PRIVACY RISKS OF SPATIO-TEMPORAL DATA TRANSFORMATIONS

by

EMRE KAPLAN

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

Sabancı University
January, 2017

PRIVACY RISKS OF SPATIO-TEMPORAL DATA TRANSFORMATIONS

Approved by:

Prof. Yücel Saygın .... [illegible]
(Thesis Supervisor)

Prof. Albert Levi .... [illegible]

Assoc. Prof. Cem Güneri .... [illegible]

Assoc. Prof. Şule Gündüz Öğüdücü .... [illegible]

Asst. Prof. Ali İnan .... [illegible]

Date of Approval: 05/01/2017

PRIVACY RISKS OF SPATIO-TEMPORAL DATA TRANSFORMATIONS

Emre Kaplan
Computer Science and Engineering
Ph.D. Thesis, 2017
Thesis Supervisor: Prof. Yücel Saygın

Keywords: privacy attack, spatio-temporal data, trajectory, distance preserving data transformation

Abstract

In recent years, we witness a great leap in data collection thanks to increasing number of mobile devices. Millions of mobile devices including smart phones, tablets and even wearable gadgets embedded with GPS hardware enable tagging data with location. New generation applications rely heavily on location information for innovative business intelligence which may require data to be shared with third parties for analytics. However, location data is considered to be highly sensitive and its processing is regulated especially in Europe where strong data protection practices are enforced. To preserve privacy of individuals, first precaution is to remove personal identifiers such as name and social security number which was shown to be problematic due to possible linking with public data sources. In fact, location itself may be an identifier, for example the locations in the evening may hint the home address which may be linked to the individual. Since location cannot be shared as it is, data transformation techniques have been developed with the aim of preventing user re-identification. Data transformation techniques transform data points from their initial domain into a new domain while preserving certain statistical properties of data.

In this thesis, we show that distance-preserving data transformations may not fully preserve privacy in the sense that location information may be estimated from the transformed data when the attacker utilizes information such as public domain knowledge and known samples. We present attack techniques based on adversaries with various background information. We first focus on spatio-temporal trajectories and propose an attack that can reconstruct a target trajectory using a few known samples from the dataset. We show that it is possible to create many similar trajectories that mimic the target trajectory according to the knowledge (i.e. number of known samples). The attack can identify locations visited or not visited by the trajectory with high confidence. Next, we consider relation-preserving transformations and develop a novel attack technique on transformation of sole location points even when only approximate or noisy distances are present. We experimentally demonstrate that an attacker with a limited background information from the dataset is still able to identify small regions that include the target location points.

KONUM ZAMAN VERİLERİNİN DÖNÜŞÜMÜNDE GİZLİLİK RİSKLERİ

Emre Kaplan
Bilgisayar Bilimi ve Mühendisliği

Doktora Tezi, 2017
Tez Danışmanı: Prof. Dr. Yücel Saygın

Anahtar Sözcükler: gizlilik atakları, konum zaman verisi, hareket yörüngeleri, mesafe koruyan veri dönüşümü

Özet

Son yıllarda artan mobil cihazlar sayesinde üretilen ve saklanan verinin miktarında büyük artışlar gerçekleşmektedir. Milyonlarca mobil cihaz (akıllı telefon, tablet ve hatta giyilebilir teknolojiler) GPS çipi ile topladığı verileri konum-zaman verisi ile eşleştirerek saklamaktadır. Yeni nesil uygulamalar konum verisine dayalı geliştirilmekte olup topladıkları bu veriler üzerinden yürütülen analiz çalışmalarıyla ticari fayda sağlamaktadırlar. Toplanan bu veriler, analiz için üçüncü parti kimselerle de paylaşılabilir. Konum verisi hassas kabul edilerek işlenmesi, başta Avrupa'da olmak üzere kanunlarla belirlenmiş olup, veri işleme için öncelikle veri koruma uygulamaları tatbik edilmesi zorunlu kılınmıştır. Paylaşım esnasında kişinin sadece kimlik bilgilerinin çıkarılması mahremiyeti korumaya yetmemektedir. Kamuya açık bilgiler ile eşleştirilerek mahremiyet açıklarına sebebiyet verdiği bilinmektedir. Örneğin kişinin akşam saatindeki konumu ev adresini işaret etmektedir ve buradan kimliğine dair bilgilere erişilebilir. Konum verisinin bu şekilde açıklara yol açmaması için veri dönüşüm teknikleri geliştirilmiştir. Veri dönüşüm teknikleri, veriyi, istatistiksel özelliklerini koruyarak, bir tanım kümesinden başka bir tanım kümesine dönüştüren ve böylece kişinin kimliğini gizlemeyi hedefleyen mahremiyet koruyucu tekniklerden biridir. Bu tez çalışmasında, mesafe koruyan veri dönüşüm tekniklerinin de mahremiyeti koruma açısından güvenilir olmadığını göstermekteyiz. Bu çalışmada iki farklı atak yöntemi ortak bir atak senaryosunu icra etmektedirler.

Çalışmalarımızı konum verisi alanına yoğunlaştırıp konum ve hareket yörüngeleri üzerinde detaylandırdık. Bu çalışmalarda saldırganın veri tabanına dayalı, erişebildiği tüm kaynaklardan edinebileceği bilgileri de kullanarak gerçekleştireceği ataklar ile mahremiyet açıkları ortaya çıktığını göstermekteyiz. Bu ataklar ile hedef hareket yörüngesinin eldeki bilgiler ışığında benzerlerinin tekrar oluşturulmasının mümkün olduğu gösterilmiştir. Ayrıca bu ataklar ile bir hareket yörüngesinin geçtiği veya geçmediği yerler hakkında yorum yapmak mümkün hale gelmektedir. Konum verisi üzerinde olan diğer çalışmamızda geliştirdiğimiz teknik ile, mesafe koruyan dönüşüm teknikleri ile dönüştürülen bir veri tabanının ilişkileri yayınlandığında, saldırgan bu veriler üzerinden veri tabanındaki diğer konum bilgilerine erişebilmekte ve mahremiyet ihlallerini göstermektedir. Bu çalışmada, saldırgan büyük bir şehirde toplanan konum veri tabanı hakkında biraz bilgi ile hedef konumları sokak seviyesinde bulabilmektedir.

To my grandparents Müzeyyen, Mehmet Ali and my aunt Nermin

Acknowledgments

I wish to express my sincere gratitude to Prof. Yücel Saygın, for his continuous support, guidance, patience and help in both my thesis and graduate studies. He has always been helpful, positive, and supportive.

I am especially grateful to Assoc. Prof. Mehmet Ercan Nergiz for his continuous support throughout my thesis work. Without his support, his guidance, and his great ideas, it would be not be possible to carry out this research.

I also thank Mehmet Emre Gürsoy for valuable discussions and comments throughout my thesis work.

I would like to thank the thesis committee for their helpful comments. Last, but not the least, I would like to thank my family, especially my dear mother for their patience and support throughout my life.

Contents

1 Introduction **1**

1.1 Contributions 5

1.2 Outline 7

2 Related Work **9**

2.1 Privacy Preserving Techniques 9

2.2 Preserving Privacy in Spatio-Temporal Data 11

2.3 Attacks on Data Transformations 15

3 Preliminaries **17**

4 Location Disclosure Risks of Releasing Trajectory Distances **26**

4.1 Brief Summary 26

4.1.1 Problem Setting 27

4.2 Attack Algorithm 29

4.2.1 Overview of the Approach 29

4.2.2 Creating a Generic Trajectory 32

4.2.3 Solving for a Candidate Trajectory 33

4.2.4 Robustness to Noise 39

4.3 Experiments and Evaluations 45

4.3.1 Experiment Setup 45

4.3.2 Results and Evaluations 46

4.3.3 Comparison with Previous Work 60

5 Location Disclosure Risks of Releasing Relation-preserving Data Transformations 63

5.1 Brief Summary . . . 63

5.2 Attack Algorithm . . . 64

5.2.1 Attack Formalization . . . 68

5.2.2 Implementation and Noise Resilience . . . 71

5.3 Experiments and Evaluations . . . 74

5.3.1 Experiment Setup . . . 74

5.3.2 Results and Evaluations . . . 75

6 Conclusions and Future Work 83

List of Figures

3.1 Linear interpolation of partial trajectories 19

3.2 A 90° counter-clockwise rotation 23

4.1 Building a generic trajectory T^g 33

4.2 Attacking a trajectory in Milan when $|KT| = 10$ 47

4.3 Attacking a trajectory in Milan when $|KT| = 30$ 48

4.4 Attacking a trajectory in Milan when $|KT| = 50$ 49

4.5 Attacking a trajectory in San Francisco when $|KT| = 10$ 50

4.6 Attacking a trajectory in San Francisco when $|KT| = 30$ 51

4.7 Attacking a trajectory in San Francisco when $|KT| = 50$ 52

4.8 Average confidence in true positives against different number of known trajectories (Milan) 53

4.9 Average confidence in true positives against different number of known trajectories (San Francisco) 54

4.10 Average confidence in true positives against different radiuses (Milan) . . 54

4.11 Average confidence in true positives against different radiuses (San Francisco) 55

4.12 Average confidence in false positives (Milan) 57

4.13 Average confidence in false positives (San Francisco) 57

4.14 Average confidence in negative disclosure (Milan) 59

4.15 Average confidence in negative disclosure (San Francisco) 60

5.1 Sample 2-dimensional database $\mathcal{D}$ with three records. Actual locations of records in $\mathbb{R}^2$ (on the left) and the distance matrix published after transformation (on the right). 66
5.2 Discretization of the universe using uniform 2-dimensional cells. 72
5.3 Attacking a target with knowns=2 (half of the space is pruned) 75
5.4 Attacking a target with knowns=2 (target lies in the perimeter of a known sample) 76
5.5 Attacking a target with knowns=4 76
5.6 Attacking a target with knowns=4 (very small unpruned region) 76
a Gowalla dataset 78
b Istanbul dataset 78
5.7 Success rate (in percentage) in noisy and non-noisy scenarios 78
a Gowalla dataset 80
b Istanbul dataset 80
5.8 Accuracy in the noisy scenario 80
a Gowalla dataset 82
b Istanbul dataset 82
5.9 Effects of the voting threshold on accuracy 82

List of Tables

3.1 Creating the distance matrix of a spatial database 21
3.2 Trajectories and distances 22
3.3 A relation-preserving transformation of $\mathcal{D}$ 24

4.1 Comparison with previous work 61

List of Algorithms

1 Find location disclosure confidence 31
2 Find candidate trajectory 37
3 Locating a target record using a distance matrix and known samples 70

Chapter 1

Introduction

We live in a digitalized world with various smart devices which assist us in all aspects. Digital devices such as smart phones, smart watches, bracelets, and vehicles embedded with Global Positioning Systems (GPS) devices collect massive amounts of data every second. In our highly interconnected world, collected data can easily be shared among different devices and service providers seamlessly through cloud systems. Although data is collected in many different forms, it has one common key attribute that is the location with time-stamp. Time-stamped location data is mostly collected in the form of GPS coordinates as latitude, longitude pairs together with time-stamp of the measurement. This form of data is also called spatio-temporal data where 'spatio' stands for the location and 'temporal' stands for the time dimension. Seamless collection of spatio-temporal traces is even more pervasive with the use of applications deployed on aforementioned devices. This is because users with or without notice and care often report their location information via the applications that they frequently use (e.g. while taking a photo, checking in to a venue, texting to each other or even playing video games).

When we consider the mobile applications, almost all free applications collect information about its users as much as it is allowed to do. In order to use and benefit from the application, one should share his or her data. Some features of the applications depend on the location data (such as check-in) where user cannot use the application unless he shares his location data.

For instance, Swarm needs your location in order for the user to check-in which is the core functionality of the application. Facebook uses location information when the user uploads a picture in order to add location and time. In sum, most of the new generation companies base their core businesses over user data. They tend to analyze and sell data analytics as product and individuals' private information becomes the raw material of such businesses. Outcome of data analytics is about the tendency of users i.e., how frequently they visit places, how long they stay at a certain location, popular places and roads, traffic information, and many more derivatives regarding to the business needs. Data analytics is used to identify hot spots, campaign to targeted users (such as marketing to potential buyers) and hints more information about the user (his social status, wealth, etc.) which means that a company knows its users much more than his desire to share his private information. On the other side, those data can be used for research purposes to enhance technology. In all cases, users are concerned about their data and most of the time have no other chance but trust the data processors. For instance, one can analyse the user transitions from the distance and time-stamp differences between the check-ins [1] to detect user activity patterns. The relation of the social ties from the social networks with the user movement and its temporal dynamics can be captured through the location based social networks [2]. Moreover, location based social network data allows to measure user similarity to cluster similar users for targeted advertisement, product enhancement and even discover shared similar preferences and interests [3, 4]. Through location data analytics, one may discover the friendships as discussed in [5] by analysing behavioural characteristics. On the other hand, the analysis of this data can also help the society, e.g., via traffic management in metropolitan areas though the analysis of traffic and passenger flows [6, 7], road condition sensing [8], and fleet management. The potential value of location from a business perspective is clear and in the last decade, we witnessed a leap in machine learning and data analytics. Many techniques have been proposed [4, 6, 9, 10], to extract more and more value out of personal data. While sharing and mining spatio-temporal trajectory data is beneficial for the society, the sensitive nature of location data arises privacy concerns. This has led to substantial research in location privacy [11, 12], and privacy-preserving trajectory data management [13, 14, 15].

Privacy [16, 17, 18] can be informally expressed as the right of an individual in controlling the release and use of his own information. Location data is considered sensitive. In the context of privacy of individuals and digital assets, which yield business value, data holders should always be on alert and design their data release keeping privacy in mind to ensure data owners' privacy requirements. As the digital information grows very fast, combining different sources of information becomes easier from the attacker's perspective. Even more, the data sources may contain released data in anonymized, transformed or perturbed forms. Data swapping and shuffling, additive or multiplicative perturbations and aggregation-based methods are used for privacy protection. For instance, one can use noise addition or rotation perturbation as further discussed in Section 2.1. Our work applies on distance-preserving transformations and relation-preserving transformations. In the first case, the released data consists of pairwise distances of the trajectories in the dataset. In the latter, the released dataset contains only the relative order of the location point pairs. Note that we do not need to know neither the exact location points nor the pairwise distances but only the relative ordering.

Various privacy-preserving data processing and publishing techniques have been proposed [19, 20, 21, 22, 23, 24] to achieve the a desired level of privacy while retaining valuable statistical properties of the data. These techniques tend to transform or perturb data so that the privacy requirement is satisfied [25, 26, 27, 28].

As a motivating scenario, consider logistics companies with fleets that track their vehicles through a vehicle tracking service. These logistics companies have an incentive to cooperate. For example, if two vehicles that have similar routes are half-loaded, they could be merged into a single vehicle, thus reducing the cost and also reducing the carbon emission. If these two vehicles belong to the same company, then this can be done by just querying the local trajectories. However if they belong to different companies then things may get complicated since location data is commercially critical for competing companies. For example, rival mobile sales teams may observe the regions that are visited or not visited to estimate the sales figures in different areas, which may be commercially critical. Therefore, companies may not want to share their exact trajectory data with third parties. As an alternative, a distance matrix can be constructed from the trajectories by the service provider from the collective set of trajectories in order to do analytics such as finding the common routes. This way logistics companies can do distance based queries on the released distances without seeing the exact trajectories which is one of the motivating applications of releasing distances as a dissimilarity matrix. Without such a matrix, each fleet can only do local load balancing and merging of the loads for the similar routes. Publishing the matrix enables them to do global optimization without seeing each others trajectories. One may argue that, as an alternative scenario, the service provider may keep the distance matrix and do data analytics. However, in practice the fleet management service providers are specialized in machine-to-machine (M2M) data collection and they may not expected to do data analytics.

A fleet company can provide its own data and the dissimilarity matrix to data analytics companies for analysis and revealing hidden valuable information such as common routes across the all fleets in the system or pinpoint potential uncovered regions by the other fleets.

Even though privacy preserving techniques exist and data holders apply them, they should be very careful while releasing their transformed datasets due to potential privacy leaks which may not be foreseen prior to the release. Those leaks can be in various ways such as recovering the original data.

In order to empower the privacy attacks, other data sources such as public datasets or any piece of information that may be related to the transformed spatio-temporal dataset can be incorporated. There are plenty of attack techniques such as mentioned in [12, 29, 30, 31, 32] pointing to privacy risks of transformed spatio-temporal datasets, which were considered to respect privacy after applying privacy preserving methods. For instance, in [33], location traces of mobile device owners are highly unique and can be re-identified using few location points. They showed that with four data points they can identify up to 95% of the individuals in a European country. The study reveals the sensitivity of the location data from privacy perspective.

In our work, we show that we can generate sufficient candidate trajectories of the target trajectory with 50 known trajectories. With 50 known samples, we can show if the target passes through a given region with confidence over 80% to 100%. Our false positive rate is around 5%. We also show that negative disclosure of the the target around 80%.

In the latter work which we focus on sole location points, our success rate is more than 95% and even up to 100% with 10 known samples. We studied noisy data and show that our success rate decreases only by 10%.

To sum up, release of transformed data may lead to privacy risks which may not be foreseen without a careful analysis. Our thesis is that, privacy breaches are possible in the case of transformed spatio-temporal data release. We show that an attacker can utilize his background knowledge to reveal data which is otherwise considered to be safe. Our work concludes that the data transformation techniques in question is not satisfactory to meet the privacy demands when the attacker knows even few data points from the transformed dataset.

1.1 Contributions

In this thesis, we impersonate the attacker and explore privacy risks due to the release of the transformed spatio-temporal datasets. The attacker knows a few data points from the original dataset as background information. Utilizing his knowledge, he carries out a known-sample attack.

Consider each location data in the form of *(user id, latitude, longitude, time-stamp)*. The data owner can remove the *user_id* from the dataset prior to data release.

Moreover, the data owner may apply data transformation and perturbation techniques to mask original data in the released dataset. The goal is to make sure that the transformed data can not be linked back to the original data in order to ensure privacy. We assume that a distance-preserving transformation is applied on the dataset such that only the pairwise distances of the trajectories are released. In our second work, we study the privacy leaks when the relation-preserving transformation is applied so that released data contains only the ranking of the data points with respect to their pairwise distances.

We assume that the attacker knows a few data points from the original dataset. The attacker may obtain those points, for instance, as being a user of the system and storing his own data or may collaborate with others who are using the system. For sure, the attacker may benefit from any side information from public domain (e.g. over the Internet, social hacking, etc.) and use it to enhance his attack.

Our work has two folds:

1. We show the privacy risks when the transformed dataset of trajectories (collection of traces) is released and the attacker has only access to mutual distances in the form of a distance matrix beside his own few background data.

2. We analyze the privacy risks when the transformed dataset of locations (e.g. check-in data) is released but the attacker has access to the relation of mutual relations of the location points. The attacker has access to very limited background information about the dataset (i.e. knows few of locations).

In the first part, we focus on trajectory datasets. The mutual distances between trajectories are released for data mining purposes. The attacker has his own trajectory data (collected over a time with his own device). The attacker use his own data together with the released mutual distances to discover the remaining trajectories in the dataset. Moreover, the attacker can infer if the target individual passed through a given area on the map. If the inference is made, the attacker can discuss about the confidence of the inference, which tells us if the inference is strong or not in order to conclude the target individual passed through or not. Specifically, given a set of known trajectories and their distances to a private, unknown trajectory, we devise an attack that yields the locations that 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.

In the second part, we focus on inferring the possible locations of the target. The attacker can access to the relations of the dataset. The relations of the data points are derived from their pairwise distances. The attacker presumed to log his own locations that is also shared with the application. By using his own location data and the relation information that is released, the attacker tries to infer the target location (i.e. location trace of an individual at a time). The attacker limits the space using the relations and his own data until he can't limit the space anymore. The remaining region containing the target location becomes the output of the attack. As the remaining region gets smaller, it is easier to pinpoint the target location. Experiments on real datasets show that attacker can narrow the space up to 1% of the entire space. Considering the entire space as a city, target location can be pinpointed at a street level.

In our studies, we focus on privacy risks of transformed spatio-temporal data release of two different data types: trajectories and locations. In the case of indirect data release through transformation and perturbation, we can demonstrate attacks that recover a target location or a trajectory.

In both studies, we assume that the attacker has a few data points and show that the attacker can recover target trajectories from the mutual distances, infer if the target passes through a given location with high confidence. We also show that when the attacker has access to the relations, then he can discover the potential regions where the target location resides.

