop an attack against privacy-preserving transformations involving data perturbation and additive noise. They assume a stronger adversary, one that knows input samples and the corresponding outputs after transformation. Turgay et al. [32] consider cases where the adversary knows input samples as well as distances between these samples and unknown, private objects. More recently, Giannella et al. [10] study and breach the privacy offered by Euclidean distance-preserving data transformations. Although the settings of these works are similar to ours, they are based on tabular or numeric datasets. On the other hand, the data model we assume in this paper is trajectories. Kaplan et al. [33] present a distance-based, known-sample attack on trajectories. While the main goal of [33] is rebuilding a private trajectory as accurately as possible, our work is concerned with location disclosure - that is via probabilistically identifying the locations that a private trajectory has and has not visited.

3. Preliminaries

3.1. Trajectories and Distances

We represent a trajectory T as a finite list of locations with timestamps, as follows: $T = ((p_1, t_1), (p_2, t_2), ..., (p_n, t_n))$. t_i corresponds to the timestamp, and a trajectory is sorted by its timestamps, i.e., $\forall i \in T$, $t_i < t_{i+1}$. Each p_i represents a 2-dimensional location with x and y coordinates, i.e., $p_i = (x_i, y_i)$. $|T|$ denotes the size of the trajectory, i.e., $|T| = n$. We define the following operations on locations: The scalar multiplication of a constant k with location p_i is defined as $k \cdot p_i = (k \times x_i, k \times y_i)$, where $\times$ is the arithmetic multiplication operator. We use the *norm* of a location to refer to the Euclidean vector norm, i.e., $||p_i|| = x_i^2 + y_i^2$. Also, for two locations p_i and p_j, $p_i \odot p_j = (x_i \odot x_j, y_i \odot y_j)$, where $\odot$ represents addition or subtraction. When there are multiple trajectories, we use superscripts to refer to the trajectory and subscripts for the locations within a trajectory, e.g., T^j is the j'th trajectory in the database and p_i^j is the i'th location in T^j.

We consider that mobile devices signal their location at desired timestamps $Q = (t_1, t_2, ..., t_n)$, and each signal is collected and stored as an ordered pair within the device's trajectory $(p_i, t_i) \in T$. This implies that the timestamps of all trajectories in the database are synchronous. We reckon, however, that this is a strong assumption in real-life uses of mobile devices, e.g., some samples might not be gathered due to signal loss etc. If such cases are rare, the data owner may decide to keep only those timestamps for which a location entry exists in all trajectories, and drop those timestamps where one or more trajectories imply a signal loss. Alternatively, to *fill in the missing entries* in a trajectory, one can use linear interpolation as follows.

Definition 1 (Linear Interpolation Function)**.** *Let $p_i = (x_i, y_i)$ and $p_j = (x_j, y_j)$ be two locations in a trajectory, sampled at time t_i and t_j respectively, where $t_i < t_j$. A location $p_k = (x_k, y_k)$ at timestamp t_k, where $t_i < t_k < t_j$ is interpolated using the interpolation function $\mathcal{I}((p_i, t_i), (p_j, t_j), t_k) = p_k$ where:*

$$x_k = x_i + (x_j - x_i) \cdot \frac{t_k - t_i}{t_j - t_i}, \qquad y_k = y_i + (y_j - y_i) \cdot \frac{t_k - t_i}{t_j - t_i}$$

Let T be an imperfect trajectory with missing entries. For each missing entry (i.e., t_k where a $(p_k, t_k) \notin T$), e.g., the signal at time t_k was lost, we interpolate p_k: Let t_i, $t_i < t_k$, be the largest timestamp such that $(p_i, t_i) \in T$; and t_j, $t_j > t_k$, be the smallest timestamp such that $(p_j, t_j) \in T$. Then, $p_k = (x_k, y_k)$ is computed using $\mathcal{I}$ as above and (p_k, t_k) is inserted into T. After this operation is performed for all missing t_k, T is sorted using timestamps, and we end up with the interpolated trajectory.

Definition 2 (Interpolation)**.** *Let T be a trajectory and Q be the list of desired timestamps. We say that the interpolated trajectory $T^* = ((p_1, t_1), ..., (p_n, t_n))$, is constructed via:*

- *For all t_i where $(p_i, t_i) \in T$ and $t_i \in Q$, $(p_i, t_i) \in T^*$.*

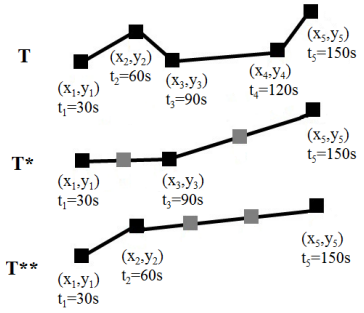


Figure 1: Linear interpolation of partial trajectories

- *For all t_i where $(p_i, t_i) \notin T$ but $t_i \in Q$, (p_i, t_i) is added to T^* using the linear interpolation process described above.*

Linear interpolation also becomes an integral part of the attack algorithm when rebuilding trajectories using partial information. Essentially, for a missing entry at time t_k, linear interpolation finds the closest timestamps to t_k, i.e., (p_i, t_i) and (p_j, t_j). It forms a line between p_i and p_j, and then places the missing location p_k on that line, using the timestamp t_k to find the distance of p_k from p_i and p_j.

We now illustrate interpolation using examples. In Fig. 1, let T be the actual trajectory of a vehicle and assume a constant location sampling rate of 30 seconds. In T^*, the samples at time 60s and 120s are lost. To reconstruct T^*, we interpolate independently to find (x_2, y_2) and (x_4, y_4). For the former, we draw a line between (x_1, y_1) and (x_3, y_3) and place (x_2, y_2) on that line, equidistant to (x_1, y_1) and (x_3, y_3) (due to constant sampling rate). Similar is done to interpolate (x_4, y_4), but this time using (x_3, y_3) and (x_5, y_5). In T^{**}, the samples at time 90s and 120s are lost. We reconstruct both with one interpolation involving (x_2, y_2) and (x_5, y_5).

As can be observed from these examples, interpolation is almost never perfect. This becomes a source of error later in the attack, which we try to quantify in Section 5. Also, the quality of interpolation depends on which sample is non-retrievable after the attack: If the non-retrievable sample actually sits on a perfect line with its neighbors, then its reconstruction will be accurate, hence minimal error. Otherwise, a larger error can be expected.

For the sake of simplicity, in the remainder of the paper we will assume that all trajectories in the database are perfectly known or already interpolated by the data owner. (This need not be linear interpolation, although it serves the purpose.) As such, we often treat a trajectory simply as a collection of locations: $T = (p_1, p_2, ..., p_n)$.

To compute distances between trajectories, we use Euclidean distance, the traditional method for distance measurement. Euclidean distance has been assumed heavily in the data privacy literature [10], and can be used as a basis

Trajectories	Distances
Trajectory 1: [(1,1),(2,2),(3,3)]	$d(T^1,T^2) = \sqrt{3}$
Trajectory 2: [(2,1),(3,2),(4,3)]	$d(T^1,T^3) = \sqrt{15}$
Trajectory 3: [(2,3),(3,4),(4,5)]	$d(T^2,T^3) = \sqrt{12}$

Table 1: Trajectories and distances

for building more complex distance measures for trajectories (e.g., DTW [34], LCSS [35]). The interested reader is referred to [36] for a thorough discussion.

Definition 3 (Euclidean Distance Between Trajectories)**.** *The Euclidean distance between two trajectories,* $T = (p_1, p_2, ..., p_n)$ *and* $T' = (p'_1, p'_2, ..., p'_n)$ *is calculated as:*

$$d(T, T') = \sqrt{\sum_{i=1}^{n} \|p_i - p'_i\|}$$

In Table 1, we provide three toy trajectories and calculate the distances between them.

3.2. Problem Setting

In this section, we formally describe the setting we consider in our attack. The data owner has a set of private trajectories $PT = \{T^1, ..., T^q\}$. The adversary has a set of known trajectories, $KT = \{T^1, ..., T^k\}$ and $KT \subset PT$. As we state in Section 1, an adversary may have a set of known trajectories due to various reasons, e.g., some trajectories correspond to a car that he or a close friend or relative drives, he can physically track some of the cars etc. The goal of the adversary is to infer the locations in a target trajectory, $T^r \in (PT - KT)$. Without loss of generality, we assume KT constitutes the first k trajectories in PT and the target trajectory is T^r for some $k < r \leq q$.

When an adversary attacks T^r and knows the trajectories in $KT = \{T^1, ..., T^k\}$, the pairwise distances between T^r and each trajectory in $\{T^1, ..., T^k\}$ are of interest. We denote these distances $\Delta = \{\delta_1, ..., \delta_k\}$, where $\delta_i = d(T^i, T^r)$, i.e., each distance is calculated via the Euclidean formula. At this point we do not consider how the Euclidean distance is actually retrieved. It can be done using, e.g., a published database or distance matrix, or following the secure distance calculation protocol for trajectories given in [8]. We discuss appropriate techniques in detail in Section 4.5.

Given KT and Δ, one can build many trajectories that satisfy Δ. In other words, there are many trajectories that have the desired distances Δ to a set of known trajectories KT.

Definition 4 (Distance Compliant Trajectory)**.** *Given* $KT = \{T^1, ..., T^k\}$ *and* $\Delta = \{\delta_1, ..., \delta_k\}$*, a trajectory* T *is distance compliant if and only if* $d(T^i, T) = \delta_i$ *for all* $i \in [1, k]$*.*

Without further information, any distance compliant trajectory can potentially correspond to the target T^r, which the adversary is trying to infer. The brute force attack method is to generate all distance compliant trajectories and make inferences based on them. While finding all such trajectories is infeasible, the adversary can use side-channel information to limit the domain of distance compliant trajectories and prune out some trajectories that cannot be T^r. We will discuss sources of side-channel information in Section 4.3. In the next sections, we will refer to those trajectories that satisfy side channel information and are distant compliant as *candidate trajectories*.

In our attack, the goal of the adversary is to infer, with high confidence, where a victim has been and has not been. We argue that even though the complete trajectory of the victim T^r cannot be reconstructed based on KT, Δ and side channel information, an adversary can still gain significant knowledge regarding the locations of the victim at some points of interest. For instance, a hospital could be one point of interest. If the adversary is very confident that T^r passes through the hospital, then he infers that the victim could have a health problem. We call this *positive disclosure*. On the other hand, if the adversary is confident that T^r does not pass through a certain area, then the adversary infers e.g., that the victim did not take part in a social event or rally. We call this *negative disclosure*. We make use of the following formalism to solidify these notions.

Definition 5 (Proximity Oracle)**.** *Given a location p, radius u and trajectory T, let $\mathcal{C}_{p,u}$ denote a circle with center p and radius u. We define the proximity oracle $\mathcal{O}$ as:*

$$\mathcal{O}_{p,u}(T) = \begin{cases} 1 & \text{if } T \cap \mathcal{C}_{p,u} \neq \emptyset \\ 0 & \text{otherwise} \end{cases}$$

Definition 6 (Location Disclosure Confidence)**.** *Given a set of candidate trajectories CT, a location p and radius u, the location disclosure confidence of the adversary is given by:*

$$conf_{p,u}(CT) = \frac{\sum_{T \in CT} \mathcal{O}_{p,u}(T)}{|CT|}$$

The proximity oracle builds a circular area centered at location p and with radius u. p is often the main point of interest, e.g., the hospital. If a trajectory passes through this area, the oracle returns 1. The attack employs the oracle as follows: The adversary will build, to the best of his abilities, a set of candidate trajectories CT for victim V. If $\text{conf}_{p,u}(CT)$ is greater than a certain threshold, i.e., the vast majority of trajectories in CT agree that V passes through the proximity of p, the adversary has achieved positive disclosure. If $\text{conf}_{p,u}(CT)$ is small (e.g., below a threshold such as 0.1 or 0.05), the adversary has achieved negative disclosure.

