Robust Fingerprint of Privacy-Preserving Location Trajectories

Yuzhou Jiang
Case Western Reserve University
Cleveland, Ohio, USA
yxj466@case.edu

Emre Yilmaz
University of Houston-Downtown
Houston, Texas, USA
yilmaze@uhd.edu

Erman Ayday
Case Western Reserve University
Cleveland, Ohio, USA
exa208@case.edu

ABSTRACT

Location-based services have brought significant convenience to people in their daily lives, and trajectory data are also in high demand. However, directly releasing those data raises privacy and liability (e.g., due to unauthorized distribution of such datasets) concerns since location data contain users' sensitive information, e.g., regular moving patterns and favorite spots. To address this, we propose a novel fingerprinting scheme that simultaneously identifies unauthorized redistribution of location trajectory datasets and provides differential privacy guarantees for shared data. Observing data utility degradation due to differentially private mechanisms, we introduce a utility-focused post-processing scheme to regain spatio-temporal correlations between points in a location trajectory. We further integrate this post-processing scheme into our fingerprinting scheme as a sampling method. The proposed fingerprinting scheme alleviates the degradation in the utility of the shared dataset due to the noise introduced by differentially private mechanisms (i.e., adds the fingerprint by preserving the publicly known statistics of the data). Meanwhile, it does not violate differential privacy throughout the entire process due to immunity to post-processing, a fundamental property of differential privacy. Our proposed fingerprinting scheme is robust against known and well-studied attacks against a fingerprinting scheme including random flipping attacks, correlation-based flipping attacks, and collusions among multiple parties, making it difficult for the attackers to infer the fingerprint codes and avoid accusation. Through experiments on two real-life location trajectory datasets and two synthetic ones, we show that our scheme achieves high fingerprint robustness and outperforms existing approaches. Furthermore, the proposed fingerprinting scheme increases data utility for differentially private datasets, which is beneficial to data analyzers.

KEYWORDS

digital fingerprinting, data privacy, location privacy, differential privacy

ACKNOWLEDGMENTS

The work was partly supported by the National Library of Medicine of the National Institutes of Health under Award Number R01LM013-429 and by the National Science Foundation (NSF) under grant numbers 2141622, 2050410, 2200255, and OAC-2112606.

Proceedings on Privacy Enhancing Technologies 2023(4), 5–20

https://doi.org/10.56553/popets-2023-0095

1 INTRODUCTION

Location-based services have become one of the most popular services in our daily lives thanks to rapid evolution in mobile technologies and the internet of things. Location-based service providers often require a large amount of location-based information from users to support their services. For example, Google Maps [2] collects accurate location data from users in real time and plans optimal routes during navigation and offers place suggestions while users search the app. Food delivery services, e.g., Doordash [1], demand approximate location information from users for restaurant recommendation and keep track of food couriers for a better user experience. Most individuals are subtly accustomed to the convenient lifestyles using these location-based services, and hence share their location data with such location-based service providers voluntarily (with consent). Thus, such service providers build large location datasets.

Datasets of location trajectories are of great use, and sharing them brings vast benefits. In addition to moving patterns, much more information is included and can be inferred from these datasets (e.g., age, job, or home address). By analyzing datasets, data analytics companies can offer proper suggestions to the service providers in order to improve their user experience, adjust marketing strategies, or even determine locations of new facilities. Advertisement companies can learn from these data to accurately promote to specific customers. Researchers can propose new approaches and validate them on these datasets.

Location-based service providers (e.g., Google) can share such trajectory datasets with a limited number of parties, called data analyzers. Some examples of data analyzers are researchers and analytic institutions. Access to location datasets is typically restricted within such analyzers parties as location datasets contain sensitive information. Nevertheless, malicious data analyzers, e.g., motivated by profit, may leak their copies to unauthorized parties, which brings significant privacy concerns. In order to prevent unauthorized redistribution, service providers should embed a unique fingerprint into datasets for each data analyzer to enable the traceability of potential leakage. Such fingerprint should be robust against multiple attacks, e.g., distortion attacks and collusion attacks, since the attackers may try distort it by modifying some points or even colluding with other malicious parties to get rid of accusation. By analyzing the embedded fingerprint in the leaked dataset, the service provider can identify the source of the leakage, withdraw its access to the dataset, and even punish it. Thus, knowing that the leaked dataset will be traced back to them, attackers become less motivated to leak the copies of the received datasets.