1.2 Outline

The organization of this thesis is as follows:

We describe the contributions and the motivation behind the discussions in this thesis in Section 1. We discuss the related work and the background information regarding to our work in Chapter 2. Preliminaries of the work including the definitions and the basic concept are defined in Chapter 3. We describe two privacy attacks that embrace each other from the attacker's perspective. In Chapter 4, we discuss location leaks from a transformed dataset preserving the distances of the spatio-temporal dataset. We point out the leaks in the form of location points, given an area or even a segment of a trajectory. We further analyse the probability of the outputs and verifying the target is actually there, turns out to be the performance indicator of the attack. In Chapter 5, we discuss location leaks from a transformed spatio-temporal dataset resulting an area from the map that the target data of the individual resides in. The attacker's performance measured through how small the regions, the attack concludes. Finally, in Chapter 6 we conclude the thesis stating the results and discuss the future work.

Chapter 2

Related Work

Our work is related to privacy preserving data transformation techniques. We studied the privacy risks of data transformation techniques from the attacker's perspective. Since our work is on spatio-temporal datasets, location privacy is utmost important. Our work involves privacy of trajectories and locations. Although the trajectories are formed of a series of locations, privacy risks of such datasets show characteristic differences. We organize this chapter as follows: In Section 2.2, we discuss the location privacy techniques, in Section 2.1, we provide a general overview of the data privacy literature discussing perturbation and aggregation methods from recent works. Finally, in Section 2.3, we describe previous works on attacking various types of data transformations to prepare the reader for our work discussed in Chapter 5.

2.1 Privacy Preserving Techniques

Data swapping and shuffling. Perhaps the oldest and most basic techniques in privacy preservation are the simple swapping or shuffling of data values [34],[35]. A desirable property of these techniques is that the shuffled values have the same marginal distribution as the original values. Hence, univariate analyses on shuffled data yield the same results as the original data [36]. On the other hand, no guarantees can be given in terms of attribute correlation and multivariate analyses. Thus, although these techniques were studied in the earlier days of data privacy, they are no longer popular in the literature. In addition, neither swapping nor shuffling guarantees that its output will be distance-preserving.

Additive perturbation. Another common technique is based on *noise addition* [37, 38]. In this technique, instead of releasing the original data X, the data owner releases $Y = X + R$ where R is a collection of random values (sometimes called *white noise*) drawn from a statistical distribution. The distribution is often Gaussian or uniform. Additive perturbation techniques have been heavily criticized in the literature, as several studies have shown that it is possible to estimate original data from perturbed data, thus violating privacy [23, 39, 40]. Additive perturbation is often not distance-preserving, but may preserve relation depending on the magnitude of noise. We evaluate our attack with and without additive noise in Section 5.2.2.

Multiplicative perturbation. Multiplicative perturbation techniques can either perfectly or approximately preserve distances between tuples. Oliveira and Zaiane introduced *rotation perturbation* in [41] and showed its applicability to data clustering. In [42] and [43], Chen et al. showed that rotation perturbation is also useful in classification, as many classifiers are rotation-invariant, i.e., they are unaffected by arbitrary rotations. Rotation-invariant classifiers include k-NN, SVM (polynomial and radial basis) and hyperplane-based classifiers. Note that rotation perturbation perfectly preserves distances between tuples, and are therefore susceptible to the attack presented in our work discussed in Chapter 5.

In contrast, *random projection* based methods approximately preserve distances between tuples [19]. Giannella et al. argue that by tuning the parameters of projection, one can ensure arbitrarily high probabilities of preserving distances [44]. They point to [30] for preliminary results in this direction. Even though distance preservation is desirable from a utility point of view, it also makes our attack more plausible. As we show in Section 5.3, higher distance preservation increases the success rate of our attack.

Aggregation-based methods. Aggregation relies on grouping similar tuples together. Then, one can sanitize and release either some statistical aggregates [45], some representative tuple [46] etc. from each group. Among popular aggregation-based methods are

k-anonymization and micro-aggregation.

Sweeney and Samarati proposed k-anonymity [26], and sparked a plethora of work in this area. We refer the interested reader to [47] for a survey. In k-anonymity, each tuple is grouped with $k-1$ other tuples and these tuples' values are generalized so that an adversary that knows quasi-identifying information regarding an individual can, at best, map this individual to a group of k tuples.

Micro-aggregation assigns tuples into groups of size at least k, and then computes and releases average values per group [48]. Groups are formed based on similarity. The recent work of Domingo-Ferrer et al. [49] provides a detailed overview of micro-aggregation and its applications.

Aggregation-based methods are, in general, not distance or relation-preserving. A straightforward example demonstrates this: Two tuples with non-zero distance can be placed in the same group and aggregated (or generalized) to the same set of values, in which case the distance between them will be zero.

Differential privacy. Differential privacy is a recent definition of statistical database privacy [50]. It ensures that the computation of an algorithm stays insensitive to changes in one tuple. The protection of differential privacy is different than the data model we consider - differential privacy releases statistical properties after noise addition, whereas in our studies, we assume that tuples (or pairwise distances between tuples) are released after a transformation. Thus, our works are not applicable to differential privacy.

2.2 Preserving Privacy in Spatio-Temporal Data

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 crowd-sourcing. Since it is implausible to review all of these areas in detail, we present only some of the major approaches and findings.

In [51], 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 [52] describe the state of the art approaches in building mix-zones over road networks. We refer the reader to [53] for a comparison between mix-zones and spatial cloaking. Gedik and Liu [54] offer privacy in mobile systems via the application of *location k-anonymity*. Andres et al. [55] introduce the notion of *geo-indistinguishability*, a generalization of differential privacy for location-based services. Alternative approaches to protect location privacy include path confusion [56], data obfuscation [20] and addition of dummies [22].

There have also been efforts to unify the aforementioned approaches. Shokri et al. [28] 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 [11], 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. [12] offer a survey on attacks and defenses in location privacy.

The main differences between our work 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 detailed in Chapter 4 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 our work.

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 our work in Chapter 4, we do not assume that a trajectory database must be shared with the adversary in order 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 [13], 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 [57], where the authors implement the $(K, C)_L$ privacy model for trajectory anonymization. Abul et al. [58] propose (k, δ)-*anonymity*, similar to k-anonymity but with an additional δ factor to account for location imprecision. Nergiz et al. [59] introduce generalizations in the area of trajectory anonymization, and study extensions of k-anonymity as their privacy model. Domingo-Ferrer et al. [60] 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 [61], 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. [62] 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. [14] use differentially private generalizations and merging of trajectories to publish trajectory data. In contrast, He et al. [63] 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 [64], 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 [65], supporting many types of queries useful for spatio-temporal data mining, e.g., range, distance and k-NN queries. Liu et al. [15] 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. [66], where authors describe a protocol to compute the distance between two time-series in a client-server setting. In [67], 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 our work discussed in Chapter 4. Liu et al. [32] 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. [42] 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. [31] consider cases where the adversary knows input samples as well as distances between these samples and unknown, private objects. More recently, Giannella et al. [44] 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 our first work is trajectories. Kaplan et al. [29] present a distance-based, known-sample attack on trajectories. While the main goal of [29] 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.

2.3 Attacks on Data Transformations

We start this section with attacks on distance-preserving data transformations, and then describe attacks on other types of transformations (e.g., approximate distance-preserving transformations, additive perturbation etc.)

In [32], Liu et al. develop two attacks on distance-preserving data transformations: (1) The attacker has a set of known samples, which are i.i.d. from the same distribution as the private (original) data. The attack is based on mapping the principal components of the sample (which represents the distribution of the original data) to that of the perturbed data. This helps the attacker estimate the perturbation matrix. Since this attack is a known-sample attack, it is comparable to ours. However, as pointed out in [44], it requires a significant number of samples that accurately represent the distribution of the original data, otherwise it will be unsuccessful. (2) The second attack is a known input-output attack, in which the attacker has a set of original data tuples and their perturbed versions. The attacker then constructs a perturbation matrix that would yield the input-output pairs. They assume the attacker has several (v, v') pairs and reverse-engineer the matrix R that would satisfy $v' = Rv$ for the pairs the attacker has. In this attack, the attacker's background information is different and stronger than ours.

In [31], Turgay et al. extend the attacks in [32] by assuming that the attacker only has a similarity/distance matrix (instead of the perturbed data) and the global distribution of the original data. They develop attacks based on principal component analysis, with and without known samples. Our proposed attack works without knowledge of the global distribution.

Mukherjee et al. [68] use a perturbation algorithm based on Fourier transform to achieve privacy. The proposed approach approximately preserves distances between tuples. Their privacy relies on a random permutation of the Fourier transform parameters. Therefore, they analyze cases where the permutation is known by the attacker. However, they do not consider known sample attacks. Since one of their goals is to preserve distances, their approach could well be susceptible to attacks on distance-preserving and relation-preserving transformations, such as our work.

Both [19] and [69] study independent component analysis (ICA) based attacks on multiplicative perturbation. Specifically, in [19], Liu et al. consider ICA-based attacks on random projection. In [69], Guo and Wu assume that the attacker knows some data columns and aims to retrieve the remaining columns using ICA. On the other hand, Chen and Liu [43] argue that ICA attacks are ineffective against random perturbation and rotations.

Closely related to our work is Giannella et al.'s attack in [44]. In their study, Giannella et al. assume that the attacker has a set of known samples, and focus on the case where the number of known samples is less than the number of data dimensions. Their attack links the known samples to their perturbed tuples, and furthermore, for unlinked perturbed tuples they estimate the probability of retrieving their original values.

Finally, for a recent and more detailed survey on deriving information from data transformations and perturbed data, we refer the reader to [70].

Chapter 3

Preliminaries

In this chapter, we provide the main terms and the notions that we use in our works. These are the building block components given in the form of definitions. Note that some of the definitions are common for the entire work while the remaining definitions are referred only within its own chapter.

We first formally define what a trajectory is in Definition 1. In order to discuss the interpolation based attack detailed in Chapter 4, we formally define the linear interpolation function in Definition 2. Then, we also describe an interpolated trajectory in Definition 3. Formal definition of Euclidean Distance which is used in entire work given in Definition 4. We maintain our attack scenarios over the distance matrix defined in Definition 5. Based on Euclidean distance, we define the distance between trajectories in Definition 6. We also present the notion of distance compliant trajectory to differentiate generated trajectories according to predefined constraints formally described in 7. We discuss the notion of proximity oracle in Definition 8. We introduce the location disclosure confidence in Definition 9. In our latter work detailed in Chapter 5, we describe distance-preserving transformation in Definition 10. Then, we define the notion of relation-preserving transformation in Definition 11. In order to discuss the attack operators, we formally define the high dimension concepts of Hypersphere, Hyperball in Definition 12 and Definition 13 respectively. Then, we present the notion of equidistant hyperplane in Definition 14. We finally introduce the notion of Half-space in Definition 15 and conclude this chapter.

Definition 1 (Trajectory). *We represent a trajectory T as a finite list of locations with time-stamps, as follows: $T = ((p_1, t_1), (p_2, t_2), ..., (p_n, t_n))$. t_i corresponds to the time-stamp, and a trajectory is sorted by its time-stamps, 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 time-stamps $Q = (t_1, t_2, ..., t_n)$, 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 time-stamps 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 time-stamps for which a location entry exists in all trajectories, and drop those time-stamps 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 2 (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 time-stamp 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 a imperfect trajectory with missing entries. For each missing entry (i.e.,

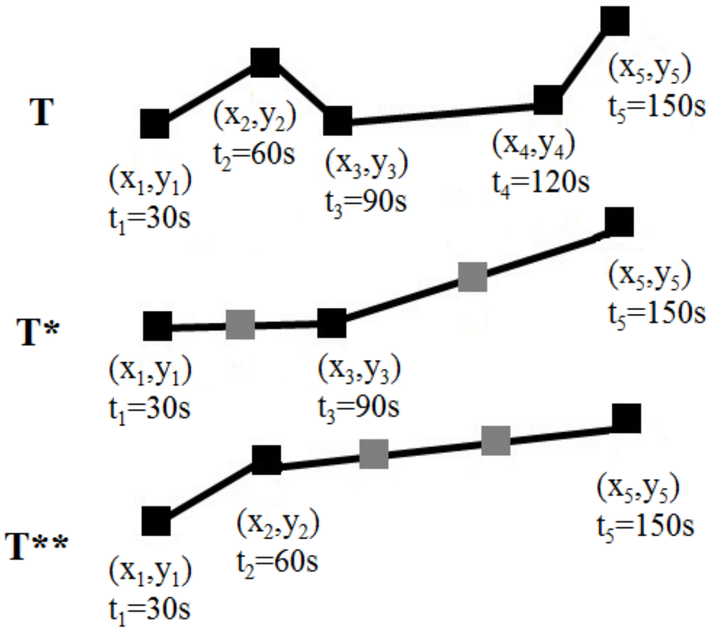


Figure 3.1: Linear interpolation of partial trajectories

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 time-stamp such that $(p_i, t_i) \in T$; and t_j, $t_j > t_k$, be the smallest time-stamp 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 time-stamps and we end up with the interpolated trajectory.

Definition 3 (Interpolation)**.** *Let T be a trajectory and Q be the list of desired time-stamps. 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 $t_i \in Q$, $(p_i, t_i) \in T^*$.*
- *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 time-stamps 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 time-stamp t_k to find the distance of p_k from p_i and p_j.

We now illustrate interpolation using examples. In Figure 3.1, 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 4.3. 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, 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 [44], and can be used as a basis for building more complex distance measures for trajectories (e.g., Dynamic Time Warping [71], Longest Common Subsequence [72]). The interested reader is referred to [73] for a thorough discussion.

Definition 4 (Euclidean distance). *Let x and y be two data points in $\mathbb{R}^m$, with coordinates $x = (x_1, ..., x_m)$ and $y = (y_1, ..., y_m)$. We say that the Euclidean distance between x and y is: $\delta(x, y) = ||x - y|| = \sqrt{\sum_{i=1}^{m} (x_i - y_i)^2}$, where $||.||$ denotes the L^2-norm.*

Definition 5 (Distance matrix)**.** *The distance matrix of a database $\mathcal{D}(r_1, .., r_n)$ is an $n \times n$, symmetric, real-valued matrix M such that $M_{i,j} = M_{j,i} = \delta(r_i, r_j)$.*

Table 3.1: Creating the distance matrix of a spatial database

(a)

ID	Coordinates
r_1	(34.0, 122.6)
r_2	(13.1, 57.8)
r_3	(2.5, 51.9)
r_4	(98.4, 193.2)

(b)

	r_1	r_2	r_3	r_4
r_1	0	68.1	77.4	95.6
r_2	68.1	0	12.1	160
r_3	77.4	12.1	0	170.8
r_4	95.6	160	170.8	0

We introduce the distance matrix (also known as the dissimilarity matrix presented in [41]) that captures pairwise distances between records in a database. For example let $\mathcal{D}$ be a spatial database containing (latitude, longitude) coordinates of 2D data points. A sample database $\mathcal{D}$ is shown in Table 3.1a. $\mathcal{D}$'s distance matrix is given in 3.1b. As an example, we compute one of the entries in the distance matrix: $M_{1,2} = \delta(r_1, r_2) = \sqrt{(34.0 - 13.1)^2 + (122.6 - 57.8)^2} = 68.1$.

Definition 6 (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 3.2, we provide three simple trajectories and calculate the distances between them.

Definition 7 (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 $i \in [1, k]$.*

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 3.2: Trajectories and distances

Definition 8 (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:*

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

Definition 9 (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:*

$$\textit{conf}_{p,u}(CT) = \frac{\sum\limits_{T \in CT} \mathcal{O}_{p,u}(T)}{|CT|}$$

Definition 10 (Distance-preserving transformation)**.** *A function $\mathcal{T} : \mathbb{R}^m \to \mathbb{R}^m$ is a distance-preserving transformation if for all $x, y \in \mathbb{R}^m$, $\delta(x, y) = \delta(\mathcal{T}(x), \mathcal{T}(y))$.*

Let $\mathcal{D}$ be the data owner's private database. Instead of releasing $\mathcal{D}$, for privacy protection the data owner first perturbs $\mathcal{D}$ using a distance-preserving transformation $\mathcal{T}$ and then releases the perturbed data $\mathcal{D}' = (\mathcal{T}(r_1), ..., \mathcal{T}(r_n))$. By definition, $\mathcal{T}$ preserves pairwise distances between records, and thus the distance matrix M is constant before and after $\mathcal{T}$.

Informally, a distance-preserving transformation satisfies the condition that the distance between a pair of tuples in the transformed data is the same as their distance in the original data. Many popular clustering and classification algorithms rely solely on distances between tuples. Such algorithms are unaffected by distance-preserving transformations. In [74], Chen and Liu show that the following are unaffected: k-NN classifiers, kernel methods, SVM classifiers using polynomial, radial basis and neural network kernels, linear classifiers, and some clustering and regression models. Considering this,

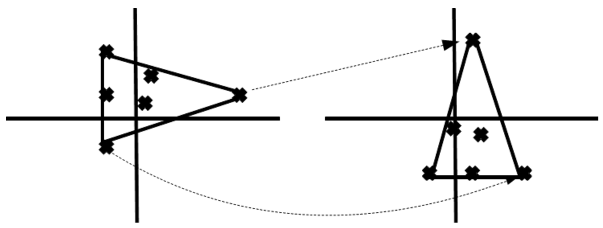


Figure 3.2: A 90° counter-clockwise rotation

distance-preserving transformations are of great interest.

The three fundamental techniques for distance-preserving transformations are *translations*, *reflections* and *rotations* [41, 43]. Translations shift tuples a constant distance in parallel directions. Reflections map tuples to their mirror images in fixed-dimensional space. Rotations can be written in various ways, one of which is in terms of a matrix multiplication: Let v be a column vector containing the coordinates of a 2D data tuple. Then, $v' = Rv$ is a rotation perturbation where the 2×2 rotation matrix R is defined as:

$$R = \begin{bmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{bmatrix}$$

The angle θ is the *rotation angle*, measuring the amount of clockwise rotation. Similar rotation matrices can be written for 3D and 4D space. Any rotation matrix R is an orthogonal matrix, and thus has a determinant of $+1$ or -1.

In more recent work, Huang et al. [75] present FISIP. They aim to preserve first order and second order sums and inner products of tuples. Then, they retrieve distances and correlations based on these properties.

In Figure 3.2, we present a rotation on 2D data as a sample distance-preserving transformation. The original data on the left is kept private and never released. The rotated data on the right is released. The adversary that only has the rotated data and no other background information cannot determine the perturbation angle θ, and thus cannot retrieve the original data. In this example, the perturbation angle is 90° in counter-clockwise direction.

Definition 11 (Relation-preserving transformation)**.** *A function $\mathcal{S} : \mathbb{R}^m \to \mathbb{R}^m$ is a* *relation-preserving transformation if for all $x, y, z, t \in \mathbb{R}^m$ and for arithmetic comparison operators op $\in \{<, >, =\}$, $\delta(\mathcal{S}(x), \mathcal{S}(y))$ op $\delta(\mathcal{S}(z), \mathcal{S}(t))$ if and only if $\delta(x, y)$ op $\delta(z, t)$.*

Theorem 1. *Every distance-preserving transformation is relation-preserving.*

Proof. Let $\mathcal{T} : \mathbb{R}^m \to \mathbb{R}^m$ be a distance-preserving transformation. Since $\mathcal{T}$ is distance-preserving, for all $x, y, z, t \in \mathbb{R}^m$, $\delta(\mathcal{T}(x), \mathcal{T}(y)) = \delta(x, y)$ and $\delta(\mathcal{T}(z), \mathcal{T}(t)) = \delta(z, t)$. Therefore, trivially for any comparison operator *op*, $\delta(\mathcal{T}(x), \mathcal{T}(y))$ *op* $\delta(\mathcal{T}(z), \mathcal{T}(t))$ if and only if $\delta(x, y)$ *op* $\delta(z, t)$. $\square$

We use Theorem 1 to show that distance-preserving transformations are a subset of relation-preserving transformations. Notice that the converse of Theorem 1 is not true: A relation-preserving transformation is not necessarily distance-preserving. For example, let $\mathcal{D}'$ in Table 3.3b be obtained via transforming $\mathcal{D}$ in Table 3.1 in a relation-preserving manner. One can verify that the pairwise relation of distances in Table 3.3b is the same as in Table 3.1, but none of the distances actually stay the same.