The success of the attack obviously depends on how accurate the trajectories in CT are, i.e., do they closely resemble the victim's trajectory? Therefore, the

attack needs to ensure that an accurate set of CT is built. This will be the main purpose of the attack algorithm, which we describe in the next section.

4. Attack Algorithm

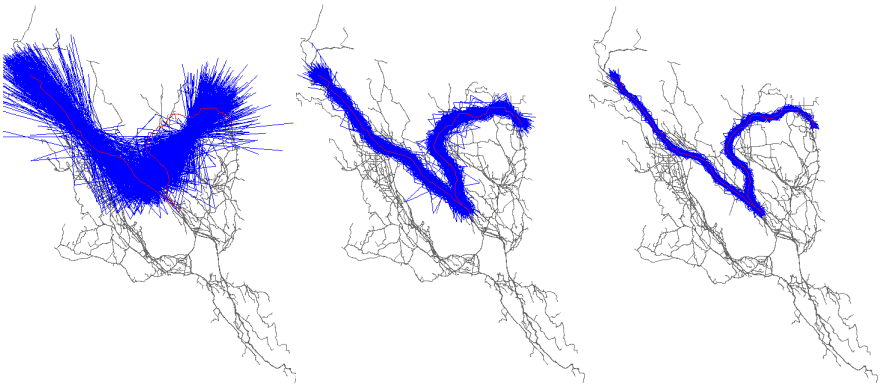
4.1. Overview of the Approach

In this section we present the main algorithm for our attack, which returns the location disclosure confidence of an area of interest. Although we built the proximity oracle using a circular area, the area of interest may in fact be of arbitrary shape. This has no bearing on the attack.

Given KT and Δ, the main idea of the attack is to build a set of candidate trajectories CT, such that any one of the trajectories in CT could be the victim's trajectory T^r. It is also possible that none of the trajectories in CT is actually T^r. If every possible candidate could be generated, only then we would be certain that T^r is one of the trajectories in CT. But this is computationally infeasible: Locations can be in high granularity, or the adversary's knowledge can be very limited (i.e., KT might not be sufficient to effectively limit the number of candidates). Thus, we try to generate only some of the candidate trajectories, and this acts as a sample of all possible candidates. Since candidates are generated randomly (i.e., we have a random sample) we argue that our sample captures the probabilistic properties of all possible candidates. This is an accurate assumption if the sample size is reasonably large.

As implied above and in Section 3.2, our attack relies on two assumptions: First, a random sample of candidate trajectories captures the properties of all possible candidate trajectories, with reasonable accuracy. Second, the set of candidate trajectories generated by our attack captures the properties of the target trajectory, with reasonable accuracy. We show that these assumptions hold with the help of Fig. 2. This figure illustrates a sample attack on the San Francisco dataset. (Details regarding the dataset will be given in Section 5.) The target trajectory is marked in red and the candidate trajectories are marked in blue. The figure illustrates that there is a clear association between the general behavior of the candidate trajectories and the target, and the candidates resemble the target. The quality of resemblance is associated with the number of trajectories known by the adversary, $|KT|$.

We observe that with $|KT| = 10$, the candidate trajectories give a rough idea on the whereabouts of the target. Also, the adversary can rule out more than half of the map as not visited by the target. However, with $|KT| = 10$, the candidates are crude rather than smooth: In Fig. 2a, they appear as collections of lines with sharp edges, and do not follow the smooth movement patterns observed in the target trajectory. With $|KT| = 30$ or 50, candidate trajectories become much more refined: They appear smoother, better resembling the curvatures and movement patterns of the target trajectory. Furthermore, they cover a smallar area and condense more on the target. By definition, this increases the location disclosure confidence $\text{conf}_{p,u}$ for locations p that are actually visited by the target, and decreases the disclosure confidence for locations that



(a) $|KT| = 10$ (b) $|KT| = 30$ (c) $|KT| = 50$

Figure 2: Attacking a trajectory in San Francisco

are not visited by the target. The prior increases true positives, and the latter decreases false positives, both of which are desired properties for our attack.

The pseudocode for our attack is given in Algorithm 1. Apart from KT and Δ, we take the proximity oracle $\mathcal{O}$ and an iteration parameter *itr* as inputs. *itr* is used to limit the number of candidate trajectories generated by the algorithm. When we think of the generated candidate trajectories as a random sample of all possible candidates, we observe that higher *itr* yields a larger sample, which is (in expectation) a better approximation of all candidates. Thus, we expect higher *itr* to decrease the variance and improve the precision of our attack. On the other hand, due to the non-trivial computation needed to generate more candidates, attack efficiency decreases.

The attack works as follows: Using KT and Δ, we are able to infer $t = \lfloor |KT|/2 \rfloor$ locations in a candidate trajectory T^c. However, T^c is often much larger, i.e., the adversary has 20 known trajectories but the victim's trajectory (and therefore T^c) contains 25 locations. Then, $20/2 = 10$ of the locations in T^c can be calculated using the *FindCandidate* function (described in the next section), but the remaining 15 are still unknown. These unknown locations are linearly interpolated by *FindCandidate*.

We now go through Algorithm 1 line by line. We initialize our set of candidate trajectories CT as empty, and in line 3, we find how many main locations will be calculated by *FindCandidate*, i.e., $t = \lfloor |KT|/2 \rfloor$. Within the *while* loop (lines 4-9) we randomly generate the set of indices S for interpolation. S contains the indices that determine which locations will be calculated and which ones will be interpolated by *FindCandidate* (details on this procedure will come in the next sections). The *FindCandidate* method is invoked in line 6, which returns a set of candidate trajectories. There can be cases where a valid candidate cannot be built with the given parameters, and in such cases the set returned by

ALGORITHM 1: Find location disclosure confidence

Input : KT: set of known trajectories
Δ: set of distances between the trajectories in KT and the target trajectory T^r
$\mathcal{O}_{p,u}$: the proximity oracle
itr: number of iterations

Output: $\text{conf}_{p,u}$: location disclosure confidence

$CT \leftarrow \{\}$
$j \leftarrow 0$
$t \leftarrow \left\lfloor \frac{|KT|}{2} \right\rfloor$
while $j < itr$ **do**
    Randomly create an ordered set of non-negative integer indices $S = (s_1, ..., s_{t-1})$ such that $\sum_{i=1}^{t-1} s_i = |T^r| - t$
    $R \leftarrow \text{FindCandidate}(KT, \Delta, S)$
    $CT \leftarrow CT \cup R$
    $j \leftarrow j + 1$
end
$cnt \leftarrow 0$
for $T \in CT$ **do**
    if $\mathcal{O}_{p,u}(T) = 1$ **then**
        $cnt \leftarrow cnt + 1$
    end
end
return $cnt/|CT|$

FindCandidate will be empty. If one or more candidates are returned, they are added to CT and we move on to the next iteration of the main *while* loop. After itr iterations of the loop, we start computing $\text{conf}_{p,u}$ according to Definition 6. The result is returned in line 16.

Example 1. We present an example to demonstrate the creation of indices. Let $|KT| = 6$ and the trajectory sizes be 5 (i.e., $|T^r| = 5$). In line 3, we get $t = \lfloor 6/2 \rfloor = 3$. Therefore in line 5, $S = (s_1, s_2)$, and $\sum_{i=1}^{2} s_i = 5 - 3 = 2$. Say that the random creation of indices yields $S = (1, 1)$. This will be passed over to Algorithm 2.

4.2. Creating a Generic Trajectory

In this section we start studying the *FindCandidate* algorithm given in Alg. 2. As can be seen from the algorithm, the first step is to build a generic trajectory T^g using S (the set of indices for interpolation). The current section will focus on this step, while the next section will present the second and third steps of *FindCandidate*.

As discussed earlier, *FindCandidate* is able to calculate $t = \lfloor |KT|/2 \rfloor$ locations and interpolates the rest. We refer to the locations that are actually calculated as the *main locations* and denote them by m_i, where $i \in [1...t]$. The rest are *interpolated locations*, denoted by $n_{i,j}$. $n_{i,j}$ sit between m_i and m_{i+1}, and the index s_i determines how many $n_{i,j}$ sit between m_i and m_{i+1}. If $s_i = 0$, then m_i and m_{i+1} are directly adjacent without any $n_{i,j}$ between them. If $s_i > 0$, then there is one interpolated location $n_{i,j}$ per $j \in [1...s_i]$. In the remainder of this and the next section, a *generic trajectory* refers to a collection of m_i and $n_{i,j}$.

For example, consider the generic trajectory in Fig. 3. Let $s_6 = 4$ and $s_9 = 0$. Let the main locations m_6, m_7, m_9 and m_{10} be placed as shown. Then, since $s_6 = 4$, we place the interpolated locations $n_{6,1}$, $n_{6,2}$, $n_{6,3}$ and $n_{6,4}$ between m_6 and m_7. Since $s_9 = 0$, there are no interpolated locations between m_9 and m_{10}. Notice that interpolated locations are uniformly distributed on the linear interpolant (i.e., line) between m_6 and m_7. That is, they are all equi-distant from one another, and also from the main points. This is because we do not know any information regarding the timestamps or speed, hence we assume uniform speed. At this point, the actual coordinates of all of the points are *unknowns*, and they need to be solved for in the next steps of *FindCandidate*. Once they are actually solved, the coordinates will be populated, and the resulting trajectory will become a candidate trajectory.

We write the interpolated locations $n_{i,k}$ in terms of the main locations. This allows us to reduce the number of unknown locations in a generic trajectory such that inferring only $\lfloor |KT|/2 \rfloor$ locations will be sufficient to solve for a candidate trajectory T^c later, using a generic trajectory. Mathematically, using the interpolation function $\mathcal{I}$ given in Definition 1 we write:

$$n_{i,k} = \mathcal{I}((m_i, 0), (m_{i+1}, s_i + 1), k)$$

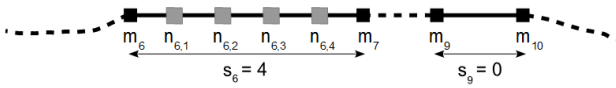


Figure 3: Building a generic trajectory T^g

The choice of timestamps 0, $s_i + 1$ and k comes from the uniform speed assumption explained above. Let $m_i = (x_i, y_i)$, $m_{i+1} = (x_{i+1}, y_{i+1})$ and $n_{i,k} = (x_{i,k}, y_{i,k})$. Applying Definition 1, we derive:

$$x_{i,k} = x_i + (x_{i+1} - x_i)\frac{k}{s_i + 1} \qquad y_{i,k} = y_i + (y_{i+1} - y_i)\frac{k}{s_i + 1}$$

Notice that $n_{i,k}$ is dependent only on the main points m_i, m_{i+1} (will be obtained by *FindCandidate*), s_i (generated by Algorithm 1) and k (its position on the linear interpolant).

Example 2. We continue from Example 1. Recall that trajectory size was 5, $S = (1, 1)$ and thus $s_1 = 1$ and $s_2 = 1$. Then in *FindCandidate*, T^g is: $(m_1, n_{1,1}, m_2, n_{2,1}, m_3)$.