There are several existing fingerprinting mechanisms, e.g., Boneh-Shaw codes [5] and Tardos codes [31]. However, those traditional digital fingerprinting schemes cannot be directly applied to the location datasets because of correlations in the location datasets

and their particular utility requirements. In a location trajectory, i.e., an ordered sequence of location points in a location dataset, location points are highly correlated with each other, especially the adjacent location points. For instance, in a walking trajectory, recorded every 10 seconds, it is not likely to have two contiguous location points one kilometer apart from each other. Also, by knowing the previous and the following points in a given location trajectory, one can accurately estimate/infer the intermediate point with high confidence. Thus, using publicly available correlation models (constructed from public location datasets), an attacker can identify the points that violate the expected correlations as the fingerprinted data points. It can then distort or remove such identified data points (i.e., distort the fingerprint), making it harder for service providers to detect the source of a leaked dataset. We observed (and show via experiments) that existing fingerprint codes are vulnerable to such correlation-based attacks, since they do not consider pairwise correlations. Therefore, in this paper, we propose a robust correlation-based fingerprinting scheme that is robust against multiple attacks, e.g., including correlation attacks, majority collusion attacks, and probabilistic collusion attacks.

However, in recent years, privacy concerns about sensitive datasets have attracted massive attention. Researchers have also been investigating the privacy of trajectory datasets [4, 11]. User identities have been shown to be deanonymized with high confidence given only a pattern of four location points [11]. Therefore, simple anonymization on identifiers/quasi-identifiers is not sufficient to protect the individuals' location privacy. Under differential privacy (DP - a state-of-the-art concept for privacy preservation that quantifies and limits the information acquired from the attacker's perspective), researchers have proposed several solutions to mitigate privacy leakage while sharing location data, e.g., PIM [33] and AdaTrace [16]. However, existing privacy-preserving approaches for location data and datasets (i) do not provide liability guarantees against dataset leakage (unauthorized redistribution); and (ii) bring excessive noise to datasets and thus sacrifice data utility. Some location-based services (e.g., navigation) that do not tolerate such low utility may be unwilling to apply privacy protection to their datasets.

To the best of our knowledge, no existing work can tackle both issues, i.e., guaranteeing differential privacy and offering fingerprinting robustness, simultaneously. It is true that one can apply an arbitrary differentially private mechanism followed by an existing fingerprinting scheme, or vice versa. However, such differentially private mechanisms or fingerprinting schemes have their own drawbacks for location datasets. For instance, existing methods that achieve differential privacy on location datasets either omit critical information [16] or require impractical restrictions [20]. In terms of fingerprinting, existing schemes [5, 19, 31] are limited in their ability to account for correlations in location datasets. These schemes often require specific types of data, and they do not incorporate such correlations in their methodology, thus resulting in significant utility loss in the shared dataset. To solve these problems, we propose our solution that ensures a differential privacy guarantee and high fingerprinting robustness along with high data utility at the same time.

In this work, we introduce a robust fingerprinting scheme for location datasets that are protected under differential privacy using probabilistic sampling. The proposed scheme checks spatial and temporal correlations along the trajectories and considers highly probable location points based on public correlations during fingerprinting. The fingerprinting scheme offers high detection accuracy against multiple attacks against a fingerprinting scheme, e.g., random flipping attacks, correlation-based flipping attacks, majority collusion attacks, and probabilistic collusion attacks [34]. The selection of the privacy-preserving technique can be arbitrary. We select the planar isotropic mechanism (PIM) [33] as the building block to protect trajectory privacy. Other differentially private approaches can also be used (e.g., AdaTrace [16], a state-of-the-art synthetic approach to release location datasets under differential privacy). We demonstrate this flexibility of the proposed scheme through evaluations in Section 6.4.2. To mitigate data utility degradation due to privacy-preserving methods, we propose a utility-focused post-processing scheme that aims to restore correlations between adjacent points along a trajectory. During this process, we check the 2-gram transitions in the trajectory and replace each location point that has a low probability with a highly probable one by considering the directional information of the transition. We integrate this post-processing scheme into our proposed fingerprinting scheme such that the fingerprinting scheme can protect unauthorized redistribution and boost data utility at the same time.

We implement our proposed scheme using two real-life datasets, i.e., the GeoLife dataset [37] and the Taxi dataset [25], and two synthetic datasets generated from the Brinkhoff generator [7]. We compare our scheme with state-of-the-art fingerprinting approaches, i.e., Boneh-Shaw codes and Tardos codes, and evaluate the fingerprint robustness against random flipping attacks, correlation-based flipping attacks, majority collusion attacks, and probabilistic collusion attacks. We also evaluate data utility in terms of query answering of location points and patterns, area popularity, trip error, diameter error, and trajectory similarity. We observe that our scheme provides significantly better data utility than the existing approaches.