Table 3.3: A relation-preserving transformation of $\mathcal{D}$

(a)

ID	Coordinates
r'_1	(32.7, 123.6)
r'_2	(14.5, 60.3)
r'_3	(8.8, 54.1)
r'_4	(100.0, 196.9)

(b)

	r'_1	r'_2	r'_3	r'_4
r'_1	0	65.9	73.5	99.5
r'_2	65.9	0	8.4	161.2
r'_3	73.5	8.4	0	169.4
r'_4	99.5	161.2	169.4	0

Definition 12 (Hypersphere)**.** *In n-dimensional Euclidean space, a hypersphere $S_{C,r}$ is defined by a fixed centre point $C \in \mathbb{R}^n$ and a radius r, and denotes the set of points in the n-dimensional space that are at distance r from C. That is, each point $X(X_1, X_2, ..., X_n)$ on $S_{C,r}$ satisfies: $r^2 = \sum_{i=1}^{n}(X_i - C_i)^2$.*

Definition 13 (Hyperball)**.** *In n-dimensional Euclidean space, given a hypersphere $S_{C,r}$, the hyperball $B_{C,r}$ denotes the space enclosed by $S_{C,r}$. $B_{C,r}$ is said to be closed if it includes $S_{C,r}$ and open otherwise.*

Definition 14 (Equidistant hyperplane)**.** *In n-dimensional Euclidean space, the locus of points equidistant from two points $A, B \in \mathbb{R}^n$ is a hyperplane H_{AB} that contains all points P that satisfy the equality $||P - A|| = ||P - B||$.*

Definition 15 (Half-space)**.** *A half-space is either of the two parts into which a hyperplane divides the n-dimensional Euclidean space. A half-space is said to be closed if it includes the hyperplane, and open otherwise.*

Chapter 4

Location Disclosure Risks of Releasing Trajectory Distances

4.1 Brief Summary

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 [15] or [67], 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 work 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. Therefore, we 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 [15, 67]. 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 robustness to noise: 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 exact distances.

4.1.1 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 4.1, 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 defined in Definition 4. We do not consider how the Euclidean distance is actually retrieved, it can be done due to for example a published database, or following the secure distance calculation protocol for trajectories given in [15].

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 defined as distance compliant trajectory defined in Definition 7.

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 4.2.3. 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* which is defined in Definition 9.

The proximity oracle, defined in Definition 8, 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.

4.2 Attack Algorithm

4.2.1 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.

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 sample size discussed above. Higher itr yields a larger set of candidates, which increases the accuracy of the attack, but decreases efficiency. 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*. 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. The *FindCandidate* method is invoked in line 6, which calculates the interpolated locations and returns a set of candidate trajectories. There can be cases where a valid candidate cannot be built with the given parameters (due to e.g., $n_{i,k}$ being misplaced), and in such cases the set returned by *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 conf$_{p,u}$ according to Definition 9. 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.

ALGORITHM 1: Find location disclosure confidence

```
Input  : $KT$: set of known trajectories
         $\Delta$: 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: conf$_{p,u}$: location disclosure confidence
1  $CT \leftarrow \{\}$
2  $j \leftarrow 0$
3  $t \leftarrow \left\lfloor \frac{|KT|}{2} \right\rfloor$
4  while $j < itr$ do
5      Randomly create an ordered set of non-negative integer indices $S = (s_1, ..., s_{t-1})$ such that
       $\sum_{i=1}^{t-1} s_i = |T^r| - t$
6      $R \leftarrow$ FindCandidate($KT, \Delta, S$)
7      $CT \leftarrow CT \cup R$
8      $j \leftarrow j + 1$
9  end
10 $cnt \leftarrow 0$
11 for $T \in CT$ do
12     if $\mathcal{O}_{p,u}(T) = 1$ then
13         $cnt \leftarrow cnt + 1$
14     end
15 end
16 return $cnt/|CT|$
```

4.2.2 Creating a Generic Trajectory

In this section we describe the *FindCandidate* algorithm, given in Algorithm 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 Figure 4.1. 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 equi-distant from one another, and also from the main points. This is because we do not know any information regarding the time-stamps 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

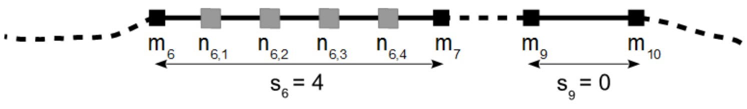


Figure 4.1: Building a generic trajectory T^g

2 we write:

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

The choice of time-stamps 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 2, 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: $(m_1, n_{1,1}, m_2, n_{2,1}, m_3)$.

4.2.3 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 $T^g = (p^g_1, p^g_2, ..., p^g_n) = ((x^g_1, y^g_1), (x^g_2, y^g_2), ..., (x^g_n, y^g_n))$ be the generic trajectory. As T^g needs to be distance compliant, due to Definition 7 we have:

$$\sum_i \|p^g_i - p^j_i\| = (\delta_j)^2 \tag{4.1}$$

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

$$\sum_i ||p_i^g - p_i^1|| = (\delta_1)^2 \tag{4.2}$$

$$\sum_i ||p_i^g - p_i^{j+1}|| - ||p_i^g - p_i^j|| = (\delta_{j+1})^2 - (\delta_j)^2 \tag{4.3}$$

where, in Equation 4.3, $j \in [1, |KT| - 1]$. The derivation of Equation 4.3 from Equation 4.1 is simple: We write Equation 4.1 for j and $j + 1$, and subtract the former from the latter, side by side. One can see that Equations 4.2 and 4.3 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.

$$\begin{aligned} ||p_i^g - p_i^j|| &= (x_i^g - x_i^j)^2 + (y_i^g - y_i^j)^2 \\ ||(x_i^g, y_i^g) - (x_i^j, y_i^j)|| &= (x_i^g - x_i^j)^2 + (y_i^g - y_i^j)^2 \\ ||(x_i^g - x_i^j, y_i^g - y_i^j)|| &= (x_i^g - x_i^j)^2 + (y_i^g - y_i^j)^2 \\ (x_i^g - x_i^j)^2 + (y_i^g - y_i^j)^2 &= (x_i^g - x_i^j)^2 + (y_i^g - y_i^j)^2 \end{aligned}$$

The first step is due to the definition of locations: $p = (x, y)$. The second step follows from the properties of arithmetic operations defined in Chapter 3. Finally, the last step is due to the definition of the Euclidean norm. □

Theorem 2. *For a fixed $j \in [1, |KT| - 1]$ and $i \in [1, n]$, Equation 4.3 can be reduced to a linear equation of the form:*

$$\sum_i (\alpha_{i,j})x_i^g + (\beta_{i,j})y_i^g = \gamma_j \tag{4.4}$$

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 4.3:

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

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 $d^2 = (y_i^g - y_i^j)^2$. By $a^2 - b^2 = (a-b)(a+b)$ and $c^2 - d^2 = (c-d)(c+d)$, we get:

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

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

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

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 4.4, which concludes the proof. $\square$

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 4.4 are the coordinates of the generic trajectory, i.e., x_i^g and y_i^g.

We would like to underline that Equation 4.4 is linear, whereas Equation 4.1 and Equation 4.3 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 2 builds a linear equation for one j among the set of known trajectories. Since Equation 4.3 holds for $j \in [1, |KT|-1]$, a linear equation can be built for all $j \in [1, |KT|-1]$. Therefore, we obtain a system of $|KT|-1$ linear equations. Plus, we have one quadratic equation due to Equation 4.2. We can solve this system for $|KT|$ unknowns. In the case when the generic trajectory is not been interpolated, then we would have $2 \times |T^g|$ 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 2. Then, we obtain one quadratic equation using Equation 4.2. 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

ALGORITHM 2: Find candidate trajectory

```
Input  : $KT$: set of known trajectories
         $\Delta$: 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 $\gamma_j = (\delta_{j+1})^2 - (\delta_j)^2 - \sum_i ||p_i^{j+1}|| + \sum_i ||p_i^j||$
    $EQNS \leftarrow EQNS \cup Q$
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 4.2 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 $R$
```

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 the following observations as side channel information in this work:

1. Geographic assumptions: The locations in each trajectory should be within the boundaries of a certain map. For example, if trajectories originate from vehicles traveling in a city, then all locations would fall within the boundaries of that city.

2. Mobility characteristics: Consecutive locations in a trajectory are not independent. For example, speed limits and road segments in urban areas constrain the mobility of a vehicle.

Due to these reasons, not all generic trajectories T^g or solutions T^{sol} yield a plausible candidate for the target trajectory. However, when Algorithm 1 is run for a reasonable number of iterations, as will be shown experimentally, Algorithm 2 usually builds a large and accurate set of candidate trajectories.

Example 3. We present an example for building and solving the system of equations using Algorithm 2. For the sake of simplicity, let trajectories contain a single location, and thus S be an empty set. 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.

4.2.4 Robustness to Noise

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 robust 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 [30] and [39]). Second, Gaussian noise is also used in the scope of differential privacy (i.e., (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 calibrated to $\Delta f \times ln(1/\delta)/\varepsilon$ (where Δf is the sensitivity of f) to the true output of f satisfies (ε, δ)-DP [50]. Although we assume Gaussian noise, a number of derivations below apply to other noise distributions with mean 0. (e.g. differential privacy also employs Laplace noise with mean 0 to achieve privacy.) We note such instances where applicable.

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 information 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 techniques (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 variant. 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 2 affected by noise? The three parameters in question are $\alpha_{i,j}$, $\beta_{i,j}$ and γ_j.

Theorem 3. *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 4. *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:

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

$\square$

Theorems 3 and 4 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 5. *$\hat{LHS}$ 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).

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

Proof.

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

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

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

$\square$

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 $R\hat{H}S$ denote its noisy version. We show below that the bias of $R\hat{H}S$ is fixed.

Theorem 7. *$R\hat{H}S$ is a biased estimator of RHS, with a fixed bias of σ^2.*

Proof.

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

$E[\mathcal{X}_1] = 0$, 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 5.)

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

□

Similar to Section 4.2.4, 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 robust to noise.

4.3 Experiments and Evaluations

4.3.1 Experiment Setup

We ran our attack on two different datasets. The first dataset was generated using Brinkhoff's spatiotemporal data generator [76]. This is a well-known framework that generates network-based moving object trajectories, and is often used to benchmark spatiotemporal 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 [1]. 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.

4.3.2 Results and Evaluations

To demonstrate the attack, we first present Figure 4.2, 4.3, 4.4 and Figure 4.5, 4.6, 4.7. In all these figures, the target trajectory is marked in red and the candidate trajectories are marked in blue. Notice that the candidates are close to the target even with few known trajectories, i.e., $|KT| = 10$. However, the candidates in this case are crude rather than smooth: In Figure 4.2, it looks as if the candidates are collections of lines with sharp edges. They do not follow the smooth movement patterns observed in the target trajectory. However, when we have $|KT| = 30$ or 50, the candidate trajectories are much more refined. First, they cover a smaller area, condensing more on the target. Second, they are smoother, better resembling the curvatures and movement patterns of the target. Yet, even when $|KT| = 10$, based on the candidate trajectories, the adversary: (1) has a rough idea on the whereabouts of the target trajectory, and (2) can rule out more than half of the map as not visited by the victim. These will be quantified in detail next sections, as we discuss positive and negative disclosure.

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 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%.