4.3. Solving for a Candidate Trajectory

The pivotal part of the attack is to compute a candidate trajectory given KT, Δ, and a generic trajectory T^g consisting of main and interpolated points. This constitutes the second and third steps of Algorithm 2.

Let $T^g = (p_1^g, p_2^g, ..., p_n^g) = ((x_1^g, y_1^g), (x_2^g, y_2^g), ..., (x_n^g, y_n^g))$ be the generic trajectory. As T^g needs to be distance compliant, due to Definition 4 we have:

$$\sum_i \|p_i^g - p_i^j\| = (\delta_j)^2 \tag{1}$$

for all $T^j \in KT$, i.e., $j \in [1, |KT|]$. We rewrite the requirement above inductively:

$$\sum_i \|p_i^g - p_i^1\| = (\delta_1)^2 \tag{2}$$

$$\sum_i \|p_i^g - p_i^{j+1}\| - \|p_i^g - p_i^j\| = (\delta_{j+1})^2 - (\delta_j)^2 \tag{3}$$

where, in Equation 3, $j \in [1, |KT| - 1]$. The derivation of Equation 3 from Equation 1 is simple: We write Equation 1 for j and $j + 1$, and subtract the former from the latter, side by side. One can see that Equations 2 and 3 hold, using also an inductive argument: T^g should be distance compliant to T^1, and then should preserve the difference in distances between all consecutive j's, i.e., j to $j + 1$.

Lemma 1. *For all trajectories T^g, T^j, and i, $||p_i^g - p_i^j|| = (x_i^g - x_i^j)^2 + (y_i^g - y_i^j)^2$.*

Proof.

$$||p_i^g - p_i^j|| = (x_i^g - x_i^j)^2 + (y_i^g - y_i^j)^2$$
$$||(x_i^g, y_i^g) - (x_i^j, y_i^j)|| = (x_i^g - x_i^j)^2 + (y_i^g - y_i^j)^2$$
$$||(x_i^g - x_i^j, y_i^g - y_i^j)|| = (x_i^g - x_i^j)^2 + (y_i^g - y_i^j)^2$$
$$(x_i^g - x_i^j)^2 + (y_i^g - y_i^j)^2 = (x_i^g - x_i^j)^2 + (y_i^g - y_i^j)^2$$

The first step is due to the definition of locations: $p = (x, y)$. The second step follows from the properties of arithmetic operations defined at the beginning of Section 3.1. Finally, the last step is due to the definition of the Euclidean norm. □

Theorem 1. *For a fixed $j \in [1, |KT|-1]$, Equation 3 can be reduced to a linear equation of the form:*

$$\sum_i (\alpha_{i,j}) x_i^g + (\beta_{i,j}) y_i^g = \gamma_j \tag{4}$$

where α,β,γ are constants, and x_i^g, y_i^g, for all i, are unknowns.

Proof. We begin by applying Lemma 1 to trajectory pairs (T^g, T^j) and (T^g, T^{j+1}) to replace the two factors of the left hand side sum in Equation 3:

$$\sum_i (x_i^g - x_i^{j+1})^2 + (y_i^g - y_i^{j+1})^2 - (x_i^g - x_i^j)^2 - (y_i^g - y_i^j)^2 = (\delta_{j+1})^2 - (\delta_j)^2$$

Let $a^2 = (x_i^g - x_i^{j+1})^2$, $b^2 = (x_i^g - x_i^j)^2$, $c^2 = (y_i^g - y_i^{j+1})^2$ and $d^2 = (y_i^g - y_i^j)^2$. Since $a^2 - b^2 = (a-b)(a+b)$ and $c^2 - d^2 = (c-d)(c+d)$, we get:

$$\sum_i (x_i^g - x_i^{j+1} - x_i^g + x_i^j)(x_i^g - x_i^{j+1} + x_i^g - x_i^j)$$
$$+(y_i^g - y_i^{j+1} - y_i^g + y_i^j)(y_i^g - y_i^{j+1} + y_i^g - y_i^j) = (\delta_{j+1})^2 - (\delta_j)^2$$
$$\sum_i (x_i^j - x_i^{j+1})(2x_i^g - x_i^{j+1} - x_i^j) + (y_i^j - y_i^{j+1})(2y_i^g - y_i^{j+1} - y_i^j) = (\delta_{j+1})^2 - (\delta_j)^2$$
$$\sum_i 2x_i^j x_i^g - x_i^j x_i^{j+1} - x_i^j x_i^j - 2x_i^{j+1} x_i^g + x_i^{j+1} x_i^{j+1} + x_i^j x_i^{j+1}$$
$$+2y_i^j y_i^g - y_i^j y_i^{j+1} - y_i^j y_i^j - 2y_i^{j+1} y_i^g + y_i^{j+1} y_i^{j+1} + y_i^j y_i^{j+1} = (\delta_{j+1})^2 - (\delta_j)^2$$

Some of the terms in the sum cancel. We group the remaining terms and break the sum into several parts.

$$\sum_i (2x_i^j - 2x_i^{j+1}) x_i^g + (2y_i^j - 2y_i^{j+1}) y_i^g$$
$$+ \sum_i (x_i^{j+1})^2 + (y_i^{j+1})^2 - \sum_i (x_i^j)^2 + (y_i^j)^2 = (\delta_{j+1})^2 - (\delta_j)^2$$

Substituting $||p_i|| = (x_i)^2 + (y_i)^2$ and re-arranging the terms, we get:

$$\sum_i (2x_i^j - 2x_i^{j+1})x_i^g + (2y_i^j - 2y_i^{j+1})y_i^g = (\delta_{j+1})^2 - (\delta_j)^2 - \sum_i ||p_i^{j+1}|| + \sum_i ||p_i^j||$$

By replacing $\alpha_{i,j} = 2x_i^j - 2x_i^{j+1}$, $\beta_{i,j} = 2y_i^j - 2y_i^{j+1}$ and $\gamma_j = (\delta_{j+1})^2 - (\delta_j)^2 - \sum_i ||p_i^{j+1}|| + \sum_i ||p_i^j||$, we arrive at Equation 4, which concludes the proof.

Notice that α and β are functions of i, j and $j+1$. Furthermore, since the adversary has a set of known trajectories, x_i^j, x_i^{j+1}, y_i^j and y_i^{j+1} are all known to the adversary. Hence, the adversary can compute α and β. γ is a function of j and $j+1$, and can also be computed by the adversary using the known trajectories and distances Δ. Consequently, the only unknowns in Equation 4 are the coordinates of the generic trajectory, i.e., x_i^g and y_i^g. □

We would like to underline that Equation 4 is linear, whereas Equation 1 and Equation 3 are quadratic. Solving a system (i.e., collection) of linear equations is achievable and well-studied. In contrast, solving a system of quadratic equations is difficult. The reduction from quadratic equations to linear equations makes the attack feasible.

Theorem 1 builds a linear equation for one j among the set of known trajectories. Since Equation 3 holds for $j \in [1, |KT| - 1]$, a linear equation can be built for all $j \in [1, |KT| - 1]$. We therefore obtain a system of $|KT| - 1$ linear equations. Plus, we have one quadratic equation due to Equation 2. We can solve this system for $|KT|$ unknowns. Had some of the locations in the generic trajectory not been interpolated, we would have had $2 \times |T^g|$ unknowns (both x and y coordinates are unknowns per location, hence twice the number of locations in T^g), and oftentimes $2 \times |T^g| > |KT|$. (If not, T^g can be completely retrieved.) This is why interpolated locations were necessary: By acknowledging that we can solve for $\lfloor |KT|/2 \rfloor$ number of unknown locations, we reduced the number of unknowns in T^g in advance and settled for approximating the rest of T^g.

We now discuss the *FindCandidate* method in Algorithm 2 in detail. The algorithm works in three steps. In the first step, we build a generic trajectory T^g using the input S (indices for interpolation). This process was explained and exemplified in the previous section. In the second step, we obtain the linear equations using Theorem 1. Then, we obtain one quadratic equation using Equation 2. In the third step, we solve this system of equations. There are various ways to solve a set of equations, e.g., writing it as a matrix and column vector multiplication, variable elimination, Gaussian elimination and row reduction. Any one of these can be used to solve the linear equations, and the solution is then fed into the quadratic equation.

The quadratic equation often yields two roots (since it is quadratic), which implies that there are two solution trajectories (denoted T^{sol} in Algorithm 2). We check whether a solution trajectory is valid and satisfies our side channel information before returning it. First, the validity of T^{sol} requires that the roots of the quadratic equation are real (i.e., not imaginary numbers). Second, we use

ALGORITHM 2: Find candidate trajectory

Input : KT: set of known trajectories
Δ: set of distances between the trajectories in KT and the target trajectory T^r
S: indices for interpolation
Output: R: a set of candidate trajectories

`/* Step 1: Build a generic trajectory` T^g `using` S `*/`
$T^g \leftarrow ()$
for $i = 1$ *to* $|S|$ **do**
  $T^g \leftarrow T^g + m_i$
  for $k = 1$ *to* s_i **do**
    Let interpolated location $n_{i,k} = \mathcal{I}((m_i, 0), (m_{i+1}, s_i + 1), k)$
    $T^g \leftarrow T^g + n_{i,k}$
  end
end
$T^g \leftarrow T^g + m_t$
`/* Step 2: Build the set of equations` $EQNS$ `*/`
$EQNS \leftarrow \{\}$
for $j = 1$ *to* $2 \times \left\lfloor \frac{|KT|}{2} \right\rfloor - 1$ **do**
  Let Q be the linear equation $\sum_i (\alpha_{i,j}) x_i^g + (\beta_{i,j}) y_i^g = \gamma_j$, where
  $\alpha_{i,j} = 2x_i^j - 2x_i^{j+1}$, $\beta_{i,j} = 2y_i^j - 2y_i^{j+1}$ and
  $\gamma_j = (\delta_{j+1})^2 - (\delta_j)^2 - \sum_i ||p_i^{j+1}|| + \sum_i ||p_i^j||$
  `// using Theorem 1`
  $EQNS \leftarrow EQNS \cup Q$
end
`/* Let the first trajectory in` KT `be denoted` $((x_1^1, y_1^1), ..., (x_n^1, y_n^1))$ `*/`
Let Q be the quadratic equation $\sum_i (x_i^g - x_i^1)^2 + (y_i^g - y_i^1)^2 = (\delta_1)^2$
`// using Equation 2 and Lemma 1`
$EQNS \leftarrow EQNS \cup Q$
`/* Step 3: Solve` $EQNS$ `and obtain candidate trajectories */`
$R \leftarrow \{\}$
for *each solution* T^{sol} *to* $EQNS$ **do**
  if T^{sol} *satisfies side channel information* **then**
    $R \leftarrow R \cup T^{sol}$
    `// see text for types of side channel information`
  end
end
return R

the side channel information explained next to check if T^{sol} is indeed physically feasible. If T^{sol} is either invalid or infeasible, it is discarded by Algorithm 2 in the last step (lines 18-21).

Side channel information. In our work, we use two sources of side channel information:

1. Geographic assumptions: T^{sol} should fall within the boundaries of a certain map. For example, in our experiments we use vehicles' GPS trajectories travelling in particular cities. Then, we check to confirm that each location in T^{sol} falls within the geographic boundaries of the respective city.
2. Mobility characteristics: Consecutive locations in a trajectory are not independent, and physical limitations on moving objects' speed constrain T^{sol}. For example, it is impossible to drive faster than 450 km/h with current cars in production. Thus, we check each consecutive location in T^{sol} to see if there is a violation of a realistic speed limit. In case there is a violation, T^{sol} is marked as infeasible.