Our main contributions can be summarized as follows.

- We propose a probabilistic fingerprinting scheme that utilizes publicly known correlations for location datasets.
- We propose a utility-focused post-processing scheme to improve data utility for the location datasets that are protected under differential privacy and further integrate it into the proposed fingerprinting scheme.
- The fingerprinting scheme achieves high fingerprint robustness on differentially private datasets against several known attacks.
- We evaluate our proposed scheme concerning fingerprint robustness and data utility on four datasets, and show that our scheme outperforms state-of-the-art approaches.

The remainder of the paper is organized as follows. We review the existing work in Section 2 and provide the preliminaries in Section 3. We present the system and threat models in Section 4. In Section 5, we introduce the proposed scheme in detail. We evaluate our proposed scheme in Section 6. In Section 7, we discuss several topics related to our approach. Section 8 concludes the paper.

2 RELATED WORK

In this section, we introduce some existing works in location privacy and digital fingerprinting, respectively.

2.1 Trajectory Privacy

Location data contain sensitive information such as moving patterns and preferred locations. Traditional privacy enhancing techniques, e.g., k-anonymity [30] and l-diversity [24], have been adapted to the location setting. However, for a location dataset, those techniques have limitations in dealing with data streams with various lengths. For instance, some works [3, 14] split the trajectories into equal-length fragments and achieve privacy on the fragment, which is not sufficient for privacy protection on trajectories. Differential privacy [12] as a popular privacy definition has been used to protect location datasets in recent years [8, 28, 29, 35]. Geo-indistinguishability [4] defines a variant of differential privacy based on the distance between the points of interests, but it only works on location points instead of trajectories. Several methods [9, 16, 18] provide differential privacy to the statistics of the original location datasets. He et al. [18] design a hierarchical tree for storing regional spatial correlations and sample trajectories by walking along the tree paths. Gursoy et al. [16] extract four statistical features from a location dataset under differential privacy and generate a synthetic dataset using those noisy features. These works completely eliminate moving features of any specific user while preserving statistics, which improves user's location privacy but significantly decreases the usability of the dataset in certain services, e.g., map navigation and carpooling. Meanwhile, some researchers use perturbation-based approaches instead. [20] releases differentially private trajectories by sampling and interpolating them, but the scheme has the additional restriction that starting and ending locations should be known to the public. PIM [33] distorts each location point in a trajectory based on prior knowledge from previously released points. This approach is the only existing one that takes spatio-temporal correlations into consideration during differentially private release. However, it introduces zigzag patterns for lower privacy budgets (i.e., privacy protection is stronger) in the shared trajectories and loses pairwise correlation along a trajectory, making it also suffers from utility loss.

2.2 Digital Fingerprinting

Digital fingerprinting embeds a unique identifier, e.g., a sequence of marks, into the data by adding, removing, or editing partial values of the data. Several works have been proposed to enable digital fingerprinting for data distribution [5, 10, 31]. Boneh and Shaw design a fingerprint code and prevent receivers from colluding [5]. Tardos et al. propose a probability-based fingerprinting scheme that can catch all suspicious individuals simultaneously [31] and has less code length than the Boneh and Shaw's. Wu et al. introduce a fingerprinting scheme that embeds binary fingerprint codes towards multimedia [32]. However, those methods are designed for binary streams, where pairwise correlations are omitted in most cases. Considering correlations, some researchers aim to provide fingerprint robustness in data with various types, i.e., relational databases [19, 21–23]. These approaches only work on specific data types and cannot be applied to location datasets since location trajectories have high pairwise correlations. Considering correlations, [34] introduces a fingerprinting scheme for sequential data that considers correlations between data points. Still, it requires the possible states for a data point to be limited, discrete, and inter-transitable.

3 PRELIMINARIES

In this section, we first introduce the definition of differential privacy and its key property: immunity to post-processing. We then introduce two popular collusion-resistant fingerprinting schemes as the baseline approaches against collusion attacks. We integrate one of the schemes into our proposed robust fingerprinting scheme (i.e., the Boneh-Shaw codes) and compare it with the vanilla versions of these schemes in Section 6.