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

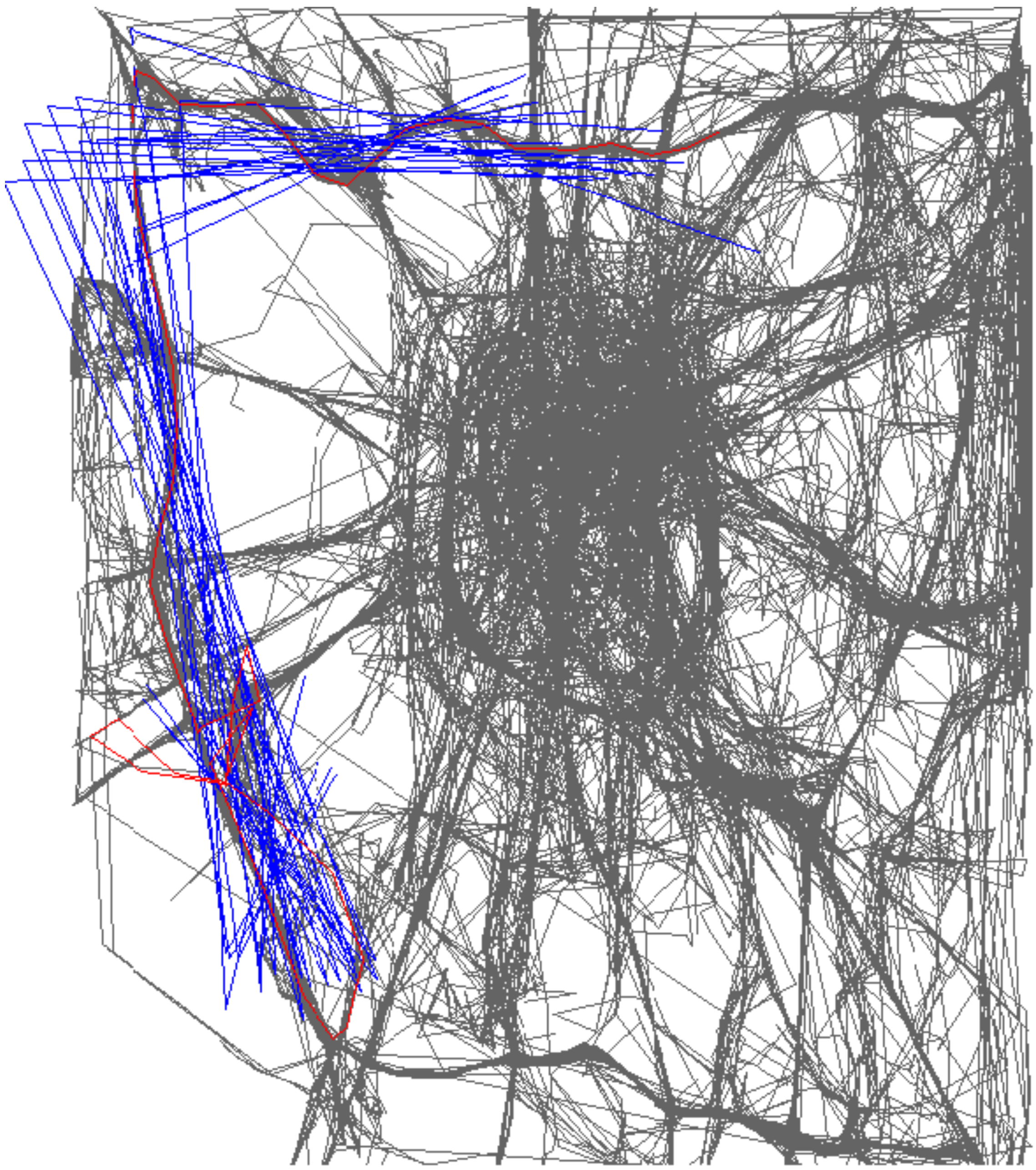


Figure 4.2: Attacking a trajectory in Milan when $|KT| = 10$

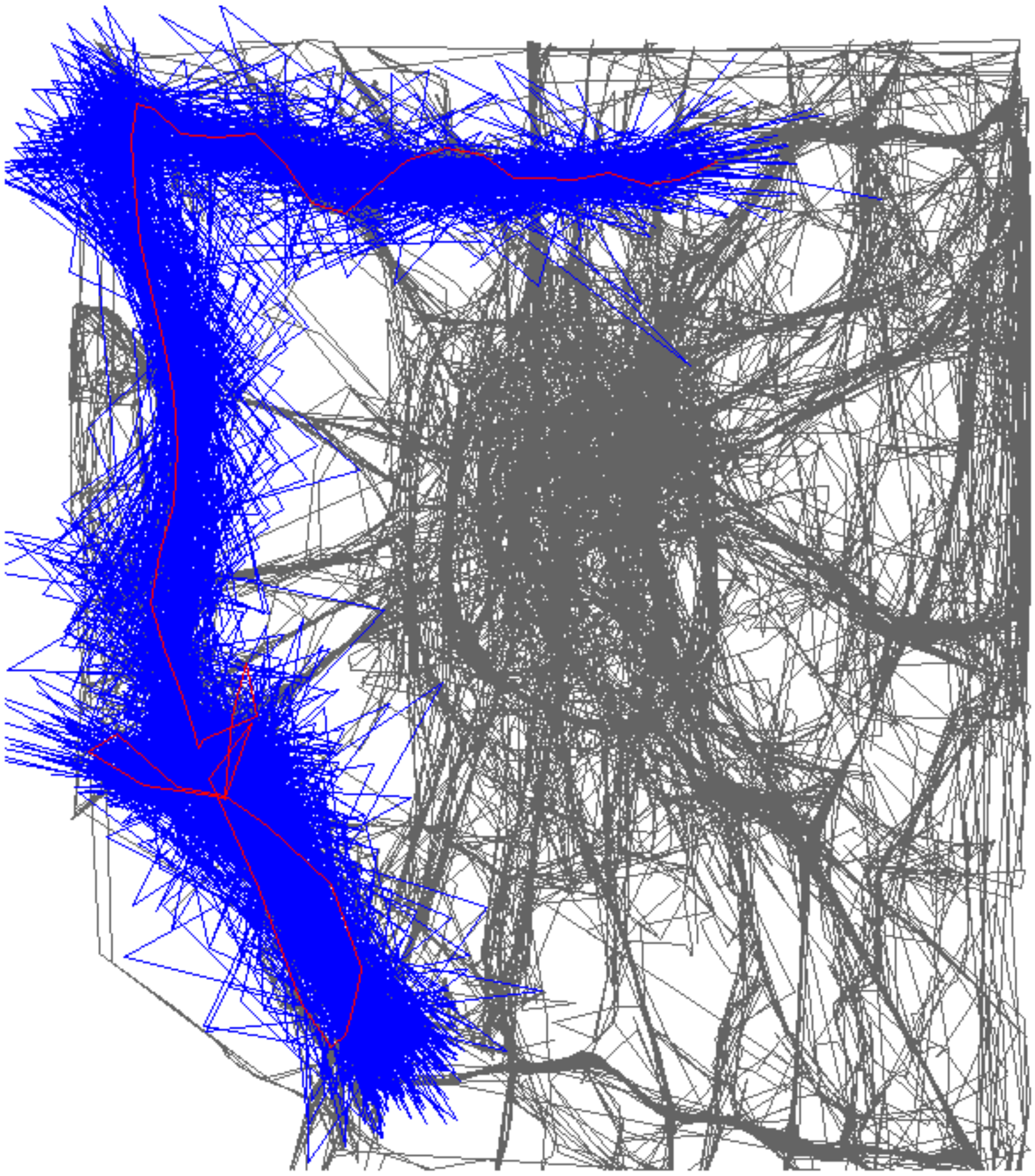


Figure 4.3: Attacking a trajectory in Milan when $|KT| = 30$

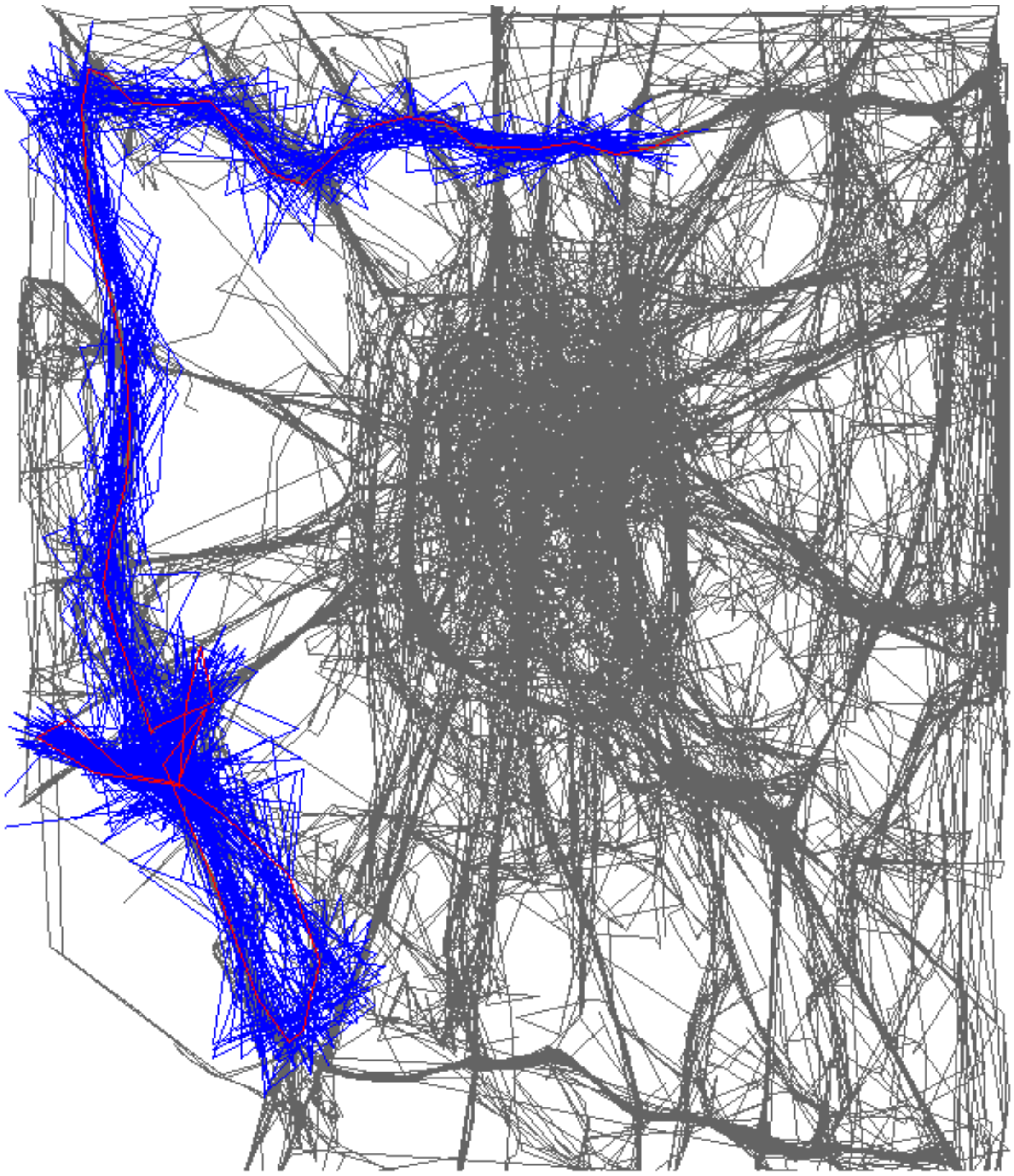


Figure 4.4: Attacking a trajectory in Milan when $|KT| = 50$

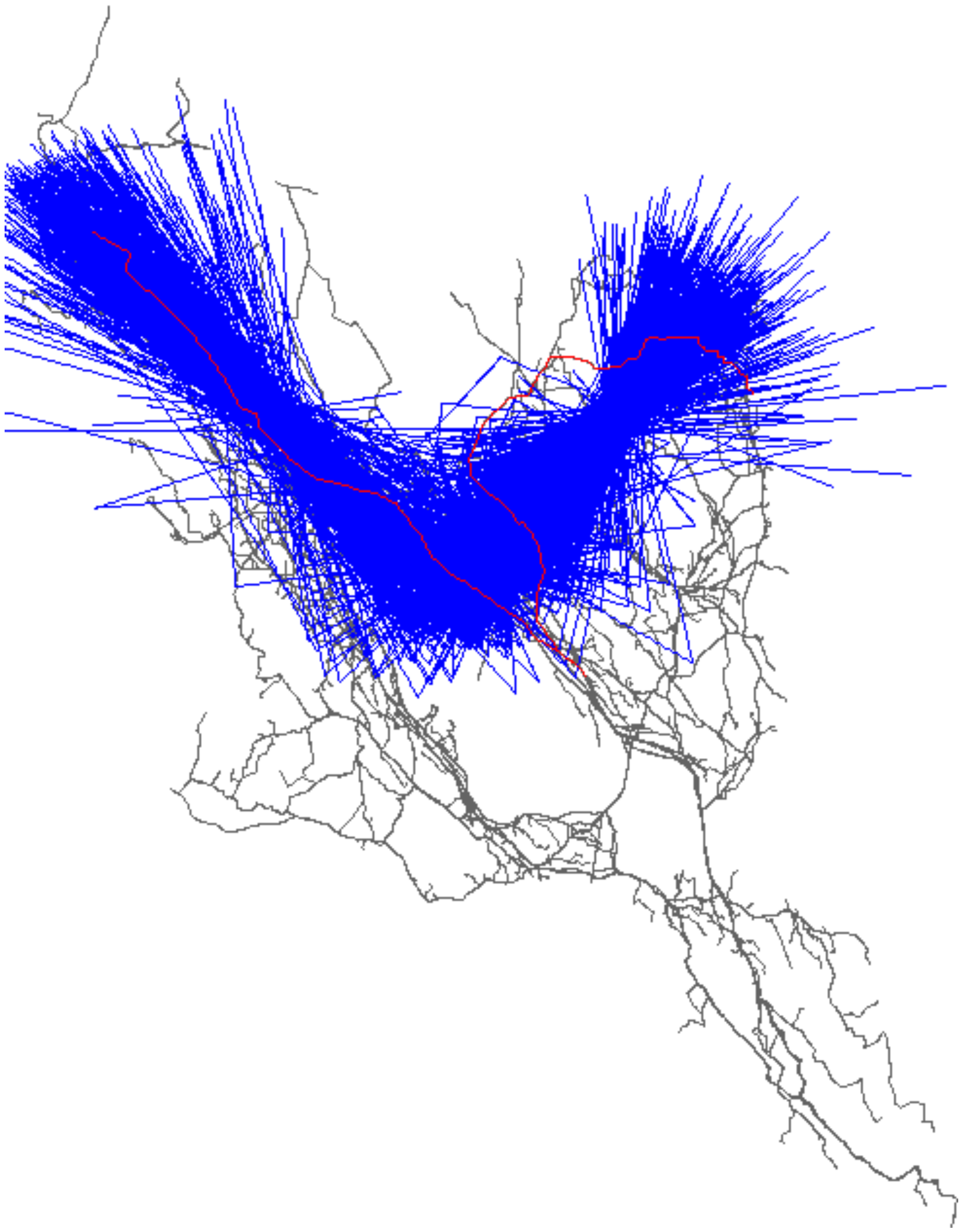


Figure 4.5: Attacking a trajectory in San Francisco when $|KT| = 10$

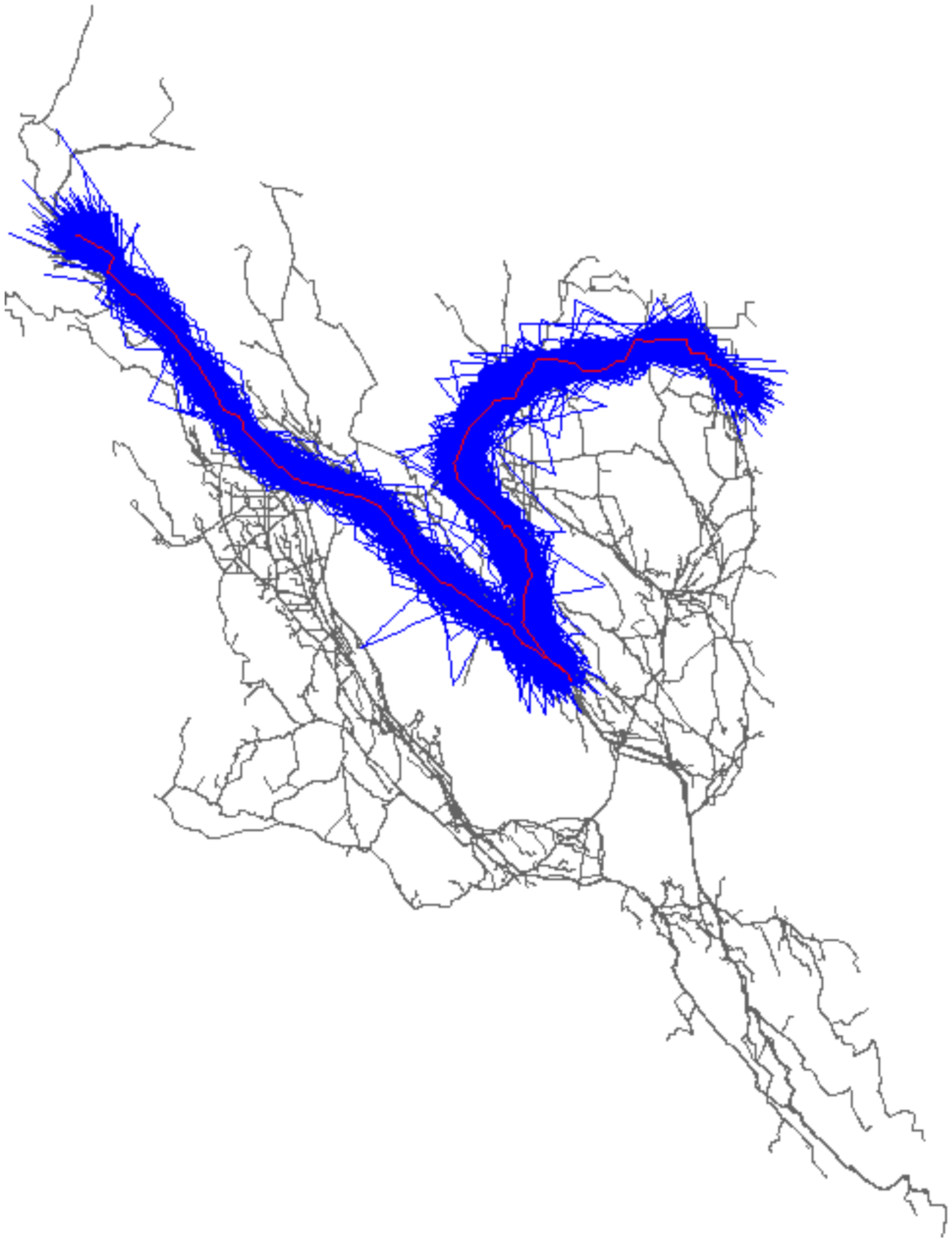


Figure 4.6: Attacking a trajectory in San Francisco when $|KT| = 30$

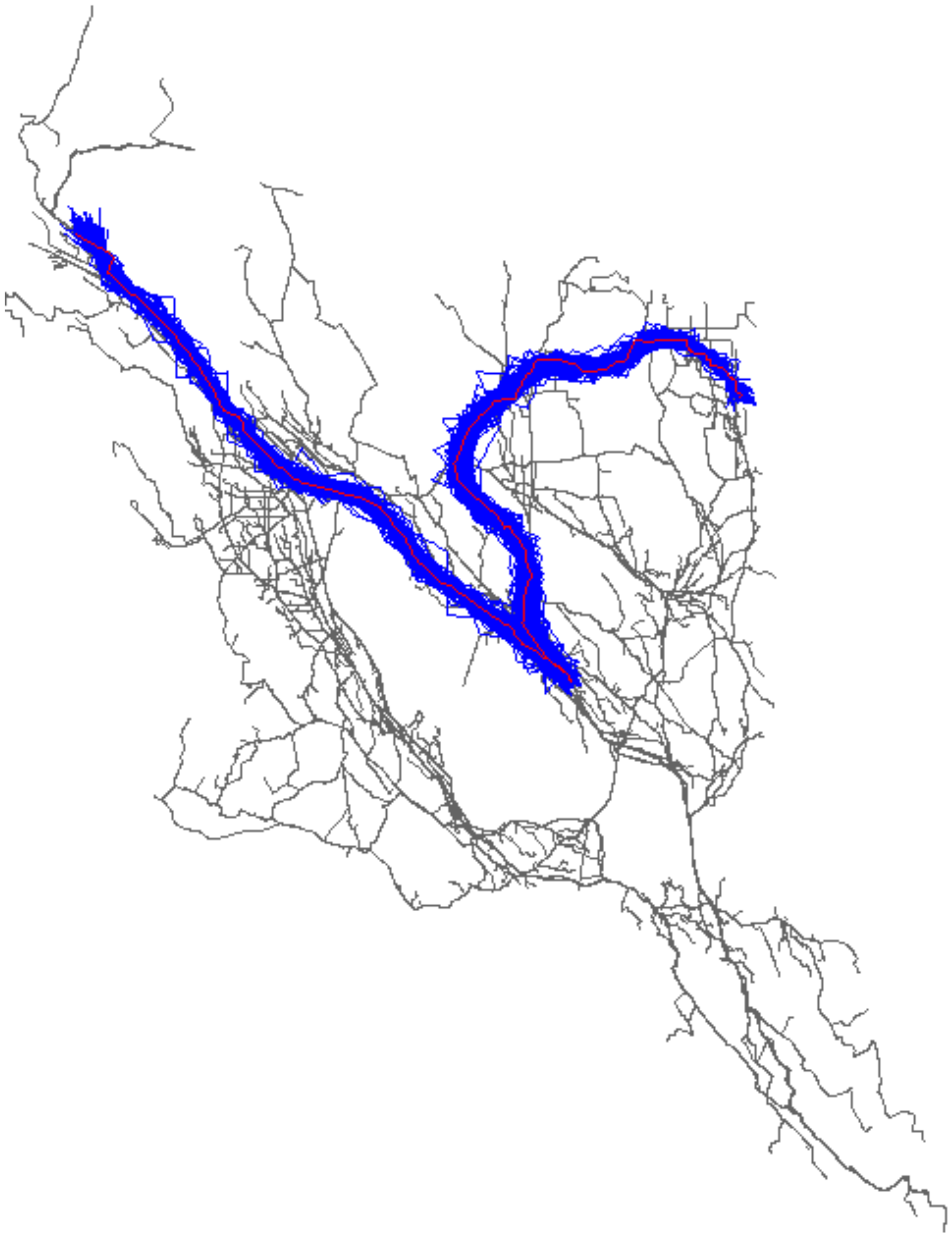


Figure 4.7: Attacking a trajectory in San Francisco when $|KT| = 50$

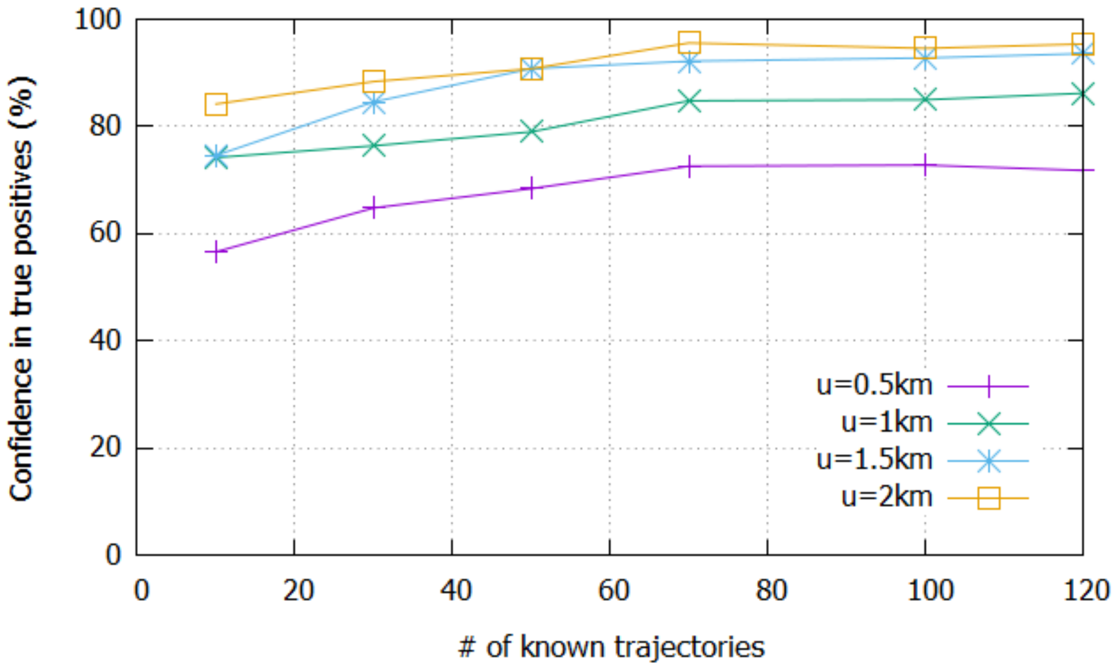


Figure 4.8: Average confidence in true positives against different number of known trajectories (Milan)

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 Figure 4.8, 4.9 and Figure 4.10, 4.11 respectively.

Analyzing Figure 4.8, we make the following observations: On a real dataset (i.e.,

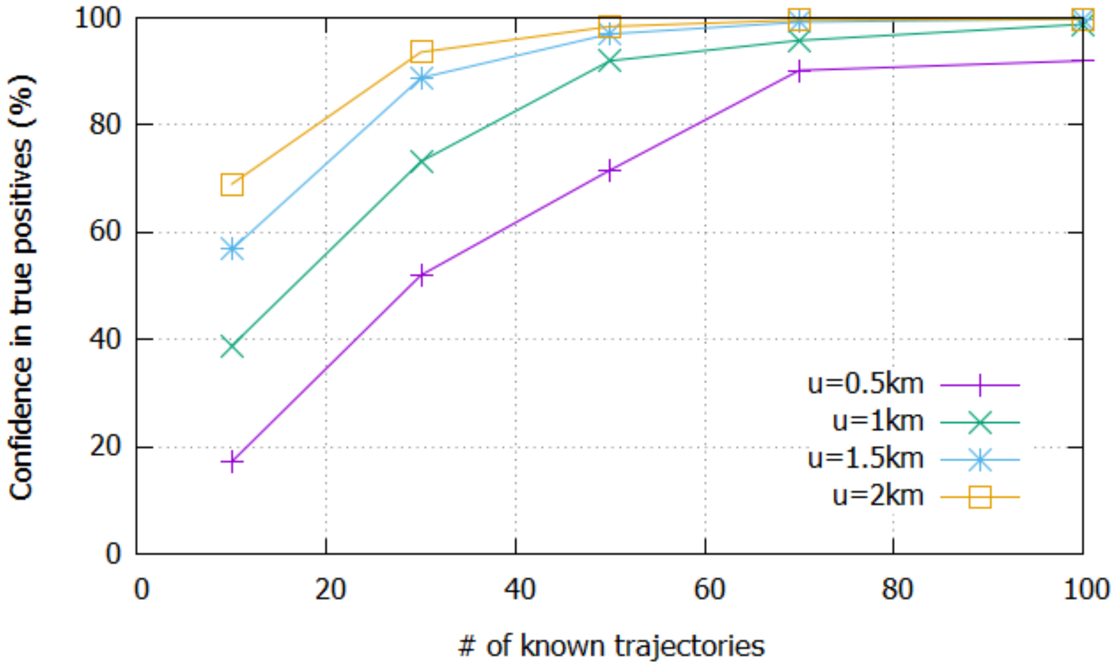


Figure 4.9: Average confidence in true positives against different number of known trajectories (San Francisco)

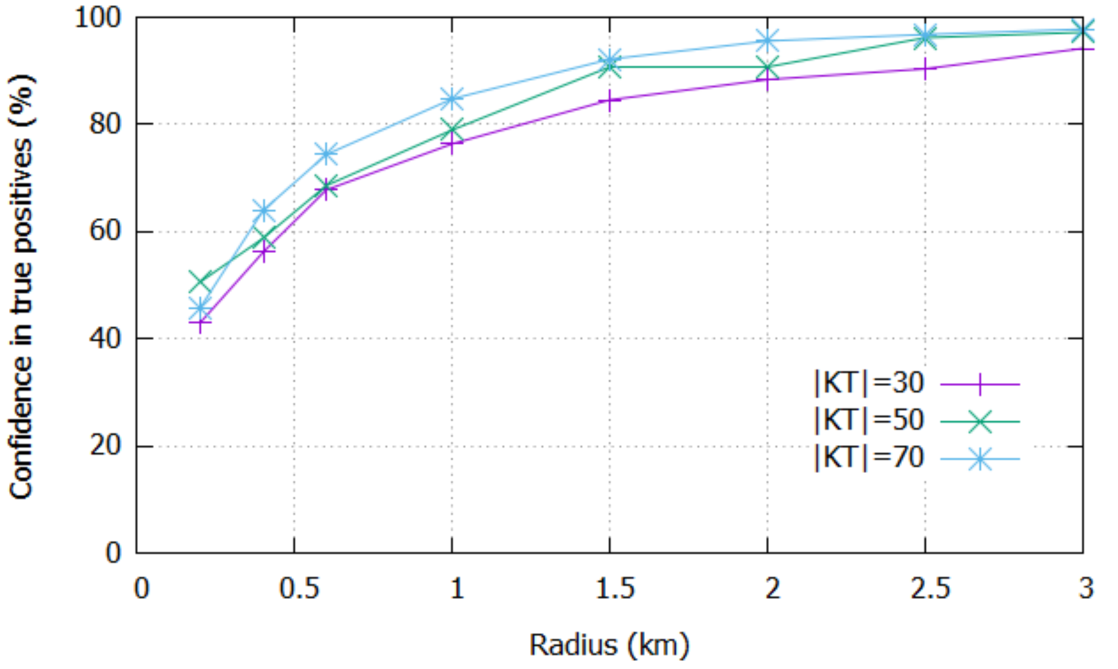


Figure 4.10: Average confidence in true positives against different radiuses (Milan)