It is possible to extend this side channel information, but many extensions are non-trivial and open to debate. An extended geographic assumption could be to say that vehicles' locations cannot fall within a sea, river, etc. However, counter-arguments can be made using bridges or car ferries that would allow vehicles to cross the water, or general GPS positioning errors in open areas. Note that the likes of seashores are prime candidates for open areas with low GPS accuracy, which makes it challenging to rule out the possibility of travelling on a bridge/shore. An extended mobility characteristic could be to take into account the individual speed limits and traffic level on the roads. Similarly, counter-arguments can be made by saying that the vehicle in question is a police car or ambulance which is not subject to speed limits and can bypass traffic. To avoid such complications and debate, we choose not to include these non-trivial instances of side channel information in our work.

Example 3. We present an example for building and solving the system of equations using Algorithm 2. Let the input parameters to Algorithm 2 be as follows: For the sake of simplicity, let trajectories contain a single location, and thus S be an empty set. Let the known trajectories be $KT = \{((2,4)),((0.5,1.5))\}$, and distances be $\Delta = \{\sqrt{40}, \sqrt{40.5}\}$.

In the first step (lines 1-9 of Algorithm 2) the following generic trajectory is built: $T^g = ((x_1^g, y_1^g))$. In the following *for* loop (lines 10-14) we construct one linear equation: $\alpha_{1,1}x_1^g + \beta_{1,1}y_1^g = \gamma_1$. We compute $\alpha_{1,1} = 2x_1^1 - 2x_1^2 = 4-1 = 3$, $\beta_{1,1} = 2y_1^1 - 2y_1^2 = 8-3 = 5$ and $\gamma_1 = (\delta_2)^2 - (\delta_1)^2 - ||p_1^2|| + ||p_1^1|| = 40.5 - 40 - 2.5 + 20 = 18$. Hence the linear equation we get is: $3x_1^g + 5y_1^g = 18$. This is added to our system of equations, $EQNS$. In lines 15-16, we add the following quadratic equation to $EQNS$: $(x_1^g - 2)^2 + (y_1^g - 4)^2 = 40$.

To solve the two equations, we can first write y_1^g in terms of x_1^g using the linear equation. That is, $y_1^g = (18 - 3x_1^g)/5$. Then, we replace y_1^g with this term

in the quadratic equation. Thus we obtain:

$$(x_1^g - 2)^2 + (\frac{18 - 3x_1^g}{5} - 4)^2 = 40$$

Solving the above for x_1^g, we find the two roots $x_1^g = -4$ or $x_1^g = 6.59$. We can retrieve $y_1^g = 6$ or $y_1^g = -0.35$ for the two roots respectively, hence we have two T^{sol}: $T^{sol1} = ((-4, 6))$ and $T^{sol2} = ((6.59, -0.35))$. We check if they satisfy side-channel information and return those that do. For example, the adversary may know that due to the borders of his map, the victim's trajectory cannot have a negative y coordinate. In this case, T^{sol2} would be eliminated in line 19 in Algorithm 2.

We had only one linear equation above, but in general we may have more than just one. However, the number of unknowns in those equations will always be one more than the number of equations (e.g., $|KT|$ unknowns but $|KT| - 1$ equations). A reliable way to solve the equations is to designate one variable as the *free variable* and the rest of the variables depend on the free variable. In the example above, x_1^g was the free variable and y_1^g depended on x_1^g. The designation of the free variable and re-writing all variables can be done in a variety of ways including row reduction and variable elimination.

4.4. Noise Resilience

There is a vast amount of work on location privacy that relies on adding noise to location data in order to protect individuals' privacy. Our goal in this section is to prove that the attack is resilient against such methods. This shows that even though the adversary's background knowledge is noisy (i.e., imperfect) the attack can be carried out with reasonable accuracy. In particular, we consider two cases: (1) trajectories are noisy, and (2) distances between trajectories are noisy.

We assume Gaussian noise with mean 0 and variance σ^2, i.e., $\mathcal{N}(0, \sigma^2)$. This has two primary reasons. First, Gaussian is by far the most commonly used method in additive data perturbation. (See [37] and [38]). Second, Gaussian noise is also used in the scope of differential privacy (in particular, (ε, δ)-DP), which is currently the most active area in statistical database privacy. Given a function f with numeric output, answering f by adding Gaussian noise with variance $\sigma^2 \geq (\Delta f)^2 \times 2ln(1.25/\delta)/\varepsilon^2$ (where Δf is the L_2-sensitivity of f) to the true output of f satisfies (ε, δ)-DP [39]. Although we assume Gaussian noise, a number of derivations below apply to other noise distributions with mean 0. (For example, differential privacy also employs Laplace noise with mean 0 to achieve privacy.) We note such instances where applicable.

4.4.1. Noisy Trajectories

We let random noise to be added to each location in a trajectory, such that the adversary's background knowledge of KT becomes imperfect. Such a setting is plausible in real life (perhaps even more plausible than having perfect knowledge of all trajectories in KT). For example, the adversary's background

might consist of trajectories that were published in an external database, but this publication was noisy to achieve privacy protection. (See the related work on privacy-preserving trajectory data publishing.) Alternatively, some location privacy technique (e.g., cloaking) might have been used to disable the adversary from observing actual locations, but instead the adversary observes similar, perturbed locations.

Let $T^j = (p_1^j, p_2^j, ..., p_n^j) = ((x_1^j, y_1^j), (x_2^j, y_2^j), ..., (x_n^j, y_n^j))$ be a trajectory and $\hat{T}^j = (\hat{p}_1^j, \hat{p}_2^j, ..., \hat{p}_n^j) = ((\hat{x}_1^j, \hat{y}_1^j), (\hat{x}_2^j, \hat{y}_2^j), ..., (\hat{x}_n^j, \hat{y}_n^j))$ be its noisy version. We say that for all $i \in [1, n]$ and for all j, $\hat{x}_i^j = x_i^j + \mathcal{X}_{i,j}$ and $\hat{y}_i^j = y_i^j + \mathcal{Y}_{i,j}$ where $\mathcal{X}_{i,j}$ and $\mathcal{Y}_{i,j}$ are independent random variables: $\mathcal{X}_{i,j} \sim \mathcal{N}(0, \sigma^2)$ and $\mathcal{Y}_{i,j} \sim \mathcal{N}(0, \sigma^2)$.

Recall that we employ several linear equations and one quadratic equation when solving for a candidate trajectory. We first study the effect of noise on the linear equations. That is, we answer the following question: How are the parameters in linear equations built using Theorem 1 affected by noise? The three parameters in question are $\alpha_{i,j}$, $\beta_{i,j}$ and γ_j.

Theorem 2. *Let $\hat{\alpha}_{i,j}$ denote the $\alpha_{i,j}$ parameter in the noisy world. Then, $\hat{\alpha}_{i,j}$ is an unbiased estimator of $\alpha_{i,j}$.*

Proof. We prove this by computing the expected value of $\hat{\alpha}_{i,j}$.

$$\begin{aligned} E[\hat{\alpha}_{i,j}] &= E[2\hat{x}_i^j - 2\hat{x}_i^{j+1}] \\ &= 2E[\hat{x}_i^j] - 2E[\hat{x}_i^{j+1}] \\ &= 2E[x_i^j + \mathcal{X}_{i,j}] - 2E[x_i^{j+1} + \mathcal{X}_{i,j+1}] \\ &= 2E[x_i^j] + 2E[\mathcal{X}_{i,j}] - 2E[x_i^{j+1}] - 2E[\mathcal{X}_{i,j+1}] \\ &= 2x_i^j - 2x_i^{j+1} = \alpha_{i,j} \end{aligned}$$

The final step follows from the fact that x_i^j, x_i^{j+1} are constants, and since $\mathcal{X} \sim \mathcal{N}(0, \sigma^2)$ it has an expected value of 0. The above holds for any noise distribution with mean 0 and finite variance (not just for Gaussian). □

It is trivial to see that $\hat{\beta}_{i,j}$ has the same guarantees - just swap x coordinates with y coordinates and the proof stays the same. That is, $\hat{\beta}_{i,j}$ is an unbiased estimator of $\beta_{i,j}$.

Theorem 3. *Let $\hat{\gamma}_j$ denote the γ_j parameter in the noisy world. Then, $\hat{\gamma}_j$ is an unbiased estimator of γ_j.*

Proof. We first expand $\hat{\gamma}_j$ and write it in open form.

$$\begin{aligned}
\hat{\gamma}_j &= (\delta_{j+1})^2 - (\delta_j)^2 + \sum_i ||\hat{p}_i^j|| - ||\hat{p}_i^{j+1}|| \\
&= (\delta_{j+1})^2 - (\delta_j)^2 + \sum_i (x_i^j + \mathcal{X}_{i,j})^2 + (y_i^j + \mathcal{Y}_{i,j})^2 \\
&\qquad - (x_i^{j+1} + \mathcal{X}_{i,j+1})^2 - (y_i^{j+1} + \mathcal{Y}_{i,j+1})^2 \\
&= (\delta_{j+1})^2 - (\delta_j)^2 + \sum_i (x_i^j)^2 + 2x_i^j \mathcal{X}_{i,j} + (\mathcal{X}_{i,j})^2 + (y_i^j)^2 + 2y_i^j \mathcal{Y}_{i,j} + (\mathcal{Y}_{i,j})^2 \\
&\qquad - (x_i^{j+1})^2 - 2x_i^{j+1}\mathcal{X}_{i,j+1} - (\mathcal{X}_{i,j+1})^2 \\
&\qquad - (y_i^{j+1})^2 - 2y_i^{j+1}\mathcal{Y}_{i,j+1} - (\mathcal{Y}_{i,j+1})^2
\end{aligned}$$

We now find $E[\hat{\gamma}_j]$. Note that δ_j, δ_{j+1}, x_i^j, y_i^j, x_i^{j+1} and y_i^{j+1} are all constants, and their expected values are equal to themselves. Therefore we have:

$$\begin{aligned}
E[\hat{\gamma}_j] = (\delta_{j+1})^2 - (\delta_j)^2 + \sum_i (x_i^j)^2 + 2x_i^j E[\mathcal{X}_{i,j}] + (y_i^j)^2 + 2y_i^j E[\mathcal{Y}_{i,j}] - (x_i^{j+1})^2 \\
- 2x_i^{j+1} E[\mathcal{X}_{i,j+1}] - (y_i^{j+1})^2 - 2y_i^{j+1} E[\mathcal{Y}_{i,j+1}] \\
+ E[(\mathcal{X}_{i,j})^2] + E[(\mathcal{Y}_{i,j})^2] - E[(\mathcal{X}_{i,j+1})^2] - E[(\mathcal{Y}_{i,j+1})^2]
\end{aligned}$$

We know that $E[\mathcal{X}_{i,j}] = E[\mathcal{Y}_{i,j}] = E[\mathcal{X}_{i,j+1}] = E[\mathcal{Y}_{i,j+1}] = 0$ due to the properties of the Gaussian distribution. Therefore some terms cancel. Also, since all $\mathcal{X}$ and $\mathcal{Y}$ are independent and identically distributed, $E[(\mathcal{X}_{i,j})^2] + E[(\mathcal{Y}_{i,j})^2]$ will cancel with $-E[(\mathcal{X}_{i,j+1})^2] - E[(\mathcal{Y}_{i,j+1})^2]$. (An alternate method for this step of the proof is to model the square of the Gaussian distribution as a Gamma distribution, and then compute the expected values of the Gamma distribution. We stick with the aforementioned argument for brevity and clarity.) Thus:

$$\begin{aligned}
E[\hat{\gamma}_j] &= (\delta_{j+1})^2 - (\delta_j)^2 + \sum_i (x_i^j)^2 + (y_i^j)^2 - (x_i^{j+1})^2 - (y_i^{j+1})^2 \\
&= (\delta_{j+1})^2 - (\delta_j)^2 - \sum_i ||p_i^{j+1}|| + \sum_i ||p_i^j|| \\
&= \gamma_j
\end{aligned}$$

□

Theorems 2 and 3 together show that, in expectation, all parameters of the linear equations should stay the same despite the added noise to trajectories. That is, we expect to build the same system of linear equations regardless of whether trajectories are noisy or not.