3.1 Differential Privacy

Differential privacy (DP) quantifies privacy and limits the inference of any single individual from observing the query results between neighboring databases. The formal definition is as follows:

Definition 3.1 (Differential Privacy). [12] For any neighboring datasets D, D' that differ only in one data record, a randomized algorithm $\mathcal{M}$ satisfies ϵ-differential privacy if for all possible outputs $\mathcal{S} \subseteq Range(\mathcal{M})$

$$Pr(\mathcal{M}(D) \in \mathcal{S}) \leq e^{\epsilon} * Pr(\mathcal{M}(D') \in \mathcal{S}).$$

An important proposition of differential privacy is its immunity to post-processing. It ensures that the differential privacy guarantee still holds when a mapping function is performed on the output from a differentially private mechanism as long as the function does not utilize the actual value. The formal definition is as follows:

Proposition 3.2 (Post-processing). *[13] Let $\mathcal{M}$ be a randomized algorithm that is ϵ-differentially private. For any arbitrary randomized mapping $f : \mathcal{R}^q \rightarrow \mathcal{R}^r$ where $p, q \in \mathbb{N}^+$, $f \circ \mathcal{M}$ is ϵ-differentially private.*

Hence, perturbations to the differentially private outputs without knowing the original values do not violate the privacy guarantee.

3.2 Planar Isotropic Mechanism

The planar isotropic mechanism (PIM) [33] aims to protect each location point along an individual's location trajectory under differential privacy. It constructs the correlations of a trajectory using a Markov chain, which is treated as a hidden Markov model from the attacker's perspective. Based on adversarial knowledge, i.e., the probability distribution of the location, the method adds calibrated noise to the actual location and shares the perturbed location. At timestamp t, let p_t^- and p_t^+ respectively represent the prior and posterior probability distributions, with $p_t^-[i]$ denoting the prior probability of location s_i in the location alphabet $\mathcal{G}$, and $p_t^+[i]$ corresponding to s_i's posterior probability. To share a noisy location, PIM calculates the prior probability distribution p_t^- as $p_t^- = p_{t-1}^+ M$, where M denotes the transition matrix. Based on the prior probabilities, it builds a δ-location set ΔX_t that contains the minimum number of locations with the probability sum greater than or equal to $1-\delta$, i.e., $\Delta X_t = min\{s_i | \sum_{s_i} p_t^-[i] \geq 1-\delta\}$, which means that a subset of locations with a total probability less than δ is omitted. After that, PIM releases the perturbed location given ΔX_t at timestamp t, and calls it z_t. The posterior probability distribution is then updated to $p_t^+[i] = Pr(\boldsymbol{u}_t^* = s_i | \boldsymbol{z}_t) = \frac{Pr(z_t | \boldsymbol{u}_t^* = s_i) p_t^-[i]}{\sum_j Pr(z_t | \boldsymbol{u}_t^* = s_j) p_t^-[j]}$ for each location s_j, where u_t^* is the true location at timestamp t.

The PIM generation can be summarized as follows:

(1) Generates a convex hull K' from ΔX_t;

Table 1: Symbols and notations.

$\mathcal{X} = [x_1, x_2, \ldots, x_{\lvert\mathcal{X}\rvert}]$	A trajectory
$\hat{\mathcal{X}} = [\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_{\lvert\hat{\mathcal{X}}\rvert}]$	The trajectory released by the differential privacy mechanism
$\mathcal{X}^* = [x_1^*, x_2^*, \ldots, x_{\lvert\mathcal{X}^*\rvert}^*]$	The trajectory released by the post-processing
$\mathcal{X}'_j = [x'_{1j}, x'_{2j}, \ldots, x'_{\lvert\mathcal{X}'\rvert j}]$	The fingerprinted trajectory of the data analyzer DA_j
$\mathcal{Y} = [y_1, y_2, \ldots, y_{\lvert\mathcal{Y}\rvert}]$	The leaked trajectory
$\mathcal{G}$	The location alphabet
m	The trajectory length, i.e., $\lvert\mathcal{X}\rvert$
p	The fingerprinting ratio
n	The number of data analyzers

(2) Builds a set ΔV_t by

$$\Delta V_t = \cup_{\boldsymbol{v}_1, \boldsymbol{v}_2 \in \text{ vertices of } K'} (\boldsymbol{v}_1 - \boldsymbol{v}_2)$$

(3) Forms a sensitivity hull (a convex hull) K from ΔV_t , which is a stricter sensitivity metric in two dimensions than the l_1 norm [33];
(4) Converts K into isotropic position K_I [33];
(5) Samples a point $\boldsymbol{z}'$ from K_I using the k-norm mechanism [17], i.e., the probability of each point $\mathbf{z}$ is

$$Pr(\mathbf{z}) = \frac{1}{\Gamma(d+1)\text{VOL}(K_I/\epsilon)} exp(-\epsilon||\mathbf{z} - \mathbf{x}^*||_{K_I})$$

, where $\mathbf{x}^*$ is the true answer, $||\cdot||_{K_I}$ is the Minkowski norm of K_I, d is the dimension ($d = 2$ in the location setting), $\Gamma()$ is Gamma function and VOL() is the volume, and ϵ is the privacy budget;
(6) Converts $\boldsymbol{z}'$ back to the original space as $\boldsymbol{z}$ and releases it as the final output at timestamp t.