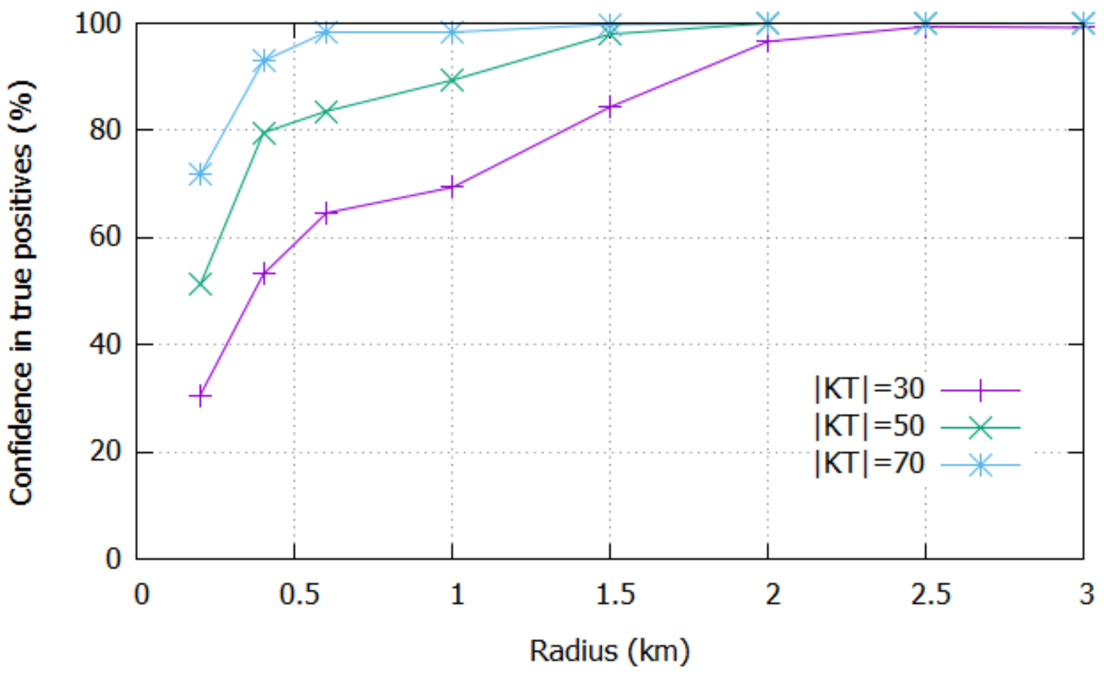


Figure 4.11: Average confidence in true positives against different radiuses (San Francisco)

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 as shown in Figure 4.9. 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 Figure 4.10 and 4.11, 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%. 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 Figure 4.10. 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., Figure 4.10 and 4.11) 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 $||p - p_i|| \approx u$; and $\forall p_j \in T^r$ such that $p_i \neq p_j$, $||p - p_j|| \geq u$. 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 Figure 4.12 and 4.13, we graph the average confidence in false positives with re-

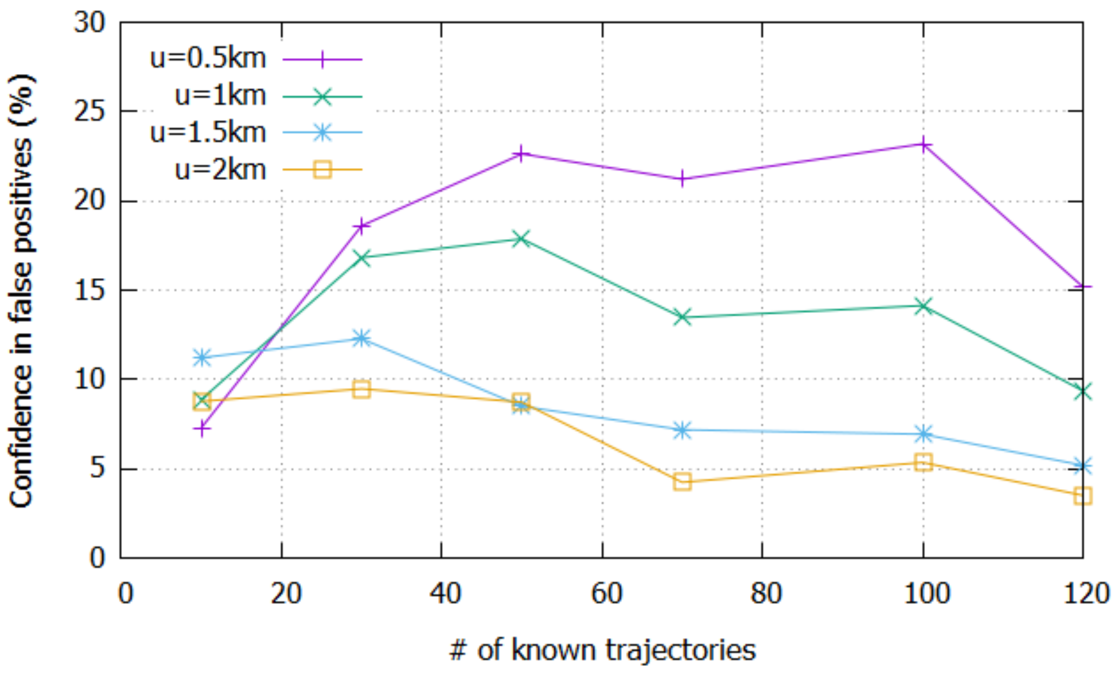


Figure 4.12: Average confidence in false positives (Milan)

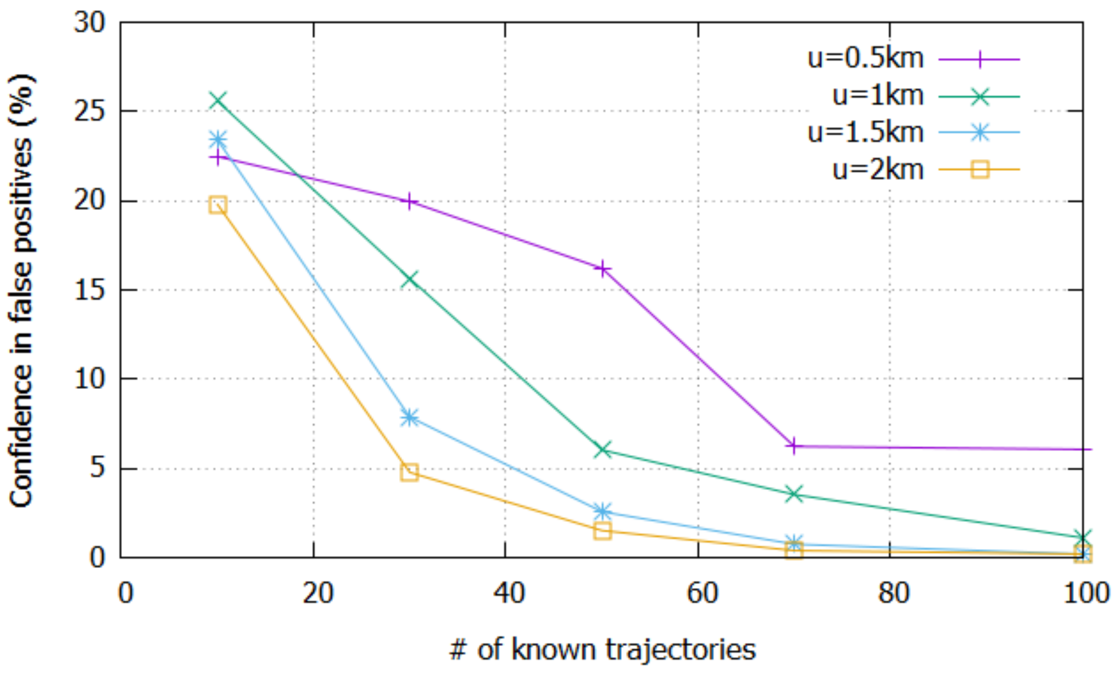


Figure 4.13: Average confidence in false positives (San Francisco)

spect 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 Figure 4.12 and 4.13 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 Figure 4.12, 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. Observe that in Figure 4.2 the algorithm can create only a few candidates. In Figure 4.3, candidates are more scattered, and in Figure 4.4, candidates are condensed again. Whereas in Figure 4.5, candidates get more and more condensed as we move from 4.5 to 4.6 and 4.7.

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 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 Figure 4.2, 4.3, 4.4 and Figure 4.5, 4.6, 4.7 that the $\text{conf}_{p,u}$ of a location far away from the victim is

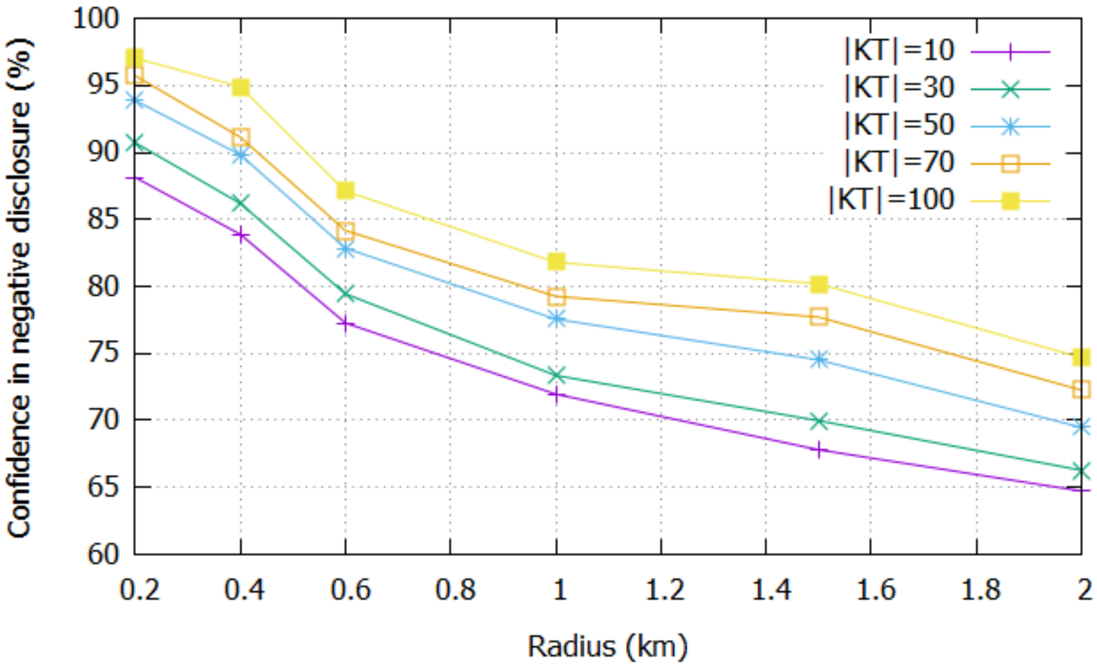


Figure 4.14: Average confidence in negative disclosure (Milan)

zero, or almost zero. For example, consider a location on the bottom-right corner of the map of Milan (Figure 4.2, 4.3, 4.4). 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 Figure 4.14 and 4.15. 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 Figure 4.2, 4.3, 4.4 and Figure 4.5, 4.6, 4.7. 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

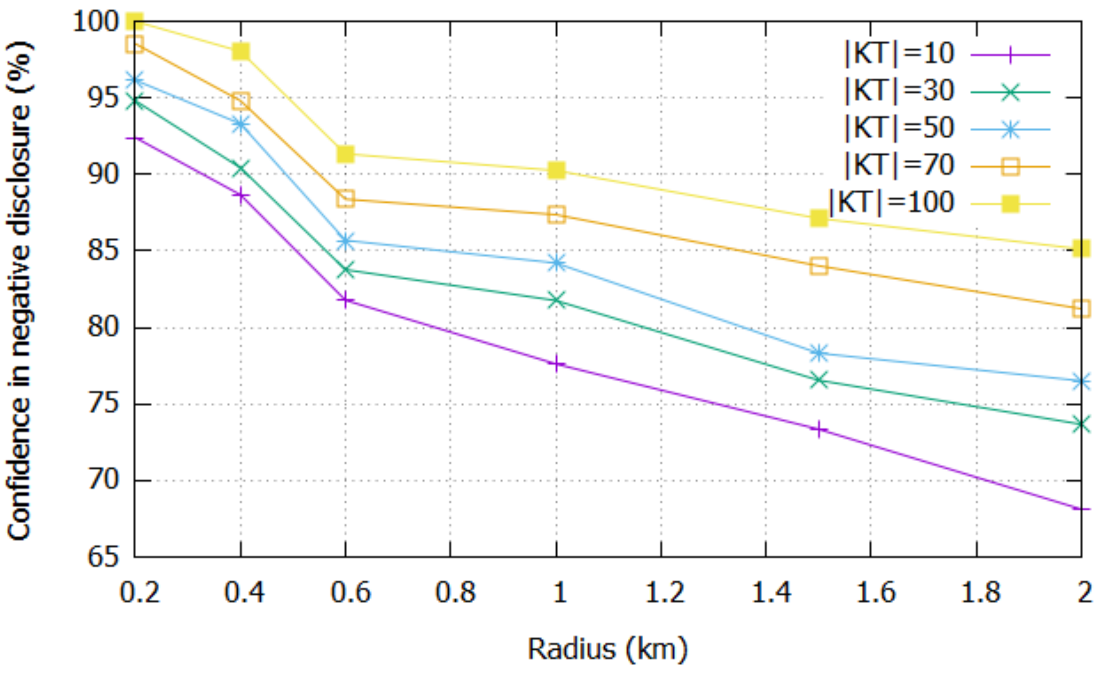


Figure 4.15: Average confidence in negative disclosure (San Francisco)

$\text{conf}_{p,u}$ 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.

4.3.3 Comparison with Previous Work

We compare our attack with the previous work of [29]. The goal of the previous work is to build a candidate trajectory that best resembles a target trajectory T^r, given adversarial background information. The authors employ a heuristic-based algorithm that randomly forms a candidate trajectory first, and then tries to converge this trajectory to T^r in an iterative manner. The heuristic is based on finding a location that is closest to T^r with a matching time-stamp, and then building the rest of the trajectory in guidance of that location.

Table 4.1: Comparison with previous work

# of Known Traj.	SR of Previous Work	SR of Our Method	Improvement %
10	0.4287	0.4567	6.53
30	0.6421	0.7391	15.11
50	0.7783	0.8280	6.39
70	0.8217	0.8511	3.58
100	0.8635	0.8942	3.56

Since the aim of [29] is building one trajectory best resembling T^r, there is no notion of keeping a set of candidate trajectories. Therefore the evaluation metrics used in this work are not applicable to [29]'s setting. On the other hand, [29] uses its own metric, *Success Rate (SR)*. To measure the success rate of a trajectory T^* in resembling T^r, i.e., $SR(T^*|T^r)$, the authors first find the *Average Sample Distance (ASD)* between T^* and T^r. The ASD is simply the Euclidean distance between T^* and T^r divided by the size of the trajectories.

$$ASD(T^*, T^r) = \frac{(T^* - T^r)_2}{|T^r|}$$

The authors then observe that ASD is dependent on the magnitude of the locations/coordinates in the trajectories. Therefore they compute the magnitude of T^r as follows:

$$MAG(T^r) = \sum_{i=1}^{n-1} ||p_i - p_{i+1}||$$

Finally, the success rate is defined as:

$$SR(T^*|T^r) = e^{-\alpha * \frac{ASD(T^*, T^r)}{MAG(T^r)}}$$

where α is a sensitivity factor which decides how steeply SR goes to 1 as the candidate approaches the target. They show that $SR(T^r|T^r) = 1$, SR tends to 0 as ASD tends to infinity (since $e^{-\infty} = 0$), $SR \in [0, 1]$, and that SR has other desirable properties.

We implemented the attack algorithm in [29] and their metric SR. We compare our work with theirs as follows: For the candidates CT we generate, we measure their SR

and compare this with the best SR obtained from [29]. We used $\alpha = 20$ to mimic their setting. The results are given in Table 4.1. We see that our work outperforms [29] in all experiments. Although the improvement with large KT is not that significant (e.g., only 3.5%) for smaller and medium-amount of background knowledge (e.g., $|KT|$ is 10, 30 or 50) our improvement is obvious (e.g., more than 15%). We emphasize that, compared to [29], we also study the positive and negative location disclosure risks. We believe that these have clearer real-life consequences and are therefore of higher impact.

Chapter 5

Location Disclosure Risks of Releasing Relation-preserving Data Transformations

5.1 Brief Summary

In this work, we are in the attacker's role exploring valuable information from transformed spatio-temporal dataset with few known points. We propose a way of breaching the privacy of relation-preserving transformations based on background knowledge in the form of a set of known input points. Those points are a bunch locations known by the attacker that are also present in the transformed dataset.

We make the following contributions:

- We generalize from distance-preserving transformation to relation-preserving transformation and attack on the relation-preserving transformation.
- The attack is based on solely on a set of known samples from the dataset (so the pairwise distances among them can be calculated) and their relations on transformed dataset.
- The attack is applicable on the perturbed data publication model.

- Our attack is computationally feasible, which has been an issue for some earlier works [29].

The data owner's private database $\mathcal{D}$ is denoted with transcript $\mathcal{D}(r_1, ..., r_n)$, where $r_i \in \mathcal{D}$ are the tuples in $\mathcal{D}$. We make no assumptions regarding the structure or type of data contained in $\mathcal{D}$, apart from the ability to map $\mathcal{D}$ to a m-dimensional Euclidean space $\mathbb{R}^m$. As such, we view each r_i as a *point* in Euclidean space, and use the terms *tuple* and *point* interchangeably in the remainder of this chapter.

The starting point of our attack is pairwise distances (or similarity) between points in Euclidean space: As the name implies, distance-preserving transformations preserve pairwise distances. Pairwise distances between elements $r_i, r_j \in \mathcal{D}$ can be computed using commonly used distance metrics, e.g., Minkowsi (p-norm) distance, Euclidean distance [29]. Without loss of generality, we assume that Euclidean distance is used for distance calculations.

5.2 Attack Algorithm

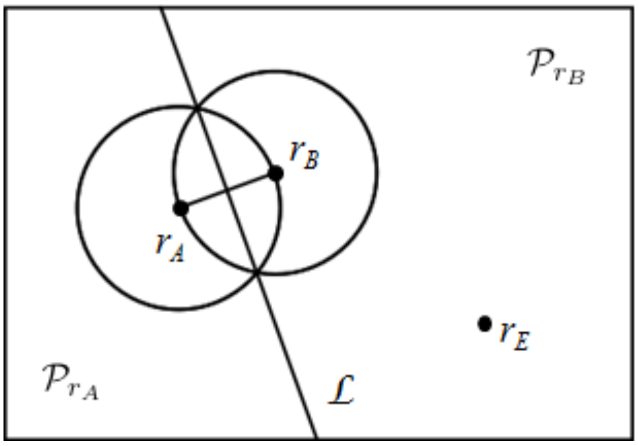
We propose a novel strategy to attack relation-preserving transformations. We assume that the following information is available to the attacker:

- **Distance matrix of the transformed data,** M'. The distance matrix can be either obtained directly as a result of distance matrix publication [31],[41] or computed by the attacker after the publication of the transformed data. For example, given the transformed database in Table 3.3a, its distance matrix in Table 3.3b can be easily computed. In this work, we consider a broader type of transformations that we call *relation-preserving transformations*. Such transformations allow the distance matrix to change, but only in a way that the relative order of the cells in the matrix is conserved. That is, assuming M is the distance matrix of the original data and M' is the distance matrix of the transformed data, if $M_{i,j}$ is greater than [less than] $M_{k,l}$, $M'_{i,j}$ must be greater than [less than] $M'_{k,l}$. Relation-preserving transformations have the desirable property that similar data mining results can be obtained although exact pairwise distances between records are not revealed. For example, a record's k nearest neighbors do not change, therefore k-NN classification on a transformed dataset would produce the same output as it would produce if run on the original dataset.

- **A set of known samples**. The attacker has a set of records $r_i \in \mathcal{D}$. That is, the attacker knows where each r_i maps to in the original $\mathbb{R}^m$ space (prior to transformation).

Known-sample attacks are popular in the literature [29, 31, 32, 44]. There are multiple ways in which an attacker can obtain a set of known samples, e.g., the attacker may know that his and a few other friends' information is in the data, or may be able to inject a tuple into the data. Notice that our attack makes no assumptions regarding the underlying transformation function used or the transformed output. That is, we do not require the attacker to obtain input-output pairs or any output points from the transformation function $\mathcal{S}$. However, we do require that the distance matrix M' of the transformed data $\mathcal{D}'$ is available. This is not a strict assumption since data owner transform data to share it for data analytics purposes.