We now study the effect of noise on the quadratic equation. Recall the quadratic equation in line 6 of Algorithm 2. The right hand side is not affected by the addition of noise to trajectories, but the left hand side (LHS) is. Let $\hat{LHS}$ denote its noisy version.

Theorem 4. *$L\hat{H}S$ is a biased estimator of LHS, with a fixed bias of $2n\sigma^2$.*

Proof.

$$\begin{aligned}
E[L\hat{H}S] &= E[\sum_i (x_i^c - \hat{x}_i^1)^2 + (y_i^c - \hat{y}_i^1)^2] \\
&= E[\sum_i (x_i^c - x_i^1 - \mathcal{X}_{i,1})^2 + (y_i^c - y_i^1 - \mathcal{Y}_{i,1})^2] \\
&= \sum_i E[(x_i^c - x_i^1)^2] - 2(x_i^c - x_i^1)E[\mathcal{X}_{i,1}] + E[(\mathcal{X}_{i,1})^2] + E[(y_i^c - y_i^1)^2] \\
&\qquad - 2(y_i^c - y_i^1)E[\mathcal{Y}_{i,1}] + E[(\mathcal{Y}_{i,1})^2]
\end{aligned}$$

As in the previous proofs, $E[\mathcal{X}_{i,1}] = E[\mathcal{Y}_{i,1}] = 0$. For $E[(\mathcal{X}_{i,1})^2]$ and $E[(\mathcal{Y}_{i,1})^2]$, we make the following observation: The square of a Gaussian variable $\sim \mathcal{N}(0, \sigma^2)$ yields a scaled chi-square, which in turn yields a random variable Q with Gamma distribution $Q \sim \Gamma(1/2, 2\sigma^2)$. The expected value of this is: $E[Q] = \sigma^2$. Plugging this into the last equation above, we get:

$$\begin{aligned}
E[L\hat{H}S] &= \sum_i ((x_i^c - x_i^1)^2 + \sigma^2 + (y_i^c - y_i^1)^2 + \sigma^2) \\
&= LHS + 2n\sigma^2
\end{aligned}$$

□

This result is significant in the following sense: An adversary knows how many entries there are in a trajectory (hence, n). If the adversary also knows the variance of the noise, he can remove the fixed bias from $L\hat{H}S$ when building the quadratic equation (i.e., subtract $2n\sigma^2$ from both sides of the equation).

4.4.2. Noisy Distances

We let random noise to be added to each distance δ between the candidate trajectory and the known trajectories. That is, the adversary's knowledge of Δ becomes imperfect. This may arise in real life because the protocol for computing trajectory distance can be (deliberately or non-deliberately) made noisy. Or, for each $\delta \in \Delta$, instead of exact δ, the adversary may have a probability distribution for δ. (This makes the attack also a known probability-distribution attack instead of a known distance attack.)

Let $\Delta = \{\delta_1, \delta_2, ..., \delta_k\}$ be the set of actual distances. Instead of Δ, we say that the adversary observes $\hat{\Delta} = \{\hat{\delta}_1, \hat{\delta}_2, ..., \hat{\delta}_k\}$, where for all $i \in [1, k]$, $\hat{\delta}_i = \delta_i + \mathcal{X}_i$ and $\mathcal{X}_i \sim \mathcal{N}(0, \sigma^2)$ is an independent random variable. (Equivalently, the adversary has a known probability distribution of distances: $\hat{\Delta} = \{\mathcal{Y}_1, \mathcal{Y}_2, ..., \mathcal{Y}_k\}$, where $\mathcal{Y}_i \sim \mathcal{N}(\delta_i, \sigma^2)$.)

Again, we first focus on the linear equations. Since parameters $\alpha_{i,j}$ and $\beta_{i,j}$ do not depend on δ, they remain unaffected from the noise added to δ. In addition, even though γ_j is affected, its noisy version $\hat{\gamma}_j$ turns out to be an unbiased estimator of γ_j.

Theorem 5. *$\hat{\gamma}_j$ is an unbiased estimator of γ_j.*

Proof.

$$\begin{aligned}
E[\hat{\gamma}_j] &= E[(\delta_{j+1} + \mathcal{X}_{j+1})^2 - (\delta_j + \mathcal{X}_j)^2 - \sum_i ||p_i^{j+1}|| + \sum_i ||p_i^j||] \\
&= E[(\delta_{j+1})^2] + E[2\delta_{j+1}\mathcal{X}_{j+1}] + E[(\mathcal{X}_{j+1})^2] - E[(\delta_j)^2] - E[2\delta_j\mathcal{X}_j] - E[(\mathcal{X}_j)^2] \\
&\quad - E[\sum_i ||p_i^{j+1}||] + E[\sum_i ||p_i^j||]] \\
&= (\delta_{j+1})^2 + 2\delta_{j+1}E[\mathcal{X}_{j+1}] + E[(\mathcal{X}_{j+1})^2] - (\delta_j)^2 - 2\delta_j E[\mathcal{X}_j] - E[(\mathcal{X}_j)^2] \\
&\quad - \sum_i ||p_i^{j+1}|| + \sum_i ||p_i^j||
\end{aligned}$$

$E[\mathcal{X}_{j+1}] = E[\mathcal{X}_j] = 0$ and $E[(\mathcal{X}_{j+1})^2]$ cancels with $E[(\mathcal{X}_j)^2]$ since they are independent and identically distributed. Hence:

$$\begin{aligned}
E[\hat{\gamma}_j] &= (\delta_{j+1})^2 - (\delta_j)^2 - \sum_i ||p_i^{j+1}|| + \sum_i ||p_i^j|| \\
&= \gamma_j
\end{aligned}$$

□

Next, we study the quadratic equation. Unlike the previous subsection, the LHS does not change due to noise, but the $RHS = (\delta_1)^2$ does. Let $\hat{RHS}$ denote its noisy version. We show below that the bias of $\hat{RHS}$ is fixed.

Theorem 6. *$\hat{RHS}$ is a biased estimator of RHS, with a fixed bias of σ^2.*

Proof.

$$\begin{aligned}
E[\hat{RHS}] &= E[(\delta_1 + \mathcal{X}_1)^2] \\
&= E[(\delta_1)^2] + E[2\delta_1\mathcal{X}_1] + E[(\mathcal{X}_1)^2] \\
&= \delta_1^2 + 2\delta_1 E[\mathcal{X}_1] + E[(\mathcal{X}_1)^2]
\end{aligned}$$

$E[\mathcal{X}_1] = 0$, and $(\mathcal{X}_1)^2 \sim \Gamma(1/2, 2\sigma^2)$, for which the expected value is σ^2. (See the proof of Theorem 4.)

$$\begin{aligned}
E[\hat{RHS}] &= \delta_1^2 + \sigma^2 \\
&= RHS + \sigma^2
\end{aligned}$$

□

Similar to Section 4.4.1, the system of equations we build using noisy distances (or probability distributions of distances) behaves as if there were no noise in the adversary's background knowledge. This shows that the attack is resilient to noise.

4.4.3. Application to Differential Privacy

Due to the current popularity of differential privacy (DP) in the research community, we discuss how our attack can apply to databases protected by DP. The use of DP on trajectory data can be studied under 3 separate categories: (1) for interactive database protection, (2) for privacy-preserving data publishing, and (3) for location privacy.

In (1), a protective DP layer is placed between a data analyst (the party who is posing queries) and the database, which often acts as a middleware. The likes of PINQ, GUPT and Explode exemplify this architecture [40, 41, 42]. In this case, the analyst's access is limited to the types of queries supported by the middleware. In most cases, the middleware only supports retrieval of aggregate statistics (e.g., counts, sums, histograms and linear algebraic functions) and applications that rely on such statistics. Furthermore, the definition of DP assumes *neighboring datasets* that differ in one record (e.g., one trajectory). Since it is possible that T^r or the trajectories in KT do not exist in a neighboring dataset, the Euclidean distance query $d(T, T')$, where $T = T^r$ or $T \in KT$, has unbounded sensitivity and therefore must be answered with unbounded noise, which makes the answer meaningless. As a result, it is not feasible to retrieve Δ from a database that is protected by interactive DP, making our attack impossible. Note that this protection comes at the expense of no longer being able to answer important types of spatio-temporal queries such as similarity, distance and k-NN queries.

In (2), the goal is to perform a one-time publishing of trajectory data while respecting DP. The prominent approach to achieve this goal is to privately obtain relevant statistics and properties from the data, generate synthetic trajectories that satisfy these statistics, and publish the synthetic trajectories. [25] and [27] take this approach. In this case, the success of our attack relies on how well the synthetic trajectories resemble their actual counterparts. Previous works show that resemblance is high, and high resemblance is desired from a utility perspective. Thus, even though there may not be an explicit one-to-one correspondence between an actual trajectory and its synthetic counterpart, a link may be established using the uniqueness and predictability of human mobility [43, 44], and our attack becomes feasible.

In (3), DP is used for location privacy, in which actual locations in trajectories are replaced by perturbed or pseudo-locations. As shown in [16, 26, 45], there are several mechanisms to use for location perturbation, such as extensions of the Laplace and Gaussian mechanisms (both of which add noise with mean 0, as assumed in Section 4.4.1) as well as the Exponential mechanism. Therefore, as long as the attack is resilient to the added noise in trajectory locations, it should produce accurate results despite DP location privacy.

Per the discussion above, we conclude that our attack is most feasible in case of (3), and least feasible in case of (1). Note that for (2) and (3), an important parameter is the privacy parameter of DP, ε. When ε is low, privacy protection is more strict, and therefore noise is higher and the outcome is more distorted. When ε is high, privacy protection is more relaxed, and therefore less noise

is needed. In the case of (2), high ε translates to synthetic trajectories very closely resembling actual trajectories, making our attack more feasible. In the case of (3), high ε translates to small distortion in each individual location in a trajectory, which means the observed locations are very similar to the actual ones, again making the attack more feasible.

4.5. Finding Indices of Known Trajectories

Our attack algorithm assumes that the attacker has Δ, which implicitly means that the attacker could compute the distance between any known trajectory and a target trajectory prior to the attack. Earlier in Section 3.2, we had noted that this could be done using a published distance matrix, a distance-preserving transformation, or a secure trajectory distance calculation protocol. However, we had deferred the discussion of a concrete method. Here, we outline a method that is applicable to the first two settings, which have also been previously assumed in the relevant literature [10, 30, 31, 32].