We introduce our attack using the example in Figure 5.1. Suppose that database $\mathcal{D}$ is 2-dimensional (i.e., records are in $\mathbb{R}^2$), and let r_A, r_B in $\mathcal{D}$ be the known samples of the adversary. Say that the goal of the adversary is to find the position of r_E, i.e., locate r_E in $\mathbb{R}^2$ space. Let M' be the distance matrix that is published after a relation-preserving transformation $\mathcal{S}$ is applied to $\mathcal{D}$.



Distance matrix of $\mathcal{D}'$

	r'_A	r'_B	r'_E
r'_A	0	3	14
r'_B	3	0	12
r'_E	14	12	0

Figure 5.1: Sample 2-dimensional database $\mathcal{D}$ with three records. Actual locations of records in $\mathbb{R}^2$ (on the left) and the distance matrix published after transformation (on the right).

Observation 1. *If $M'_{A,B} < M'_{A,E}$, then in the original dataset r_E must be located outside the circle with centre r_A and radius $\delta(r_A, r_B)$.*

Proof. From the definition of distance matrices (Definition 5), $M'_{A,B} < M'_{A,E}$ implies that $\delta(\mathcal{S}(r_A), \mathcal{S}(r_B)) < \delta(\mathcal{S}(r_A), \mathcal{S}(r_E))$. Since transformation $\mathcal{S}$ is relation-preserving, the previous statement implies that $\delta(r_A, r_B) < \delta(r_A, r_E)$ must hold. The attacker knows the locations of r_A and r_B, therefore he may compute $\delta(r_A, r_B) = ||r_A - r_B||$ and draw a circle with centre r_A and radius $\delta(r_A, r_B)$. This circle forms an infinite collection of points that have the same distance to r_A, and all points X that are in the area enclosed by the circle (including points on the circle) satisfy $\delta(r_A, X) \leq \delta(r_A, r_B)$. Since $\delta(r_A, r_E) > \delta(r_A, r_B)$, r_E cannot be located within or on the circle, and hence must be located outside the circle. □

Observation 2. *If $M'_{A,B} = M'_{A,E}$, then in the original dataset r_E must be located on the circle with centre r_A and radius $\delta(r_A, r_B)$.*

Observation 3. *If $M'_{A,B} > M'_{A,E}$, then in the original dataset r_E must be located within the area enclosed by the circle with centre r_A and radius $\delta(r_A, r_B)$.*

Observations 2 and 3 follow trivially from the first observation, therefore we omit their proofs. The key idea of our attack is that given the known points r_A, r_B and the transformed distance matrix of $\mathcal{D}'$, the attacker iteratively prunes the search space (which is initially equal to the entire data space). Observations 1, 2 and 3 demonstrate one way of "pruning" the search space while the adversary searches for r_E. The main idea is to compare the distance between the two known samples (r_A and r_B) and the distance between the target (r_E) and the known samples after the transformation is applied. The relation-preservingness of the transformation allows the adversary to make inferences on the original dataset and prune out those portions in $\mathbb{R}^2$ that r_E cannot be located in. In all of the observations above, we compare $M'_{A,B}$ to $M'_{A,E}$, however the same can be compared to $M'_{B,E}$ that would result in circles centered at r_B with radius $\delta(r_A, r_B)$. The procedure can also be repeated for every pair of samples the adversary has (we considered only one pair (r_A, r_B) so far for the sake of simplicity and clarity).

We now present a second type of pruning. For the two known data samples r_A and r_B, let $\mathcal{L}$ denote the perpendicular bisector of the hypothetical line connecting r_A and r_B. (Given the locations of r_A and r_B, it is trivial to draw both the hypothetical line and its perpendicular bisector.) As seen in Figure 5.1, $\mathcal{L}$ divides the search space into two portions. Let $\mathcal{P}_{r_A}$ denote the portion that contains r_A and $\mathcal{P}_{r_B}$ denote the portion that contains r_B.

Observation 4. *If $M'_{A,E} > M'_{B,E}$ then r_E must be located in $\mathcal{P}_{r_B}$.*

Proof. $M'_{A,E} > M'_{B,E}$ implies $\delta(\mathcal{S}(r_A), \mathcal{S}(r_E)) > \delta(\mathcal{S}(r_B), \mathcal{S}(r_E))$ due to the definition of distance matrices. Since $\mathcal{S}$ is relation-preserving, this implies $\delta(r_A, r_E) > \delta(r_B, r_E)$. $\mathcal{L}$ is a line containing points that are equidistant to r_A and r_B. All points $X \in \mathcal{P}_{r_B}$ have the property $\delta(r_B, X) < \delta(r_A, X)$, whereas points Y on $\mathcal{L}$ satisfy $\delta(r_B, Y) = \delta(r_A, Y)$ and points $Z \in \mathcal{P}_{r_A}$ satisfy $\delta(r_A, Z) < \delta(r_B, Z)$. Hence, r_E is in $\mathcal{P}_B$. $\square$

Observation 5. *If $M'_{A,E} = M'_{B,E}$ then r_E must be located on $\mathcal{L}$.*

Observation 6. *If $M'_{A,E} < M'_{B,E}$ then r_E must be located in $\mathcal{P}_{r_A}$.*

The second type of observations we make examine the distance between the target r_E and the two known samples (i.e., $\delta(r_A, r_E)$ and $\delta(r_B, r_E)$). Again, this process can be repeated for every pair of known samples. At this point, we would like to note two characteristics of our attack: (1) The pruning process is fully deterministic, i.e., the adversary is 100% confident that pruned areas may not contain the target data point. (2) We are placing constraints on where r_E can/cannot be located in the original dataset, not the transformed dataset. The adversary's goal is to locate r_E in the original data space, not in the transformed space.

5.2.1 Attack Formalization

In this section we formalize our attack and generalize it to n-dimensional space $\mathbb{R}^n$, where $n \geq 2$. Let $A, B \in \mathbb{R}^3$ have the following coordinates: $A(-2, 1, 4), B(0, 6, 3)$. We say that P has the coordinates $P(x, y, z)$ and solve:

$$
\begin{gathered}
||P - A|| = ||P - B|| \\
\sqrt{(x+2)^2 + (y-1)^2 + (z-4)^2} = \sqrt{(x-0)^2 + (y-6)^2 + (z-3)^2} \\
x^2 + 4x + 4 + y^2 - 2y + 1 + z^2 - 8z + 16 = x^2 + y^2 - 12y + 36 + z^2 - 6z + 9 \\
4x + 10y - 2z = 24
\end{gathered}
$$

H_{AB} is therefore the plane given by the final equation above. Half-spaces, defined in Definition 15, are often specified using linear inequalities derived from the hyperplane that seperates them. For the previous example, the two open half-spaces of $\mathbb{R}^3$ are given by the equations:

$$
\begin{gathered}
\mathcal{P}_A : 4x + 10y - 2z < 24 \\
\mathcal{P}_B : 4x + 10y - 2z > 24
\end{gathered}
$$

It is clear to see that since H_{AB} is the hyperplane, defined in Definition 14 that is equidistant to A and B, one of the half-spaces will contain point A whereas the other will contain B. We call these half-spaces $\mathcal{P}_A$ and $\mathcal{P}_B$ respectively. Circles are hyperspheres, defined in Definition 12, in 2-dimensional space. A circle with centre $C(3, 4)$ and radius 7 would be characterized by the equation: $(x-3)^2 + (y-4)^2 = 49$. Then, the open hyperball, defined in Definition 13, $B_{(3,4),7}$ specifies the area enclosed by this circle, excluding those points that are on the circle. This is given by the equation: $(x-3)^2 + (y-4)^2 < 49$.

We present our attack strategy in Algorithm 3. We have the following inputs: The universe U is a collection of all possible points that may exist in the database. The universe often has boundaries that are dictated by the semantics of the underlying database, e.g., the boundaries of a dimension *age* could be 0 and 110. Or, if the database contains the locations or spatial information regarding people living in a particular city, the universe is bounded by the borders of that city. M' is the distance matrix that is obtained after a relation-preserving transformation. The attacker has a set of legitimate known samples (i.e., all samples are within the universe U). We describe the attack assuming the adversary would like to compromise the location of one target record r_E, but the attack can be run on an arbitrary record. We specify the identifier of the target record, E, as one of our inputs.

Initially, we say that r_E can be anywhere in the universe, by setting the search space variable (which we denote by s) to U. For every pair of known samples, we iteratively prune the search space several times using the observations made in the previous section. Here, pruning refers to deleting certain geometric objects, or areas that do not intersect with a given geometric object, from the search space. For example, pruning a hyperball off of s would delete all points in s that intersect with that hyperball. Note that some points in the hyperball might have already been deleted in previous steps, and therefore would not be present in s. For such points no further action needs to be taken. Computing the intersection between the search space s and a given geometric object/region would delete all points in s that lie outside that object/region. The final output of the attack is the region of the universe that has not been pruned, i.e., the area where r_E must be located in.

To make the algorithm easier to follow, we give the specific observations that lead to each of the pruning operations. The pruning between lines 3-12 and 13-22 both stem from Observations 1, 2 and 3. Between lines 4-6 we apply Observation 1, between lines 7-9 we apply Observation 3, and between lines 10-12 we apply Observation 2. These three steps are based on the distances between (r_A, r_B) and (r_A, r_E). We then apply the

ALGORITHM 3: Locating a target record using a distance matrix and known samples

Input : U: the data space with its boundaries,
M': distance matrix of transformed data,
$K = \{r_1, .., r_n | r_i \in U\}$: set of known samples,
E: an identifier to denote the target record r_E

Output: $U' \subseteq U$: portion of the universe where the target record is located

```
$s \leftarrow U$
for each pair $(r_A, r_B) \in K$ do
    Build the hypersphere $S_{r_A,\delta(r_A,r_B)}$ and open hyperball $B_{r_A,\delta(r_A,r_B)}$
    if $M'_{A,B} < M'_{A,E}$ then
        $s \leftarrow s - (B_{r_A,\delta(r_A,r_B)} \cup S_{r_A,\delta(r_A,r_B)})$
    end
    else if $M'_{A,B} > M'_{A,E}$ then
        $s \leftarrow s \cap B_{r_A,\delta(r_A,r_B)}$
    end
    else
        $s \leftarrow s \cap S_{r_A,\delta(r_A,r_B)}$
    end
    Build the hypersphere $S_{r_B,\delta(r_A,r_B)}$ and open hyperball $B_{r_B,\delta(r_A,r_B)}$
    if $M'_{A,B} < M'_{B,E}$ then
        $s \leftarrow s - (B_{r_B,\delta(r_A,r_B)} \cup S_{r_B,\delta(r_A,r_B)})$
    end
    else if $M'_{A,B} > M'_{B,E}$ then
        $s \leftarrow s \cap B_{r_B,\delta(r_A,r_B)}$
    end
    else
        $s \leftarrow s \cap S_{r_B,\delta(r_A,r_B)}$
    end
    Build the equidistant hyperplane $H_{r_A r_B}$ and resulting open half-spaces $\mathcal{P}_{r_A}$, $\mathcal{P}_{r_B}$
    if $M'_{A,E} > M'_{B,E}$ then
        $s \leftarrow s \cap \mathcal{P}_{r_B}$
    end
    else if $M'_{A,E} < M'_{B,E}$ then
        $s \leftarrow s \cap \mathcal{P}_{r_A}$
    end
    else
        $s \leftarrow s \cap H_{r_A r_B}$
    end
end
return $s$
```

same approach to the distances between (r_A, r_B) and (r_B, r_E) to obtain the three steps between lines 13-22: Between lines 14-16 we apply Observation 1, between lines 17-19 we apply Observation 3, and between lines 20-22 we apply Observation 2. Afterwards, between lines 23-32, we apply Observations 4, 5 and 6. Specifically, between lines 24-26 we apply Observation 4, between lines 27-29 we apply Observation 6, and between lines 30-32 we apply Observation 5. On line 34, we return the final result after all pruning.

5.2.2 Implementation and Noise Resilience

As can be seen in Algorithm 3, our attack involves many union, intersection and difference operations. These are non-trivial to implement in continuous n-dimensional space. For the sake of reproducibility, we comment on the specifics of our implementation.

We implement the attack by discretizing the search space: We assume that the universe U is made up of uniform n-dimensional cells. Smaller the cells are, higher the total number of cells and finer the granularity of the attack will be. However, due to the increased number of cells, execution time will also be higher. We take a defensive approach when we prune cells, that is, in each pruning decision we prune only those cells that can be *completely* pruned off the search space. For example, consider Figure 5.2. When we prune the half-space $\mathcal{P}_{r_A}$, we only prune those cells that are completely contained by $\mathcal{P}_{r_A}$. For all borderline cells (e.g., those that lie on $\mathcal{L}$) we keep them instead of pruning them. As such, we guarantee that we never over-prune, e.g., if in Figure 5.2 we prune *cell Y* then we would have over-pruned by removing the portion that is to the right of $\mathcal{L}$, which includes area that r_E could actually be located in. This would violate the correctness of our attack. On the other hand, by not pruning *cell Y* we also keep the portion in *cell Y* that is to the left of $\mathcal{L}$, which we are certain that r_E is not located in. We ideally would like to prune the latter portions off, but since our cells were too coarse (i.e., too big) in this case, we could not do so. On the other hand, we safely prune *cell X* since the whole cell lies within $\mathcal{P}_{r_A}$.

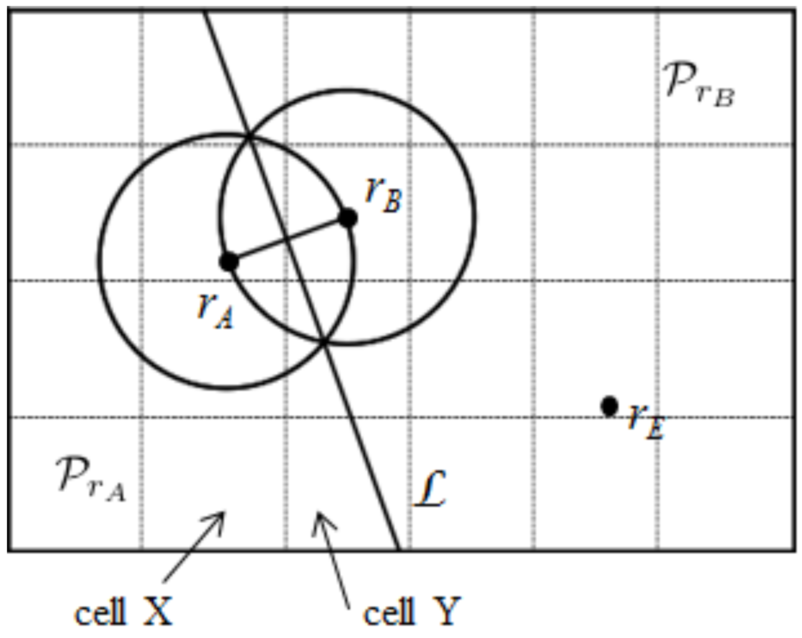


Figure 5.2: Discretization of the universe using uniform 2-dimensional cells.

Next, we comment on the noise resilience of the attack. As noted in Section 2.1, one of the prominent techniques in data privacy is *additive perturbation*, which adds noise to the published information. Thus, it is interesting to make our attack resilient to the addition of noise. Note that noise addition will likely destroy the relation-preservingness of a transformation, and the correctness of our attack can no longer be guaranteed. That is, given data space U the attack may output that r_E resides in space U', but in fact r_E resides in $U - U'$.

Algorithm 3 is not resilient to noise, since it prunes a region of the search space immediately when one pair decides that that region should be pruned. For instance, let (r_C, r_D) be a pair that decides to prune cell X when searching for target record r_E. Algorithm 3 would immediately prune X from the search space variable s and proceed. This is not problematic in the non-noisy case. However, in the noisy case, $M'_{C,E}$, $M'_{D,E}$ or $M'_{C,D}$ may be inaccurate due to the added noise. Thus, (r_C, r_D)'s decision to prune X could be wrong.

To account for these cases, we implemented a *voting mechanism*. For each cell, we keep track of the pairs of points that have *voted* in favor of pruning that cell. If, at any point, the number of votes for pruning cell X exceeds an input voting threshold, we prune X off the search space. The voting threshold depends simply on the number of known points. That is, let t denote the voting threshold. If $t\%$ of the point pairs vote in favor of pruning X, then X will be pruned. We use the voting mechanism only when the data is noisy (i.e., the transformation is not guaranteed to be relation-preserving).

Having presented our implementation details, we can now discuss metrics for measuring the *accuracy* and *success rate* of our attack. Following the inputs and outputs of Algorithm 3, we use the following transcript to denote the attack: $\mathcal{A}(U, M', K, E) = U'$, where $\mathcal{A}$ denotes the attack, (U, M', K, E) denote the four parameters our attack takes as inputs (as described in Algorithm 3), and U' is the output, i.e., the portion of the search space that the attack claims the target record r_E is located in. We quantify the accuracy of our attack as follows:

$$\text{Accuracy} = Pr(r_E \in U' | \mathcal{A}(U, M', K, E) = U')$$

The probability boils down to the ratio of trials where the attack was right in predicting that r_E was located in portion U' divided by the total number of trials. We underline that the accuracy of our attack is always 100% when there is no added noise, i.e., the data transformation is relation-preserving.

Our second metric is *success rate*, which is essentially the precision of our attack. Given that U is the universe, an attack that simply outputs $\mathcal{A}(U, M', K, E) = U$ without doing anything achieves 100% accuracy, but it cannot be considered successful since it is very imprecise. The success of the attack lies in its ability to identify a *small* portion $U' \subseteq U$ that r_E is located in. This is captured by the *success rate* metric we define below. Let $vol(.)$ denote the volume of a given n-dimensional region. Given $\mathcal{A}(U, M', K, E) = U'$:

$$\text{Success Rate} = \frac{vol(U')}{vol(U)}$$

In a uniform-cell based implementation, the success rate can be calculated simply by dividing the number of unpruned cells (i.e., those in U') by the total number of cells in the universe U.

5.3 Experiments and Evaluations

5.3.1 Experiment Setup

We implemented our attack in Java and ran experiments on the relation-preserving transformations of two datasets. We ran various experiments by changing the number of known samples and the known samples themselves. In each experiment setting, we attacked multiple target records, and we report the average results.

Consider the following practical application of our attack: The data owner runs a mobile service and collects private location check-ins of users. This data is shared with third parties after a relation-preserving transformation. Since the attacker and a few close friends of his are users of the mobile service, the attacker knows their check-in locations and this constitutes his set of known points. Then, the goal of the attacker is to infer the locations of remaining users (whose locations he cannot directly observe, since, e.g., they are not in his social circle).

Motivated by this scenario, we use 2D location data in our experiments. The datasets are explained below in detail.

Gowalla dataset [2]. Gowalla was a location-based social networking website where users shared their locations by checking-in. A total of $6,442,890$ check-ins over the period of February 2009 and October 2010 were collected and made available[1]. This dataset was recently used in other privacy related works, e.g., [77],[78]. From the Gowalla data, we extracted those check-ins made in New York.

Istanbul dataset. We collected location data of $60,000$ vehicles in Istanbul, Turkey using a vehicle tracking system. From this data, we extracted the last known locations of 200 randomly chosen vehicles and formed our experimental dataset. The data consists of the vehicle ids, latitude and longitude coordinates.

[1] `http://snap.stanford.edu/data/loc-gowalla.html`

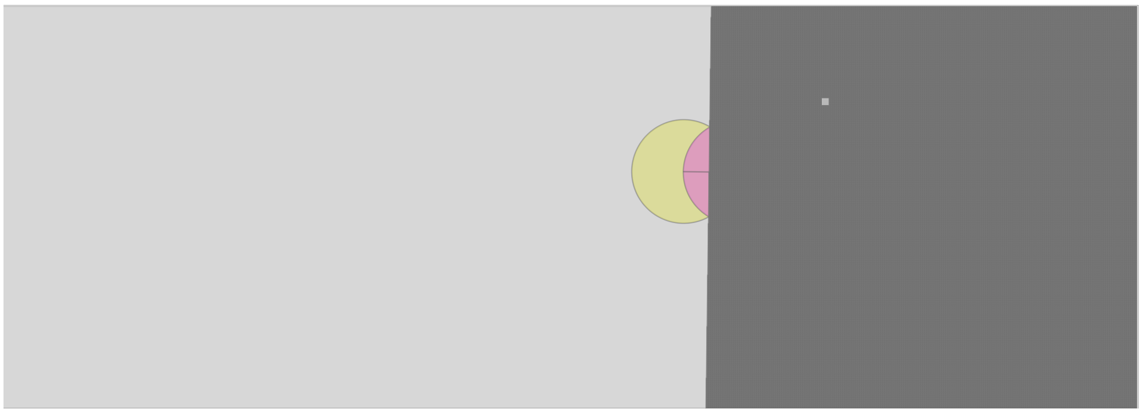


Figure 5.3: Attacking a target with knowns=2 (half of the space is pruned)

We use our grid-based implementation explained in Section 5.2.2. We pick the grid cell size such that the area of each cell is $0.01km^2$. We believe that locating an individual within such a small area is reasonable and challenging considering the entire search space is a big city. Given this cell size, we have a total of approximately 1.1millions cells in the Istanbul dataset and 240k cells in the Gowalla dataset.

5.3.2 Results and Evaluations

The most interesting aspect of our attack is how success rate changes with respect to the number of known samples. With this, we can directly infer how many samples would be needed to locate a private target record. We run this experiment in two scenarios: *noisy* and *non-noisy*. In the *non-noisy* scenario, a distance-preserving transformation is applied to the data and the attacker observes the resulting distance matrix M'. Since the transformation is distance-preserving, $M' = M$, where M is the distance matrix of the original data. In the *noisy* scenario, random noise is added to M' after the transformation. That is, the attacker observes: $M'' = M' +$ noise. In Section 2.1 we noted previous work on additive noise. Parallel with this previous work, we choose to add Gaussian noise to M', with mean 0 and variance scaled to the variance of the actual data. We calculated that the average noise is 16% in the noisy experiments.

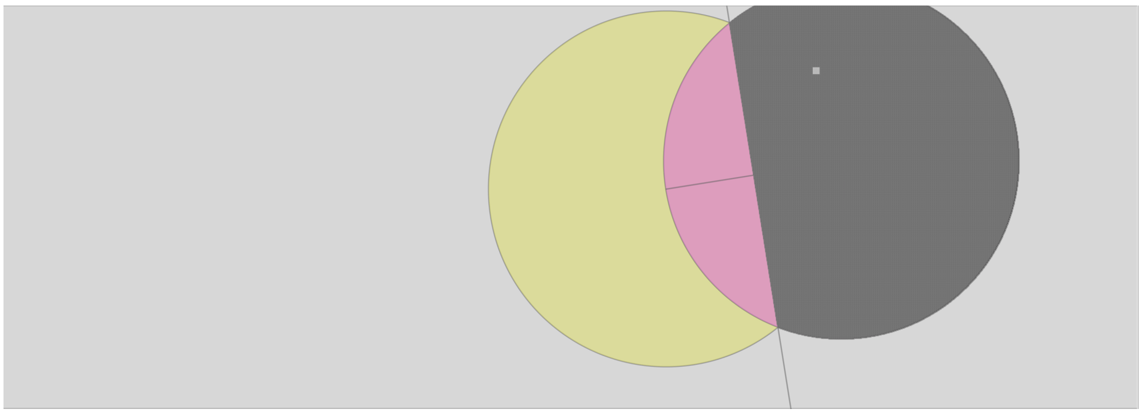


Figure 5.4: Attacking a target with knowns=2 (target lies in the perimeter of a known sample)

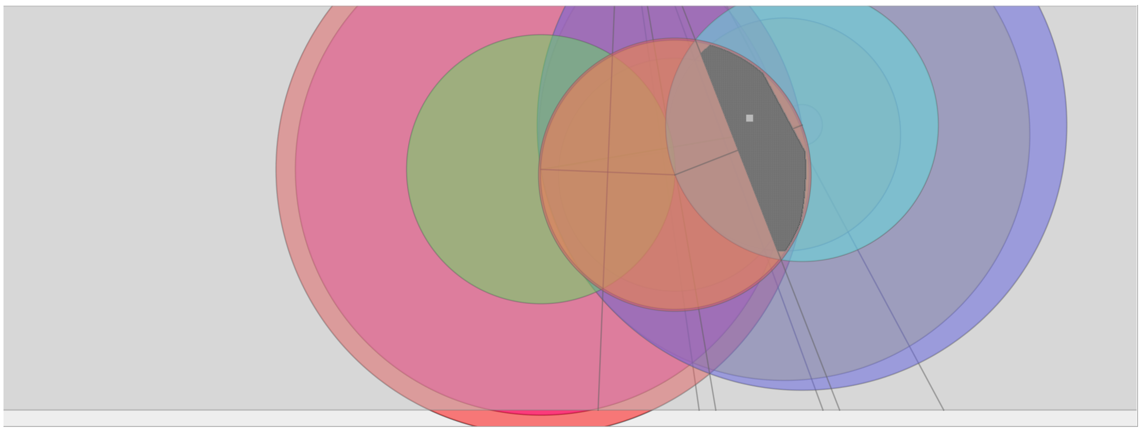


Figure 5.5: Attacking a target with knowns=4

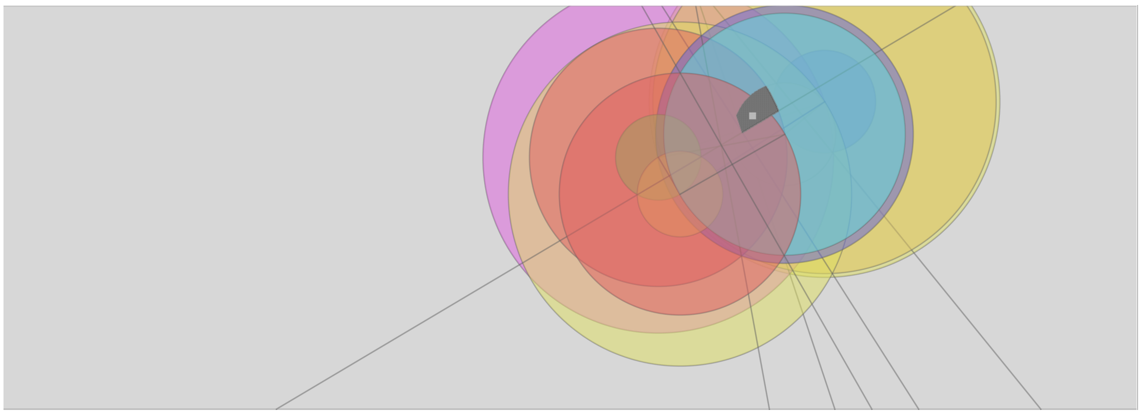


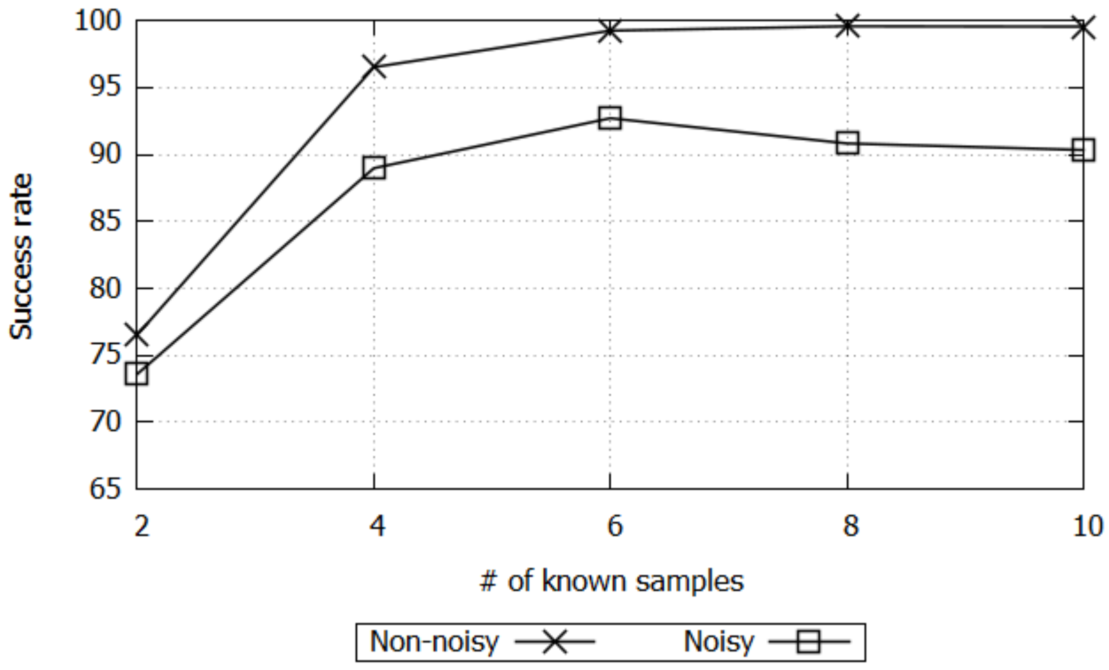
Figure 5.6: Attacking a target with knowns=4 (very small unpruned region)

We graph our results in Figure 5.7. In the non-noisy case, the attack achieves success rates of above 96% with 4 or more samples. Even with only 2 known samples, the attack achieves 69% and 76% success rates on the Istanbul and Gowalla datasets respectively. Considering that it is easy to obtain a sample of 2-4 records (e.g., check-ins of the attacker himself and 1-3 acquaintances) we can deduce that our attack is very feasible in practice. Thus, relation-preserving transformations are not robust against our known-sample attack. We present the outputs of the attacks with 2 and 4 knowns in Figure 5.3, 5.4 and Figure 5.5, 5.6 respectively. Note that dark gray color denotes unpruned regions. In Figure 5.3, hyperplane pruning dividing the search space is shown. In Figure 5.4, hypersphere pruning is shown. In Figure 5.5 and 5.6, we show that all 2-combinations of 4 known points pruned the universe to locate the target point which in fact yield better results.

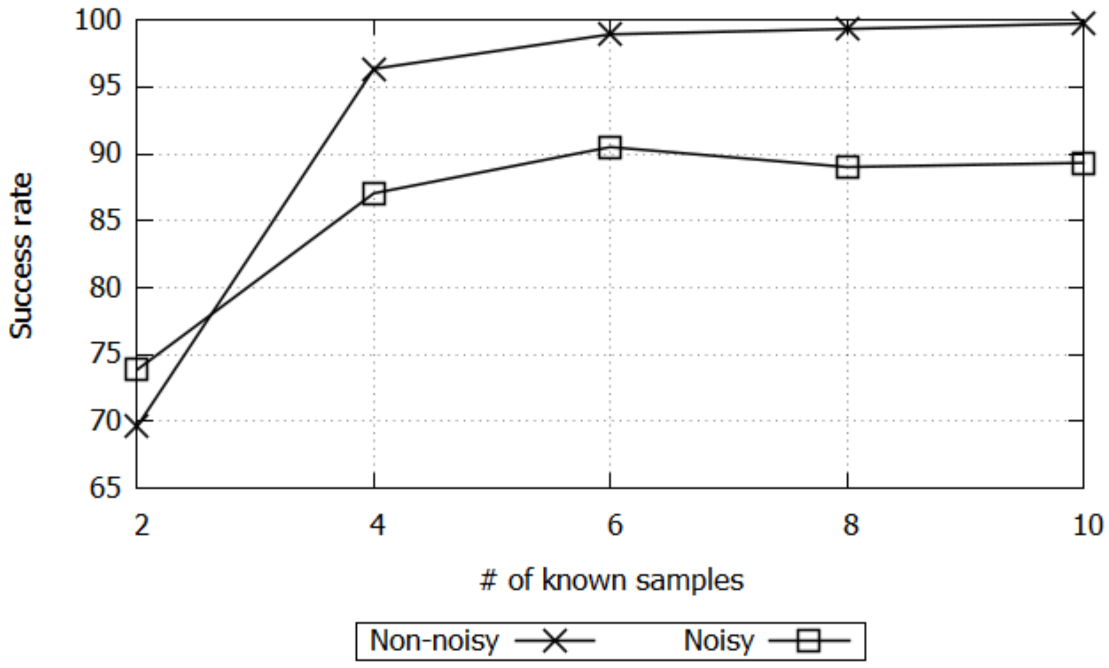
Using Figure 5.7, we also compare the success rate between noisy and non-noisy scenarios. For the noisy scenario, we only consider the success rate of those records that are correctly located, i.e., $r_E \in U'$ given that $\mathcal{A}(U, M', K, E) = U'$. Compared to the non-noisy scenario, in the noisy scenario success rate drops only by 10% in the Istanbul dataset and 8% in the Gowalla dataset. With these numbers, it is reasonable to conclude that the attack is resilient to additive white noise.

Next, in Figure 5.8, we show the accuracy of our attack with respect to the number of known samples. Interestingly, accuracy drops as we have more samples. This is because we prune for every pair of samples (see Algorithm 3) and more samples imply more pruning iterations. Since the transformed data has noise, it is more likely to hit noisy data with more pruning iterations. Then it becomes more likely to prune some regions that actually contain the target record which causes accuracy loss. This gives a clear trade-off between success rate and accuracy: If we are more aggressive by pruning many regions, we increase our success rate by definition. However, at the same time, it becomes more likely that the region containing the target record will also be pruned if we are too aggressive. This would decrease our accuracy.

Bearing in mind both Figure 5.7 and Figure 5.8, we describe some guidelines for the attacker. The goal of the attacker is to first remain accurate in his prediction, and second to



a: Gowalla dataset



b: Istanbul dataset

Figure 5.7: Success rate (in percentage) in noisy and non-noisy scenarios

make his prediction very precise by having a high success rate. Note that, if the attacker very precisely locates his target within 0.01% of the search space but the target is not actually located there, this is potentially a bigger problem than having a somewhat poorer precision, e.g., 4%, but remaining accurate in the prediction. The first factor is concerned with accuracy, while the second is concerned with success rate. As such, it is best for the attacker to choose a setting that maximizes both, but accuracy should not be sacrificed in favor of success rate - while vice versa is considerable.

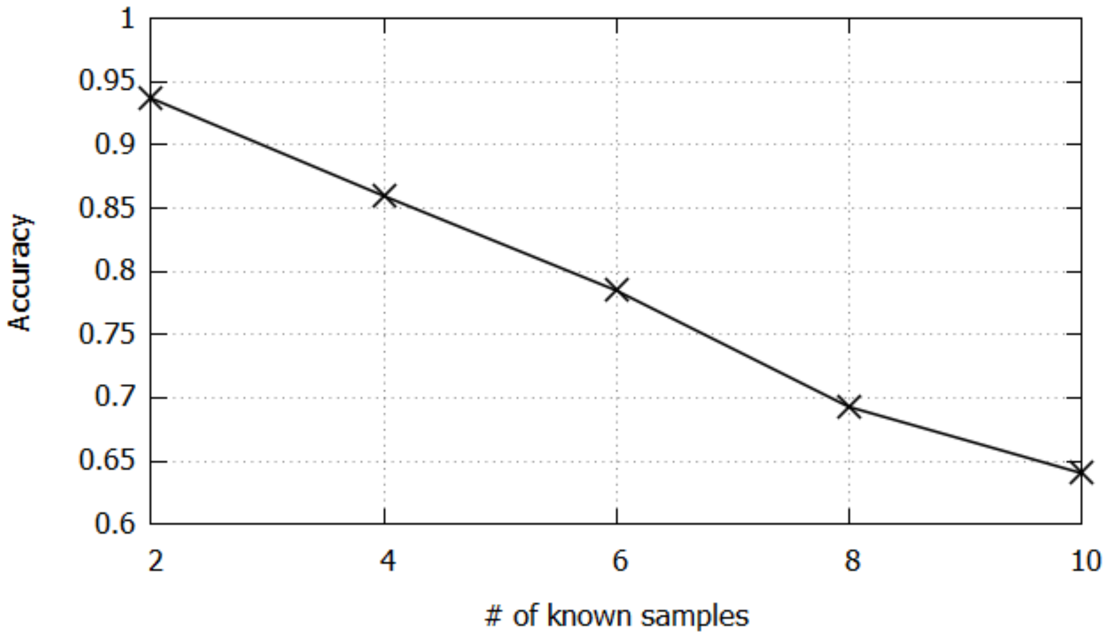
For example, assume that the attacker is operating on the Gowalla dataset and wishes to remain 85% accurate in his predictions. Then, by Figure 5.8, the attacker needs 4 or less samples. The expected success rate is between 73% and 88%, according to Figure 5.7.

In general, there are two factors affecting accuracy:

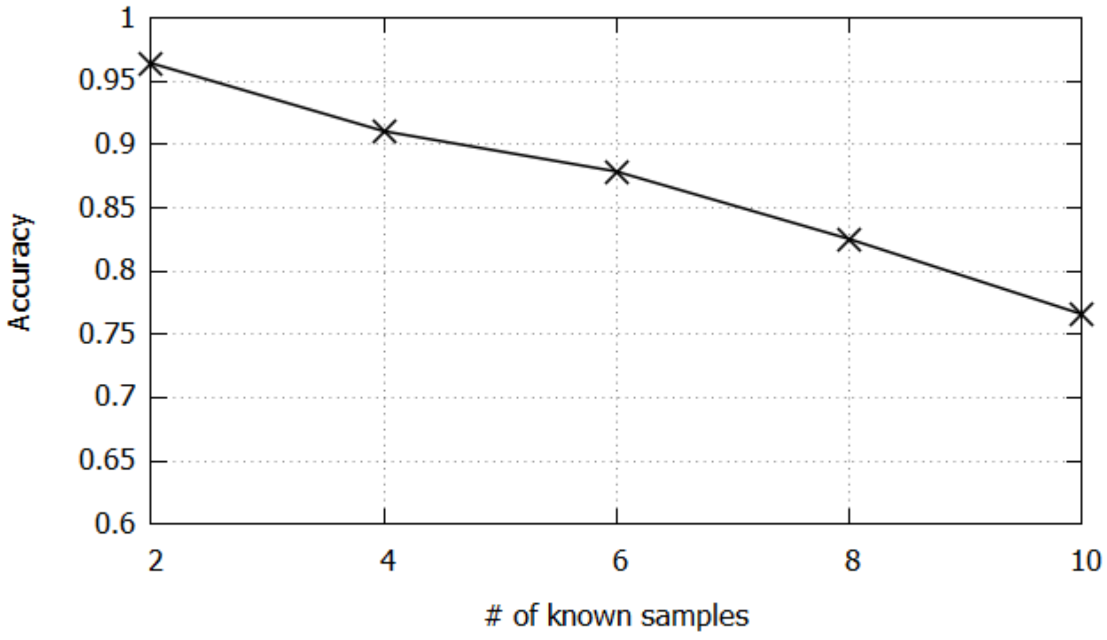
1. number of samples
2. the voting threshold

We discussed the number of samples above. The voting threshold was explained in Section 5.2.2. The attacker can set the voting threshold so high that the probability of erroneous pruning would be very small. We note that a third factor affecting accuracy would be the amount of noise added to the transformed data. If the noise is high, then the data is very distorted, and hence the accuracy would eventually drop. However, the quantity of noise is beyond the control (and usually beyond the knowledge) of the attacker. Therefore, we do not propose guidelines based on the amount of noise.