First, note that if the attacker is able to identify the records belonging to the known trajectories in the private dataset, computing Δ becomes trivial. Otherwise, due to distance-preserving property, the attacker can compute the dissimilarity matrix D for the private trajectory database $\{T^1, \ldots, T^m\}$ where $D_{i,j} = d(T^i, T^j)$. Attacker's goal is to find the row indices corresponding to known trajectories in $KT = \{T'^1, \ldots, T'^k\}$. This particular problem has been previously addressed in [10] and we now follow a similar technique.

Let $SD_{u,v}$ denote the set of pairs (i, j) such that $D_{i,j} = d(T'^u, T'^v)$. Our aim is to find all one-to-one assignment functions $\psi : \{1, \ldots, k\} \to \{1, \ldots, m\}$ such that $(\psi(u), \psi(v)) \in SD_{u,v}$ for all $u, v \in [1, k]$. All such assignments can be found by searching through the cross product of sets $SD_{u,v} \mid u, v \in [1, k]$ in $\prod_{u,v\in[1,k]} |SD_{u,v}|$ iterations.

In case of noisy data $SD_{u,v}$ can be generated within some confidence. If the parameters of the noise σ are known beforehand, given some confidence threshold th, an attacker can set $SD_{u,v} = \{(i, j) : P(|D_{i,j} - d(T'^u, T'^v)| < \sigma) > th\}$. Otherwise, given a parameter ρ, $SD_{u,v}$ can be constructed to contain the top ρ index pairs with corresponding distances closest to $d(T'^u, T'^v)$.

If there happens to be only one valid assignment function, the attacker can confidently resume the rest of the attack. Unfortunately, there may be more than one valid assignment. If this is the case, we list two strategies the attacker can follow depending on the available background knowledge:

1. An attacker with no additional background information can proceed as in [10]. The attacker tries to find a subset $\hat{KT}$ of KT (of largest possible size) for which we have one unique assignment. Attack is then carried out with $\hat{KT}$ instead of KT. By doing so, we lose some accuracy due to $|\hat{KT}| < |KT|$.
2. If the attacker has some information on the distribution of the private trajectories (e.g., underlying map, home addresses, ...) and we have a few number of valid assignments, the attacker can proceed as follows: For all valid assignments, we reconstruct the trajectory database using the

attack algorithm given in Section 4.1. The attacker outputs the database that better respects the expected distribution. Databases constructed due to incorrect assignments will most likely deviate from the expected distribution.

While having to deal with multiple valid assignments is possible, we note that this is seldom the case, especially for complex data. In practice, almost all SD sets regarding real trajectories contain either a single index pair or very few number of index pairs (i.e., $d(T^i, T^j)$ is almost never equal to $d(T^u, T^v)$ if $i \neq j \neq u \neq v$) and the above algorithm returns almost immediately with a single valid assignment.

5. Experiments and Evaluation

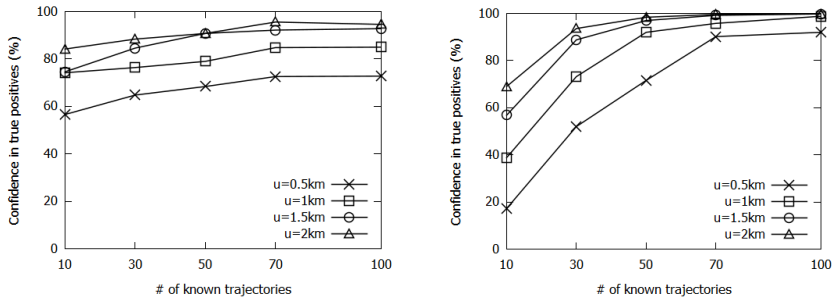
We ran our attack on two different datasets. The first dataset was generated using Brinkhoff's spatio-temporal data generator [46]. This is a well-known framework that generates network-based moving object trajectories, and is often used to benchmark spatio-temporal applications. We used the map of San Francisco to generate trajectories that each contained 500 locations. The second dataset is a real dataset obtained during the GeoPKDD project [4]. This dataset contains the GPS traces of taxis in Milan, acquired over a timespan of one month. We will refer to these datasets as San Francisco and Milan datasets, respectively.

We performed various experiments by changing the number of known trajectories (i.e., $|KT|$), the known trajectories themselves (hence, Δ) and the target trajectory T^r. In every experiment, we ran Algorithm 1 for several thousand iterations *itr* to obtain a reasonably large set of candidate trajectories. We observed that if we run Algorithm 1 on Milan for 220000 iterations with the following values of $|KT| = 10, 30, 50, 70, 100, 120$, the algorithm generates the following number of unique candidate trajectories respectively: 5743, 3483, 1292, 343, 100, 32. That is, when $|KT|$ is increased, we have fewer unique candidates. This can be justified as follows: When $|KT| = 10$, let $\mathcal{D}_{10}$ denote the set of all distance compliant trajectories to KT, per Definition 4. If 5 more trajectories were added to KT, some trajectories in $\mathcal{D}_{10}$ would still be distance compliant to the new KT, but some would not because of the additional distance compliance constraints imposed by the added trajectories. Therefore we have $\mathcal{D}_{15} \subseteq \mathcal{D}_{10}$. Following this intuition and using an inductive argument, we can prove the generalized result that $\mathcal{D}_{a+b} \subseteq \mathcal{D}_a$, where a and b are positive integers. Hence, higher $|KT|$ is expected to yield fewer possible candidates.

5.1. Positive Location Disclosure

As discussed earlier, the attack outputs the location disclosure confidence $\text{conf}_{p,u}$ of a (circular) area defined by a location p and radius u. This describes

[4] `http://www.geopkdd.eu/`



(a) Milan (b) San Francisco

Figure 4: Average confidence in true positives against different number of known trajectories

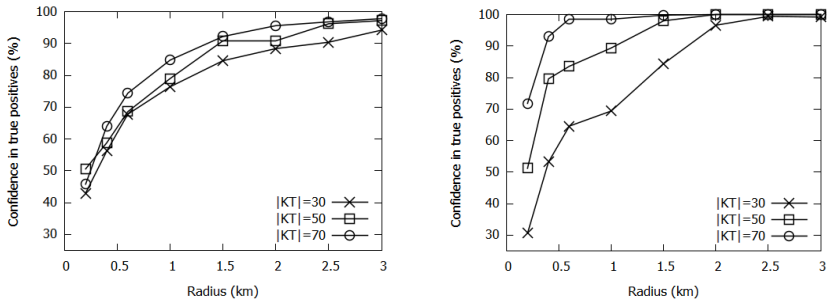
the adversary's level of confidence regarding where the victim has been, e.g., if $\text{conf}_{p,u} = 85\%$ then the attack asserts that the victim has been near p with probability 85%.

We say that *positive* location disclosure occurs when p is a location on the victim's trajectory, u is reasonably small and $\text{conf}_{p,u}$ is large. That is, at the end of the attack, the adversary is very confident that the target trajectory passes through the vicinity of p. Notice that these are essentially *true positives*, i.e., the victim was actually at/near p and the attack correctly finds that he was near p. The real-world use of such an attack is to set p to a sensitive location, e.g., a hospital or a school; and learning with very high confidence that the victim has been to the hospital.

To conduct experiments regarding positive location disclosure, we chose several locations on victims' trajectories as p and obtained $\text{conf}_{p,u}$ for various u. We repeated this experiment for different victims' trajectories T^r and known trajectories KT. We then quantified the adversary's average confidence in true positives (i.e., $\text{conf}_{p,u}$ where p is a location the victim has actually been to) versus the number of known trajectories (i.e., $|KT|$) and the radius u. The results are given in Fig. 4 and 5 respectively.

Analyzing Fig. 4, we make the following observations: On a real dataset (i.e., Milan), even with very few trajectories (e.g., 10) it is possible to infer (with confidence $> 55\%$) the whereabouts of the victim. As expected, by increasing the number of known trajectories we can increase the adversary's confidence, which implies a more successful attack. Also, $|KT|$ seems to affect the San Francisco dataset more than Milan. We believe that this is because the San Francisco dataset is synthetic and the attack ends up creating candidates that are too scattered around the city due to the random nature of the known trajectories.

Analyzing Fig. 5, we make the following observations: With a larger radius (i.e., u in $\text{conf}_{p,u}$) the area in question has a larger size, which decreases precision but increases the probability that a candidate passes through it. (In the extreme case, if u is the diameter of the map then $\text{conf}_{p,u}$ would always be 100%.)



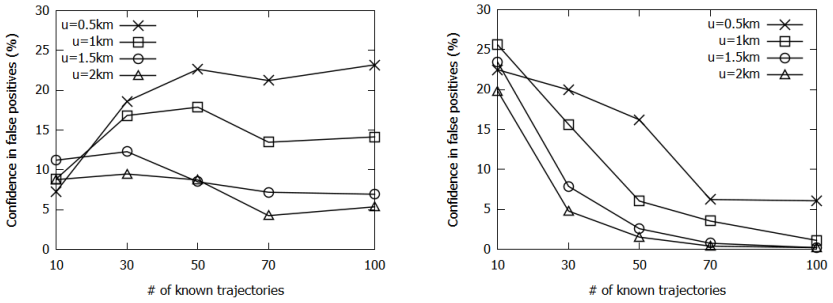
(a) Milan (b) San Francisco

Figure 5: Average confidence in true positives against different radiuses

In that sense, a positive correlation between u and $\text{conf}_{p,u}$ is expected, which was verified using the experimental results. Considering that a city block in Manhattan, NYC is $80m \times 274m$, the attack might not be able to identify precisely a street address, but can find that the victim was within a two-block radius with reasonable confidence ($> 60\%$) - see Fig. 5a. The implications of this is even more significant in non-urban settings. For example, if the adversary were to pursue whether the victim has gone near a large university campus out of town (e.g., boundaries $>$ 2km) in both datasets (i.e., Fig. 5a and 5b) the attack could output that he indeed has, with $> 85\%$ confidence.

So far, we focused on true positives. On the other hand, the attack may also yield *false positives*. We say that a false positive occurs when the location p in question is not located on (or very near) the target trajectory, but $\text{conf}_{p,u}$ is large. That is, the victim has not actually been near p but the attack says otherwise. An attack that outputs many false positives is highly undesirable, as it may lead to real-life problems. For instance, the attack claims that Alice has been to an illegal public protest, but in fact she was never there. To further motivate the need to decrease false positives, note that an attack can be devised to claim that all locations on a map have $\text{conf}_{p,u} = 100\%$, without making any calculations whatsoever. This trivially discovers all true positives and outputs perfect confidence for all of them. However, it is useless: The remaining map is full of false positives.

We note that only those locations that are visited at least once by some candidate trajectory have non-zero probability of being raised as a false positive. That is, for a location p, if no candidate passes through the vicinity of p, then trivially $\text{conf}_{p,u} = 0$ and p is never considered a false positive. On the other hand, if a candidate passes through p but the target trajectory does not, then $\text{conf}_{p,u} > 0$ and p can be a false positive. Thus, in this experiment we select those locations that appear at least once in some candidate trajectory, with an approximate distance of u to the target trajectory. Mathematically, we find $p \in T$ for $T \in CT$, where $\exists p_i \in T^r$ with $\mathbf{dist}(p, p_i) \approx u$; and $\forall p_j \in T^r$ such that



(a) Milan (b) San Francisco

Figure 6: Average confidence in false positives

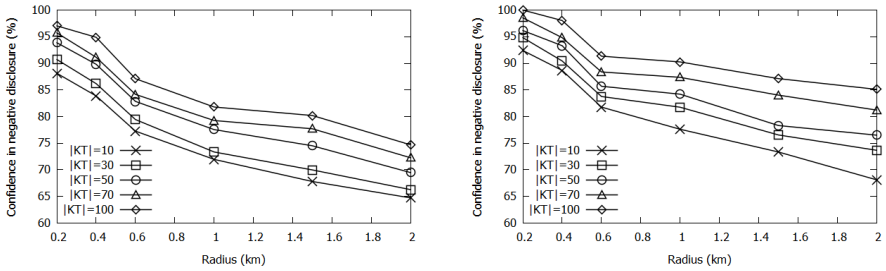
$p_i \neq p_j$, $\mathbf{dist}(p, p_j) \geq u$. In each experiment setting, we build a set of locations $\{p\}$ with these properties and measure their average $\text{conf}_{p,u}$. This measurement yields the average confidence in false positives: For locations that are not on the target trajectory, with what average confidence does the attack claim that the victim has been there?

In Fig. 6, we graph the average confidence in false positives with respect to various KT and u. In all cases, confidence in false positives is at most 25%. That is, the $\text{conf}_{p,u}$ of a location that is not on the target trajectory is less than 25% (in many cases, significantly less than 25%). We can therefore safely conclude that the attack does not raise false positives.

Two observations from Fig. 6 are: (1) With an increase in u, average confidence in false positives decreases. This is because we find locations u away from T^r; and with higher u we are farther away from T^r. Hence there is a decrease in the number of candidate trajectories that pass through those regions, and consequently a decrease in $\text{conf}_{p,u}$. (2) With an increase in $|KT|$, average confidence in false positives decreases. This is because higher $|KT|$ yields candidates that are less scattered and more concentrated on T^r, which decreases the probability of raising a false positive. In Fig. 6a, there seems to be an exception to this case, where average confidence increases from $|KT| = 10$ to 30. We believe that this is due to the length of the trajectories in Milan. We observed that on Milan, our attack algorithm can create only a few candidates with $|KT| = 10$. With $|KT| = 30$, there are more candidates and candidates appear more scattered, and with $|KT| = 50$, candidates condense on the target. On the other hand, on San Francisco, candidates get more and more condensed as we move from 2a to 2b and 2c.

5.2. Negative Location Disclosure

We say that *negative* location disclosure occurs when $\text{conf}_{p,u}$ is significantly small. That is, for a location p, an adversary is very confident that the target trajectory does not pass through the vicinity of p. A real-life use of this attack



(a) Milan (b) San Francisco

Figure 7: Average confidence in negative disclosure

could be to find if a student has been to school on a particular day, and the outcome would be that s/he most probably has not. We measure confidence in negative disclosure as $(1 - \text{conf}_{p,u}) * 100\%$. For example, let p denote the location of the school. If the attack yields $\text{conf}_{p,u} = 0.05$, then the adversary is 95% confident that the target has not been to school.

For this experiment, we could choose locations that are far away from the victim's trajectory, but this does not yield an interesting experiment. We can see from Fig. 2 that the $\text{conf}_{p,u}$ of a location far away from the victim is zero, or almost zero. For example, consider a location on the bottom-right corner of the map of San Francisco (Fig. 2). Not a single candidate trajectory passes through that region, so $\text{conf}_{p,u}$ for this location is 0, and confidence in negative disclosure is 100%. To make the experiment more challenging and meaningful, we choose only those locations roughly 3-4 km away from the target trajectory, and compute the adversary's confidence in negative disclosure for those locations.

We graph the results in Fig. 7. We make two observations from these graphs. First, with an increase in $|KT|$ we obtain higher confidence in negative disclosure. This is because when $|KT|$ increases, the candidates are more dense (i.e., concentrated) on the target trajectory, as can be observed in Fig. 2. This decreases the probability of being scattered near the trajectory, and $\text{conf}_{p,u}$ is smaller for the locations we measure. Thus, there is higher confidence in negative disclosure. Second, with an increase in the radius (i.e., u), we obtain smaller confidence in negative disclosure. Consider that in this case, $|KT|$ and the rest of the parameters are fixed, and the same candidates are generated. But, with higher u, more candidates satisfy $O_{p,u}$ and $\text{conf}_{p,u}$ increases, which in turn decreases confidence in negative disclosure.

Overall, even with limited background knowledge (e.g., $|KT| = 10$ or 30) and a reasonable radius (e.g., $u = 0.5, 1$ km), we obtain higher than 75% confidence in negative location disclosure. We remind that these are for locations that are only 3-4km away from the target. For locations farther away, we can expect even higher confidence.

5.3. Disclosure Quality

In this section we treat location disclosure as a binary classification task. Given a location p and radius u, we say that the attack classifies p as a visited location if $\text{conf}_{p,u} \geq th$, where th is a user-defined threshold. If $\text{conf}_{p,u} < th$, the attack classifies p as unvisited. Then, similar to the above sections, we make the following definitions:

- *True Positive (TP)*: $\text{conf}_{p,u} \geq th$ and the target T^r has actually visited p.
- *False Positive (FP)*: $\text{conf}_{p,u} \geq th$ but the target T^r has not visited p.
- *True Negative (TN)*: $\text{conf}_{p,u} < th$ and the target T^r has not visited p.
- *False Negative (FN)*: $\text{conf}_{p,u} < th$ but the target T^r has actually visited p.

Then, we apply the standard definitions of accuracy, precision, recall and F-score that are prevalent in the classification and information retrieval literatures, as follows:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \qquad \text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN} \qquad \text{F-score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

In Fig. 8 we show how these quality measures change with respect to two parameters: $|KT|$ and th. First, we observe that precision is positively correlated with both $|KT|$ and th. With more known trajectories, the attack is better able to identify visited locations p as true positives, therefore increasing precision. The increase in precision with higher th, on the other hand, can be explained using the decrease in false positives. As we showed in the previous section, the average confidence in false positives is low. However, when we set a low threshold (e.g., 0.5) to mark a candidate location as positive, there is higher chance that a false positive has confidence higher than that threshold. When we increase the threshold, very few FPs are obtained. In particular, observe that with $th \geq 0.7$, our precision is safely above 90% for any $|KT|$. Thus, there is small chance that our attack classifies a location that is not visited by T^r as visited.

Second, we analyze accuracy and recall. We observe once again that these measures are positively correlated with $|KT|$. We showed earlier in Fig. 2 that with more known trajectories, the candidate trajectories generated by the attack are better concentrated on or near T^r. As a result, higher $|KT|$ means that more true positives are obtained. In addition, the chance of missing a location p that is actually visited by T^r decreases. Thus, fewer false negatives are obtained. These improve both accuracy and recall. On the other hand, interestingly, as we increase th from 0.5 to 0.6, accuracy and recall improve, but increasing th

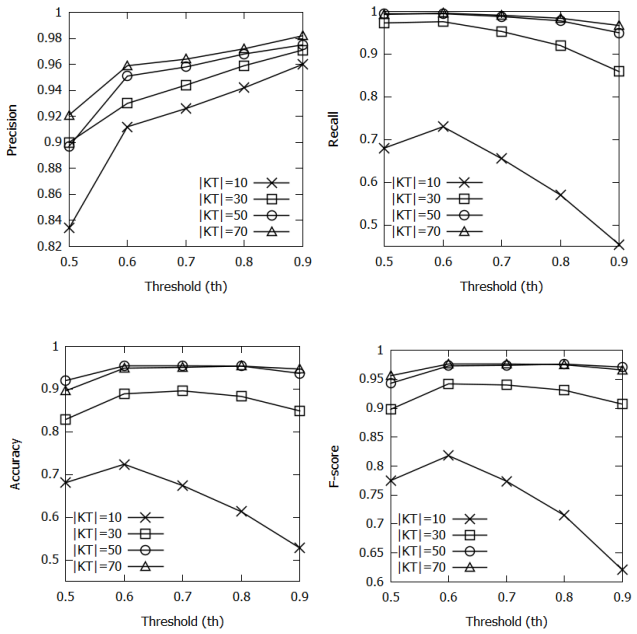


Figure 8: Measuring disclosure quality on Milan

further to $th \geq 0.7$ is damaging towards the two metrics. We observed that this is because, with higher th, it becomes more challenging to have $\text{conf}_{p,u} \geq th$, therefore the attack misses many of the true positives. Given the experimental results in Fig. 4 and 5, this is also an expected result: On Milan, the average confidence in true positives does not easily go above 0.8 or 0.9. Therefore, when we set $th = 0.8$ or $th = 0.9$, many true positives are missed by the attack. On the other hand, we emphasize that the recall of our attack is above 90% in all of the settings with $|KT| \geq 30$. Therefore, we can conclude that the vast majority of locations p that are visited by T^r are correctly found by the attack.

Finally, F-score is obtained through the precision and recall metrics, where higher F-score (closer to 1) implies better results. We observe similar tendencies in F-score compared to the precision and recall results. Therefore most of the above discussions apply.

5.4. Impact on Semantic and Sensitive Location Disclosure

In this section we discuss broader issues such as the semantic implications of the types of location disclosure achieved by our attack, when/how our attack becomes useful for disclosing sensitive locations and trajectories with high uniqueness (i.e., identifiability).

We first note that in the case of positive location disclosure, the proximity radius u is of critical importance. Consider that we set $p =$ hospital but unreasonably large u, e.g., $u = 15$km. In this case even if $\text{conf}_{p,u}$ is large, this is not

strong evidence that the victim has been near the hospital because there are expected to be many landmarks within 15km radius of the hospital. To ensure that the results of positive location disclosure are not misinterpreted, the attacker should set u according to the type (and geographic size) of the region of interest. In the case of a hospital, u can be few hundred meters. In the case of a college campus, a park, a social gathering etc., a radius of several kilometers would be more applicable.

Another interesting aspect is the semantic meaning of location disclosure, since the vicinity of some sensitive places may be highly visited. For example, a hospital can be close to bus stops and subway stations. Many individuals' trajectories will pass by these regions without stopping at the hospital. This brings forward the following question: If our attack outputs that the victim has been in the vicinity of a sensitive location, does this imply that the victim interacted with that location?

There is no simple answer to this question in the case of our attack and many other attacks (and location privacy works) in the literature in general. However, there are several ways in which the adversary can find if the victim indeed interacted with the sensitive location: (i) The victim's trajectory started or ended at the sensitive location, e.g., the victim drove his car to the hospital and parked inside the hospital's parking lot. (ii) Instead of passing near the location, the victim spent some time at that location. Even though we do not focus on the temporal aspect of trajectories in our attack, this can be implemented by modifying the proximity oracle $\mathcal{O}$. The oracle can be re-defined so that it outputs 1 if a trajectory has *multiple consecutive readings* in a sensitive region, instead of a single reading. Assuming that trajectories are collected at a constant sampling rate, the new oracle would allow measuring $\text{conf}_{p,u,t}$, where t is a user-defined amount of time that must be spent near p. This measurement would be interpreted as "the adversary's confidence that the victim spent a long time near location p".

Our final consideration is regarding the uniqueness or identifiability of victims' trajectories. In this work we focus on location disclosure, instead of constructing a single plausible trajectory T that resembles the victim's trajectory T^r. However, the latter is possible through representative trajectory generation, i.e., "averaging" all candidate trajectories and constructing one representative. Then, studying the uniqueness of this trajectory compared to the rest of the trajectories in the private database is interesting: If the representative trajectory only visits those locations that are frequented by many other trajectories (e.g., downtown areas) then the trajectory blends in a crowd, and linking this trajectory to the identity (e.g., name, Social Security Number) of a victim is difficult. On the other hand, if the trajectory visits some rural areas where few people live, then the victim's identity might be disclosed using an external dataset or additional background knowledge.