To better understand the factor of the voting threshold, we conduct the experiment in Figure 5.9. Here we show the effect of the voting threshold for different number of samples. We make the following observations: First, as the voting threshold increases, accuracy increases. This is because a higher voting threshold implies that more point pairs need to agree in order to prune a region, and therefore a single noisy entry is less likely to cause an erroneous pruning. Second, as we have more samples, we should have higher voting thresholds in order to remain accurate. This is in parallel with the results and discussion of Figure 5.8. More samples usually cause accuracy to drop, and the



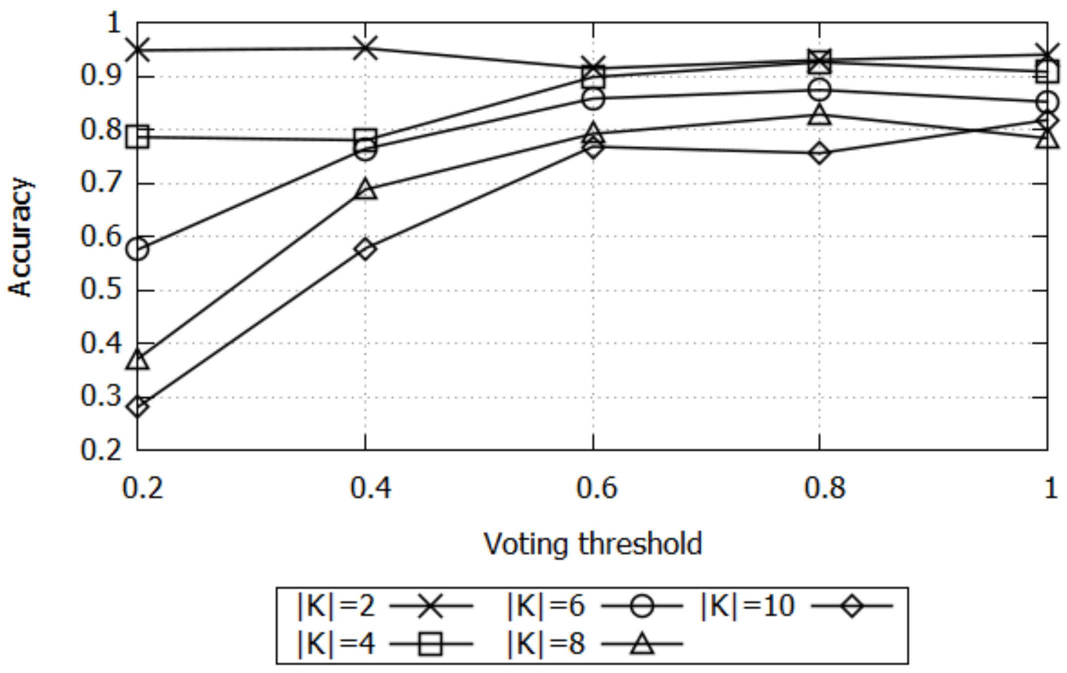
a: Gowalla dataset



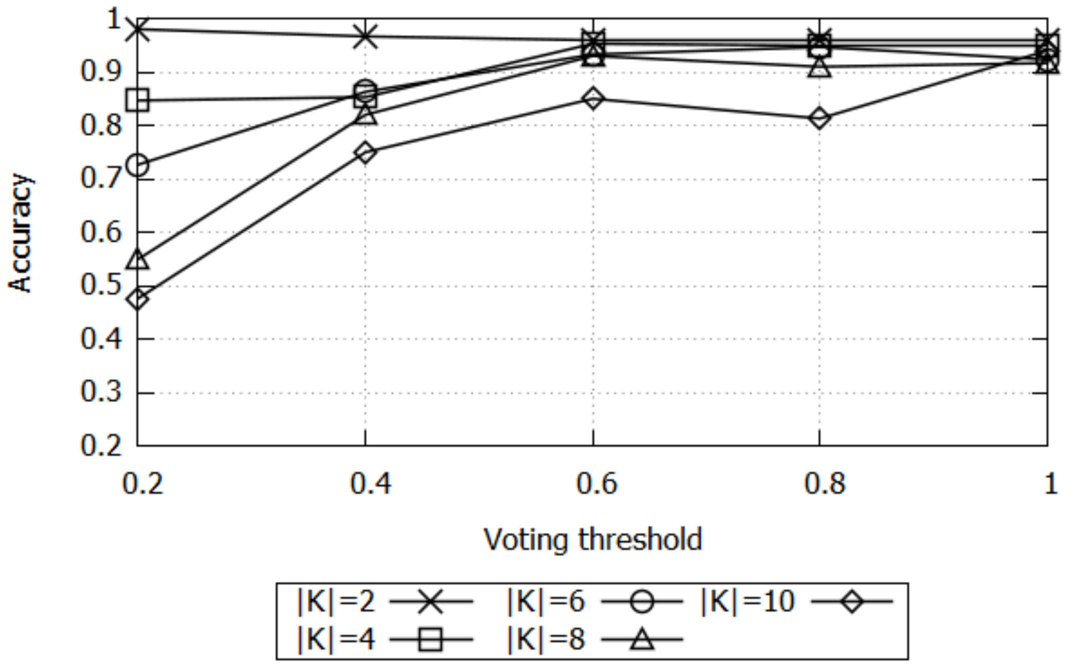
b: Istanbul dataset

Figure 5.8: Accuracy in the noisy scenario

voting threshold should be increased to compensate that. Finally, we observe that for both datasets, voting thresholds below 0.4 cause very inaccurate results, and a threshold of at least 0.6 is needed to remain roughly 80% accurate.



a: Gowalla dataset



b: Istanbul dataset

Figure 5.9: Effects of the voting threshold on accuracy

Chapter 6

Conclusions and Future Work

In this thesis, we studied the privacy risks of spatio-temporal data transformations. We highlight that the distance preserving data transformation techniques are prone to privacy breaches. We developed two different attack methods exploring the potential of a common attack scenario with different input-output settings and the context. Our work focus on both location and trajectory cases. A few tuples from the dataset itself is what the attacker actually needs as background information. Then, the attacker can predict the target data of individuals or limit the possible region as small as possible. The output of such attacks can easily answer the questions like ”is the target passed through a specific location?” or ”has the target been around a specific location?”. In both works, we explore through attacker's perspective and assume that the attacker can reach reasonable number of input data that are highly likely to be his own data or accessible data. The attacker has also access to the public properties of the dataset. He has access to released pairwise distances and relations. Both of the works discuss attacks on transformed datasets of location information without observing the transformed data itself. Only a few points from the dataset compared to the size of the dataset are sufficient to conduct the attack.

In our first work detailed in Chapter 4, we present an attack method for discovering, with significant confidence, if an unknown private trajectory passes (or does not pass) through the region of interest. The region 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 trajectories (i.e., KT) and their pairwise distances to the target trajectory. It works by generating a set of candidate trajectories that resemble the target 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 on encryption but are not resilient to known-sample attacks, will make the attacks like ours much more attainable. Naively assuming that these services are safe may lead to serious disclosure risks, as this work 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 directions for future work. We also plan to study distance metrics other than Euclidean distance, to see whether the attack is applicable to the settings other than the trajectory data. In addition, we point out that the adversaries that 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 more limited.

In our study on location disclosure of releasing relation preserving data transformations, discussed in Chapter 5, we show the privacy risks of individual location points. The attacker does not need to observe the transformed dataset but the relations are sufficient. The attacker can then infer the probable locations of the target using his limited knowledge from the dataset. A few known points from the dataset in the range of $[2 - 10]$ is sufficient to carry out a successful attack. The success of the attack, as it depends on how small the probable region, is varies in between 75% and 100%. We also examine the noisy data set as if the data owner publishes data with deliberate noise in order to prevent attacks. In this case, the attacker can perform the attack with less success rate in the range of 70% - 90% when the noise is on average 16%. In the noisy case we also discuss the accuracy of the attack with respect to the number of known data points. We show that the accuracy tends to drop as the number of known points increases because as the known points increase it is high chance to hit a noisy data from the relations. To refine the attack, we propose a voting mechanism which takes the average accuracy of the attacks up to 80%. The goal of this work is to attack relation-preserving transformations that also allow distance matrices to change slightly. In comparison, attacks on distance-preserving transformations assume no changes to distance matrices. Therefore attacks on relation-preserving transformations are applicable to distance-preserving transformations (whereas the literature mostly focuses on attacks on distance-preserving transformations).

We intend to extend our second study to allow attacks on tabular data. We theoretically show that our attack is not limited to spatio-temporal datasets but can be applicable to tabular data. We also plan to incorporate noise models other than Gaussian to test our attack against different noise scenarios. Another improvement for this attack is to include its best attack outcomes into his knowledge set after each iteration. This will increase the known points which is the main source of information. However in this case, the attacker's knowledge set will contain predicted points that will decrease the certainty of the conclusions as the number of predicted points increases. Another direction to extend this work is to attack different transformation techniques that are known to preserve data relations. This will eventually lead us to develop new techniques to prevent this kind of attacks that benefits the vulnerability of released relations.

In this thesis, we study privacy risks of spatio-temporal data transformations from the attacker's perspective utilizing his background information and the insights of the dataset if any. Using the common scenario in both works, we focus on trajectories and individual location points. We show that data transformation techniques preserving distances among the dataset as well as their relations are prone to such kind of attacks. As a future work, although the alternative techniques developed so far discussed in Chapter 2, one should consider preventive countermeasures to develop a privacy preserving transformation that maintains the pairwise distances and the relations but at the same time prevent such attacks studied in this thesis.

Bibliography

[1] A. Noulas, S. Scellato, C. Mascolo, and M. Pontil, “An empirical study of geographic user activity patterns in foursquare,” in *Proc. of the 5th Int’l AAAI Conference on Weblogs and Social Media*, pp. 570–573, 2011.

[2] E. Cho, S. A. Myers, and J. Leskovec, “Friendship and Mobility: User Movement in Location-based Social Networks,” in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD ’11, (New York, NY, USA), pp. 1082–1090, ACM, 2011.

[3] M.-J. Lee and C.-W. Chung, *A User Similarity Calculation Based on the Location for Social Network Services*, pp. 38–52. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011.

[4] Q. Li, Y. Zheng, X. Xie, Y. Chen, W. Liu, and W.-Y. Ma, “Mining user similarity based on location history,” in *Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS ’08, (New York, NY, USA), pp. 34:1–34:10, ACM, 2008.

[5] N. Eagle, A. S. Pentland, and D. Lazer, “Inferring friendship network structure by using mobile phone data,” *Proceedings of the National Academy of Sciences*, vol. 106, pp. 15274–15278, Sept. 2009.

[6] F. Giannotti, M. Nanni, F. Pinelli, and D. Pedreschi, “Trajectory pattern mining,” in *KDD ’07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 330–339, ACM, ACM, 2007.

[7] M. Ghasemzadeh, B. C. Fung, R. Chen, and A. Awasthi, "Anonymizing trajectory data for passenger flow analysis," *Transportation Research Part C: Emerging Technologies*, vol. 39, pp. 63–79, 2014.

[8] C. Y. Ma, D. K. Yau, N. K. Yip, and N. S. Rao, "Privacy vulnerability of published anonymous mobility traces," *IEEE/ACM Transactions on Networking*, vol. 21, no. 3, pp. 720–733, 2013.

[9] X. Li, J. Han, J.-G. Lee, and H. Gonzalez, "Traffic density-based discovery of hot routes in road networks," in *SSTD 2007: 10th International Symposium on Advances in Spatial and Temporal Databases*, Lecture Notes in Computer Science, pp. 441–459, Springer, 2007.

[10] "Geopkdd." http://www.geopkdd.eu.

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

[12] M. Wernke, P. Skvortsov, F. Dürr, and K. Rothermel, "A classification of location privacy attacks and approaches," *Personal and Ubiquitous Computing*, vol. 18, no. 1, pp. 163–175, 2014.

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

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

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

[16] C. Clifton, M. Kantarcioglu, and J. Vaidya, "Defining privacy for data mining," in *National Science Foundation Workshop on Next Generation Data Mining*, vol. 1, p. 1, Citeseer, 2002.

[17] J. Domingo-Ferrer and V. Torra, "Privacy in data mining," *Data Mining and Knowledge Discovery*, vol. 11, pp. 117–119, September 2005.

[18] "Privacy." http://en.wikipedia.org/wiki/Privacy.

[19] K. Liu, H. Kargupta, and J. Ryan, "Random projection-based multiplicative data perturbation for privacy preserving distributed data mining," *IEEE Trans. Knowl. Data Eng.*, vol. 18(1), no. 1, pp. 92–106, 2006.

[20] C. A. Ardagna, M. Cremonini, S. De Capitani di Vimercati, and P. Samarati, "An obfuscation-based approach for protecting location privacy," *IEEE Transactions on Dependable and Secure Computing*, vol. 8, no. 1, pp. 13–27, 2011.

[21] C. Bettini, X. S. Wang, and S. Jajodia, "Protecting privacy against location-based personal identification," in *Workshop on Secure Data Management*, pp. 185–199, Springer, 2005.

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

[23] H. Kargupta, S. Datta, Q. Wang, and K. Sivakumar, "On the privacy preserving properties of random data perturbation techniques," in *ICDM*, pp. 99–106, IEEE, 2003.

[24] X. LIU, "Protecting privacy in continuous location-tracking applications," *Minnesota Web-Based Traffic Generator*, 2004.

[25] V. S. Iyengar, "Transforming data to satisfy privacy constraints," in *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, (New York, NY, USA), pp. 279–288, ACM Press, 2002.

[26] P. Samarati and L. Sweeney, “Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression,” tech. rep., Technical report, SRI International, 1998.

[27] P. Samarati, “Protecting respondents’ identities in microdata release,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 13, pp. 1010–1027, 2001.

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

[29] E. Kaplan, T. B. Pedersen, E. Savaş, and Y. Saygın, “Discovering private trajectories using background information,” *Data & Knowledge Engineering*, vol. 69, no. 7, pp. 723–736, 2010.

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

[31] E. O. Turgay, T. B. Pedersen, Y. Saygın, E. Savaş, and A. Levi, “Disclosure risks of distance preserving data transformations,” in *SSDBM 2008: Scientific and Statistical Database Management Conference*, pp. 79–94, Springer, 2008.

[32] K. Liu, C. Giannella, and H. Kargupta, “An attacker’s view of distance preserving maps for privacy preserving data mining,” in *European Conference on Principles of Data Mining and Knowledge Discovery*, pp. 297–308, Springer, Springer, 2006.

[33] Y.-A. de Montjoye, C. A. Hidalgo, M. Verleysen, and V. D. Blondel, “Unique in the crowd: The privacy bounds of human mobility,” *Scientific Reports*, vol. 3, pp. 1376 EP –, 03 2013.

[34] S. E. Fienberg and J. McIntyre, “Data swapping: Variations on a theme by dalenius and reiss,” in *International Workshop on Privacy in Statistical Databases*, pp. 14–29, Springer, 2004.

[35] K. Muralidhar and R. Sarathy, “Data shuffling-a new masking approach for numerical data,” *Management Science*, vol. 52, no. 5, pp. 658–670, 2006.

[36] J. Domingo-Ferrer, K. Muralidhar, and G. Rufian-Torrell, “Anonymization methods for taxonomic microdata,” in *International Conference on Privacy in Statistical Databases*, pp. 90–102, Springer, 2012.

[37] R. Agrawal and R. Srikant, “Privacy-preserving data mining,” in *Proc. of the ACM SIGMOD Conference on Management of Data*, vol. 29, pp. 439–450, ACM, ACM Press, May 2000.

[38] L. Liu, M. Kantarcioglu, and B. Thuraisingham, “The applicability of the perturbation based privacy preserving data mining for real-world data,” *Data & Knowledge Engineering*, vol. 65, no. 1, pp. 5–21, 2008.

[39] J. Domingo-Ferrer, F. Sebé, and J. Castella-Roca, “On the security of noise addition for privacy in statistical databases,” in *International Workshop on Privacy in Statistical Databases*, pp. 149–161, Springer, 2004.

[40] Z. Huang, W. Du, and B. Chen, “Deriving private information from randomized data,” in *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, pp. 37–48, ACM, 2005.

[41] S. R. Oliveira and O. R. Zaïane, “Achieving privacy preservation when sharing data for clustering,” in *Workshop on Secure Data Management*, pp. 67–82, Springer, 2004.

[42] K. Chen, G. Sun, and L. Liu, “Towards attack-resilient geometric data perturbation.,” in *SDM*, pp. 78–89, SIAM, 2007.

[43] K. Chen and L. Liu, “Privacy preserving data classification with rotation perturbation,” in *IEEE International Conference on Data Mining (ICDM), 2005*, IEEE, 2005.

[44] C. R. Giannella, K. Liu, and H. Kargupta, “Breaching euclidean distance-preserving data perturbation using few known inputs,” *Data & Knowledge Engineering*, vol. 83, pp. 93–110, 2013.

[45] C. C. Aggarwal and S. Y. Philip, “A condensation approach to privacy preserving data mining,” in *International Conference on Extending Database Technology*, pp. 183–199, Springer, 2004.

[46] I. Ozalp, M. E. Gursoy, M. E. Nergiz, and Y. Saygin, “Privacy-preserving publishing of hierarchical data,” *ACM Transactions on Privacy and Security (TOPS)*, vol. 19, no. 3, 2016.

[47] B. Fung, K. Wang, R. Chen, and P. S. Yu, “Privacy-preserving data publishing: A survey of recent developments,” *ACM Computing Surveys (CSUR)*, vol. 42, no. 4, p. 14, 2010.

[48] J. Domingo-Ferrer and J. M. Mateo-Sanz, “Practical data-oriented microaggregation for statistical disclosure control,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 14, no. 1, pp. 189–201, 2002.

[49] J. Domingo-Ferrer, D. Sánchez, and J. Soria-Comas, “Database anonymization: Privacy models, data utility, and microaggregation-based inter-model connections,” *Synthesis Lectures on Information Security, Privacy, & Trust*, vol. 8, no. 1, pp. 1–136, 2016.

[50] C. Dwork and A. Roth, “The algorithmic foundations of differential privacy,” *Foundations and Trends in Theoretical Computer Science*, vol. 9, no. 3-4, pp. 211–407, 2014.

[51] M. Gruteser and 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*, pp. 31–42, ACM, 2003.

[52] B. Palanisamy and L. Liu, “Attack-resilient mix-zones over road networks: architecture and algorithms,” *IEEE Transactions on Mobile Computing*, vol. 14, no. 3, pp. 495–508, 2015.

[53] B. Palanisamy and L. Liu, “Effective mix-zone anonymization techniques for mobile travelers,” *GeoInformatica*, vol. 18, no. 1, pp. 135–164, 2014.

[54] B. Gedik and L. Liu, “Protecting location privacy with personalized k-anonymity: Architecture and algorithms,” *IEEE Transactions on Mobile Computing*, vol. 7, pp. 1–18, January 2008.

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

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

[57] R. Chen, B. C. Fung, N. Mohammed, B. C. Desai, and K. Wang, “Privacy-preserving trajectory data publishing by local suppression,” *Information Sciences*, vol. 231, pp. 83–97, 2013.

[58] O. Abul and F. Bonchi, “Never walk alone: Uncertainty for anonymity in moving objects databases,” in *The 24th International Conference on Data Engineering (ICDE 2008)*, pp. 376–385, IEEE, 2008.

[59] E. Nergiz, M. Atzori, and Y. Saygin., “Towards trajectory anonymization: a generalization-based approach,” in *In Proceedings of ACM GIS Workshop on Security and Privacy in GIS and LBS*, (CA, USA), pp. 52–61, ACM, 2008.

[60] J. Domingo-Ferrer, M. Sramka, and 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, pp. 26–33, ACM, 2010.

[61] R. Chen, B. Fung, B. C. Desai, and 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*, pp. 213–221, ACM, 2012.

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

[63] X. He, G. Cormode, A. Machanavajjhala, C. M. Procopiuc, and D. Srivastava, "Dpt: differentially private trajectory synthesis using hierarchical reference systems," *Proceedings of the VLDB Endowment*, vol. 8, no. 11, pp. 1154–1165, 2015.

[64] A. Gkoulalas-Divanis and V. S. Verykios, "A privacy-aware trajectory tracking query engine," *ACM SIGKDD Explorations Newsletter*, vol. 10, no. 1, pp. 40–49, 2008.

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

[66] H. Zhu, X. Meng, and G. Kollios, "Privacy preserving similarity evaluation of time series data.," in *EDBT*, pp. 499–510, 2014.

[67] M. Gowanlock and 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*, pp. 41–50, 2014.

[68] S. Mukherjee, Z. Chen, and A. Gangopadhyay, "A privacy-preserving technique for euclidean distance-based mining algorithms using fourier-related transforms," *VLDB Journal*, vol. 15, no. 4, pp. 293–315, 2006.

[69] S. Guo and X. Wu, "Deriving private information from arbitrarily projected data," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 84–95, Springer, 2007.

[70] B. D. Okkalioglu, M. Okkalioglu, M. Koc, and H. Polat, "A survey: deriving private information from perturbed data," *Artificial Intelligence Review*, vol. 44, no. 4, pp. 547–569, 2015.

[71] S. Sankararaman, P. K. Agarwal, T. Mølhave, J. Pan, and 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*, pp. 234–243, ACM, 2013.

[72] M. Vlachos, G. Kollios, and D. Gunopulos, "Discovering similar multidimensional trajectories," in *Data Engineering, 2002. Proceedings. 18th International Conference on*, pp. 673–684, IEEE, 2002.

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

[74] K. Chen and L. Liu, "Geometric data perturbation for privacy preserving outsourced data mining," *Knowledge and Information Systems*, vol. 29, no. 3, pp. 657–695, 2011.

[75] J.-W. Huang, J.-W. Su, and M.-S. Chen, "Fisip: A distance and correlation preserving transformation for privacy preserving data mining," in *2011 International Conference on Technologies and Applications of Artificial Intelligence*, pp. 101–106, IEEE, 2011.

[76] T. Brinkhoff, "A framework for generating network-based moving objects," *GeoInformatica*, vol. 6, no. 2, pp. 153–180, 2002.

[77] Y. Xiao and L. Xiong, “Protecting locations with differential privacy under temporal correlations,” in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp. 1298–1309, ACM, 2015.

[78] K. Chatzikokolakis, C. Palamidessi, and M. Stronati, “Constructing elastic distinguishability metrics for location privacy,” *Proceedings on Privacy Enhancing Technologies*, vol. 2015, no. 2, pp. 156–170, 2015.