6. Conclusion

In this paper, we presented an attack for discovering, with significant confidence, if an unknown private trajectory passes (or does not pass) through regions of interest. The regions of interest could be arbitrary, and sometimes even passing through an area constitutes a privacy attack, e.g., the adversary could learn if the victim attended a public protest. The attack uses a set of known samples (i.e., KT) and their pairwise distances to the unknown trajectory. It works by generating a set of candidate trajectories that resemble the private trajectory, and then studying the properties of the candidate trajectories. Our experiments on real and synthetic datasets show that: (1) The attack can disclose, with high confidence, the locations that are visited and not visited by the private trajectory. (2) The attack has a low probability of raising false positives, i.e., identifying un-visited locations as visited.

We believe that having a set of known samples in a private database is a reasonable assumption in today's world. Also, with the introduction of trajectory querying services that rely for instance on encryption, but are not resilient to known-sample attacks, will make attacks like ours much more attainable. Naively assuming that these services are safe may lead to serious disclosure risks, as this paper shows. To combat this danger, one needs additional countermeasures: E.g., limit the number of trajectories that can be queried, or limit the number of times a trajectory can be queried by the same user, or block distance-retrieval queries altogether. These can be investigated as potential countermeasures in future work.

Also as future work, we plan to study distance metrics other than Euclidean distance, to see whether the attack is applicable to settings other than trajectory data. In addition, we point out that adversaries who have better knowledge of the trajectories or a city's road map can use different types of interpolation, e.g., polynomial or spline interpolation, rather than linear interpolation. Given two locations (x_1, y_1) and (x_2, y_2) retrieved by the attack, these locations can be marked on the map and the interpolation between them can be decided by taking into account the shape and directions of the roads between them. Such a study will enable the adversary to achieve better results when the sampling rate is low or the background information is limited.

References

[1] F. Giannotti, M. Nanni, F. Pinelli, D. Pedreschi, Trajectory pattern mining, in: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2007, pp. 330–339.

[2] M. Ghasemzadeh, B. C. Fung, R. Chen, A. Awasthi, Anonymizing trajectory data for passenger flow analysis, Transportation Research Part C: Emerging Technologies 39 (2014) 63–79.

[3] C. Y. Ma, D. K. Yau, N. K. Yip, N. S. Rao, Privacy vulnerability of published anonymous mobility traces, IEEE/ACM Transactions on Networking 21 (3) (2013) 720–733.

[4] R. Shokri, G. Theodorakopoulos, J.-Y. Le Boudec, J.-P. Hubaux, Quantifying location privacy, in: IEEE Symposium on Security and privacy (S&P), 2011, IEEE, 2011, pp. 247–262.

[5] M. Wernke, P. Skvortsov, F. Dürr, K. Rothermel, A classification of location privacy attacks and approaches, Personal and Ubiquitous Computing 18 (1) (2014) 163–175.

[6] M. Terrovitis, N. Mamoulis, Privacy preservation in the publication of trajectories, in: 9th International Conference on Mobile Data Management 2008, IEEE, 2008, pp. 65–72.

[7] J. Hua, Y. Gao, S. Zhong, Differentially private publication of general time-serial trajectory data, in: IEEE Conference on Computer Communications (INFOCOM), 2015, IEEE, 2015, pp. 549–557.

[8] A. Liu, K. Zhengy, L. Liz, G. Liu, L. Zhao, X. Zhou, Efficient secure similarity computation on encrypted trajectory data, in: IEEE 31st International Conference on Data Engineering (ICDE) 2015, IEEE, 2015, pp. 66–77.

[9] N. Pelekis, A. Gkoulalas-Divanis, M. Vodas, D. Kopanaki, Y. Theodoridis, Privacy-aware querying over sensitive trajectory data, in: Proceedings of the 20th ACM International Conference on Information and Knowledge Management, ACM, 2011, pp. 895–904.

[10] C. R. Giannella, K. Liu, H. Kargupta, Breaching euclidean distance-preserving data perturbation using few known inputs, Data & Knowledge Engineering 83 (2013) 93–110.

[11] M. Gowanlock, H. Casanova, In-memory distance threshold queries on moving object trajectories, in: Proceedings of the Sixth International Conference on Advances in Databases, Knowledge, and Data Applications, 2014, pp. 41–50.

[12] M. Gruteser, D. Grunwald, Anonymous usage of location-based services through spatial and temporal cloaking, in: Proceedings of the 1st International Conference on Mobile Systems, Applications and Services, ACM, 2003, pp. 31–42.

[13] B. Palanisamy, L. Liu, Attack-resilient mix-zones over road networks: architecture and algorithms, IEEE Transactions on Mobile Computing 14 (3) (2015) 495–508.

[14] B. Palanisamy, L. Liu, Effective mix-zone anonymization techniques for mobile travelers, GeoInformatica 18 (1) (2014) 135–164.

[15] B. Gedik, L. Liu, Protecting location privacy with personalized k-anonymity: Architecture and algorithms, IEEE Transactions on Mobile Computing 7 (1) (2008) 1–18.

[16] M. E. Andrés, N. E. Bordenabe, K. Chatzikokolakis, C. Palamidessi, Geo-indistinguishability: Differential privacy for location-based systems, in: Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, ACM, 2013, pp. 901–914.

[17] B. Hoh, M. Gruteser, Protecting location privacy through path confusion, in: First International Conference on Security and Privacy for Emerging Areas in Communications Networks (SecureComm), 2005, IEEE, 2005, pp. 194–205.

[18] C. A. Ardagna, M. Cremonini, S. De Capitani di Vimercati, P. Samarati, An obfuscation-based approach for protecting location privacy, IEEE Transactions on Dependable and Secure Computing 8 (1) (2011) 13–27.

[19] H. Kido, Y. Yanagisawa, T. Satoh, Protection of location privacy using dummies for location-based services, in: 21st International Conference on Data Engineering Workshops, 2005, IEEE, 2005, pp. 1248–1248.

[20] R. Shokri, J. Freudiger, M. Jadliwala, J.-P. Hubaux, A distortion-based metric for location privacy, in: Proceedings of the 8th ACM Workshop on Privacy in the Electronic Society, ACM, 2009, pp. 21–30.

[21] R. Chen, B. C. Fung, N. Mohammed, B. C. Desai, K. Wang, Privacy-preserving trajectory data publishing by local suppression, Information Sciences 231 (2013) 83–97.

[22] O. Abul, F. Bonchi, M. Nanni, Never walk alone: Uncertainty for anonymity in moving objects databases, in: 24th IEEE International Conference on Data Engineering, 2008, IEEE, 2008, pp. 376–385.

[23] M. E. Nergiz, M. Atzori, Y. Saygin, Towards trajectory anonymization: a generalization-based approach, in: Proceedings of the SIGSPATIAL ACM GIS 2008 International Workshop on Security and Privacy in GIS and LBS, ACM, 2008, pp. 52–61.

[24] J. Domingo-Ferrer, M. Sramka, R. Trujillo-Rasúa, Privacy-preserving publication of trajectories using microaggregation, in: Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Security and Privacy in GIS and LBS, ACM, 2010, pp. 26–33.

[25] R. Chen, B. Fung, B. C. Desai, N. M. Sossou, Differentially private transit data publication: a case study on the montreal transportation system, in: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2012, pp. 213–221.

[26] K. Jiang, D. Shao, S. Bressan, T. Kister, K.-L. Tan, Publishing trajectories with differential privacy guarantees, in: Proceedings of the 25th International Conference on Scientific and Statistical Database Management, ACM, 2013, p. 12.

[27] X. He, G. Cormode, A. Machanavajjhala, C. M. Procopiuc, D. Srivastava, Dpt: differentially private trajectory synthesis using hierarchical reference systems, Proceedings of the VLDB Endowment 8 (11) (2015) 1154–1165.

[28] A. Gkoulalas-Divanis, V. S. Verykios, A privacy-aware trajectory tracking query engine, ACM SIGKDD Explorations Newsletter 10 (1) (2008) 40–49.

[29] H. Zhu, X. Meng, G. Kollios, Privacy preserving similarity evaluation of time series data, in: International Conference on Extending Database Technology (EDBT), 2014, pp. 499–510.

[30] K. Liu, C. Giannella, H. Kargupta, An attackers view of distance preserving maps for privacy preserving data mining, in: Knowledge Discovery in Databases: PKDD 2006, Springer, 2006, pp. 297–308.

[31] K. Chen, G. Sun, L. Liu, Towards attack-resilient geometric data perturbation, in: Proceedings of the 2007 SIAM International Conference on Data Mining (SDM), SIAM, 2007, pp. 78–89.

[32] E. O. Turgay, T. B. Pedersen, Y. Saygin, E. Savas, A. Levi, Disclosure risks of distance preserving data transformations, in: SSDBM, Springer, 2008, pp. 79–94.

[33] E. Kaplan, T. B. Pedersen, E. Savaş, Y. Saygın, Discovering private trajectories using background information, Data & Knowledge Engineering 69 (7) (2010) 723–736.

[34] S. Sankararaman, P. K. Agarwal, T. Mølhave, J. Pan, A. P. Boedihardjo, Model-driven matching and segmentation of trajectories, in: Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, ACM, 2013, pp. 234–243.

[35] M. Vlachos, G. Kollios, D. Gunopulos, Discovering similar multidimensional trajectories, in: Proceedings of the 18th International Conference on Data Engineering, 2002, IEEE, 2002, pp. 673–684.

[36] H. Wang, H. Su, K. Zheng, S. Sadiq, X. Zhou, An effectiveness study on trajectory similarity measures, in: Proceedings of the 24th Australasian Database Conference, Australian Computer Society, Inc., 2013, pp. 13–22.

[37] J. Domingo-Ferrer, F. Sebé, J. Castella-Roca, On the security of noise addition for privacy in statistical databases, in: Privacy in Statistical Databases, Springer, 2004, pp. 149–161.

[38] K. Liu, C. Giannella, H. Kargupta, A survey of attack techniques on privacy-preserving data perturbation methods, in: Privacy-Preserving Data Mining, Springer, 2008, pp. 359–381.

[39] C. Dwork, A. Roth, The algorithmic foundations of differential privacy, Foundations and Trends in Theoretical Computer Science 9 (3-4) (2014) 211–407.

[40] F. D. McSherry, Privacy integrated queries: an extensible platform for privacy-preserving data analysis, in: Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data, ACM, 2009, pp. 19–30.

[41] P. Mohan, A. Thakurta, E. Shi, D. Song, D. Culler, Gupt: privacy preserving data analysis made easy, in: Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data, ACM, 2012, pp. 349–360.

[42] E. Esmerdag, M. E. Gursoy, A. Inan, Y. Saygin, Explode: an extensible platform for differentially private data analysis, in: 2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW), IEEE, 2016, pp. 1300–1303.

[43] C. Song, Z. Qu, N. Blumm, A.-L. Barabási, Limits of predictability in human mobility, Science 327 (5968) (2010) 1018–1021.

[44] Y.-A. De Montjoye, C. A. Hidalgo, M. Verleysen, V. D. Blondel, Unique in the crowd: The privacy bounds of human mobility, Scientific Reports 3 (2013) 1376.

[45] L. Yu, L. Liu, C. Pu, Dynamic differential location privacy with personalized error bounds, in: Network and Distributed System Security Symposium (NDSS '17), 2017.

[46] T. Brinkhoff, A framework for generating network-based moving objects, GeoInformatica 6 (2) (2002) 153–180.