By observing the output at each timestamp and knowing the transition matrix as auxiliary information, the attacker cannot infer the actual locations since the generation process models the attacker in the exact same way. This mechanism achieves ϵ-differential privacy for the trajectories in the location datasets. For further details, we refer the reader to the original paper [33].

4 PROBLEM STATEMENT

In this section, we describe the system setting, including the data model, the system model, and the threat model. Table 1 shows the commonly used notations in the paper.

4.1 Data Model

We introduce the data model for our system, including the format of trajectories, discretization, and correlations.

4.1.1 Trajectories. A trajectory $\mathcal{X} = [x_1, x_2, \ldots, x_{|\mathcal{X}|}]$ is an ordered sequence of location data points with the same time interval between any adjacent location points. In our setting, a location point x consists of GPS coordinates only, since we preprocess the trajectories to have uniform time interval and thus omit the timestamps. Although some secondary metadata, such as velocities and directions, can occur, we leave these to future work.

4.1.2 Map Discretization. In location settings, a map area is often discretized into cells for simplicity [9, 15, 18, 33]. Following those works, we divide the continuous two-dimensional space using a uniform grid of $N \times N$. Throughout the rest of the paper, we still use the term "points" to represent a cell of the grid for generalization.

4.1.3 Correlations. We build our correlations using the Markov chain. For each location $g \in \mathcal{G}$, the transition probability of the k-gram model is represented as $Pr[x_k|x_{k-1}, x_{k-2}, \cdots, x_1]$. We use the 2-gram model in our scheme ($k = 1$). We provide a discussion of the correlation model in Section 7.2.

4.2 System Model

The general workflow of the framework is shown in Figure 1. There are two parties in our setting: a service provider and several data analyzers. The service provider, e.g., Google Maps or a carpooling application, collects users' location trajectories while offering the corresponding service(s) to the users. The service provider stores the location dataset on their data server and is willing to share them with other parties. Meanwhile, researchers and businesses, classified as data analyzers, want to access such location datasets. As discussed above, the release of location data may raise privacy concerns. Therefore, the service provider aims to ensure users' location privacy before sharing. More specifically, it can apply a privacy-preserving approach that prevents recipients (data analyzers) from knowing the users' exact locations. This process inevitably perturbs the data and influences data utility, which is not desired by the analyzers, especially when strong protection is applied. To best serve the analyzers and keep the user privacy intact simultaneously, we propose a utility-focused postprocessing scheme at the service provider to partially regain data utility.

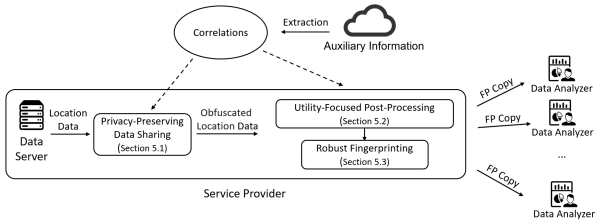


Figure 1: The system model.

As also discussed, a misbehaving data analyzer may distribute (leak) a copy of the received location dataset to other unauthorized parties without permission. Therefore, we propose a novel fingerprinting scheme for location trajectories, which embeds unique fingerprint patterns into each shared location dataset. The proposed scheme is robust in case the attacker tries to distort the fingerprint by exploiting the correlations among the location data from public sources or by colluding with other misbehaving data analyzers who also receive the same location dataset (with different unique fingerprint patterns). Furthermore, we convert the utility-focused post-processing method into a sampling strategy and integrate it into the fingerprinting scheme. In this way, we manage to mitigate utility degradation if differentially private mechanisms are applied in the shared dataset.

The fingerprint detection workflow (for the source of an unauthorized redistribution) is shown in Figure 2. Once a location dataset is found publicly or from unauthorized sources, the service provider performs an aggregate detection scheme to identify the source of the leakage. More specifically, it runs the detection scheme for each trajectory in the leaked dataset. The service provider aggregates the detection results (a set of accused analyzers), and finally accuses

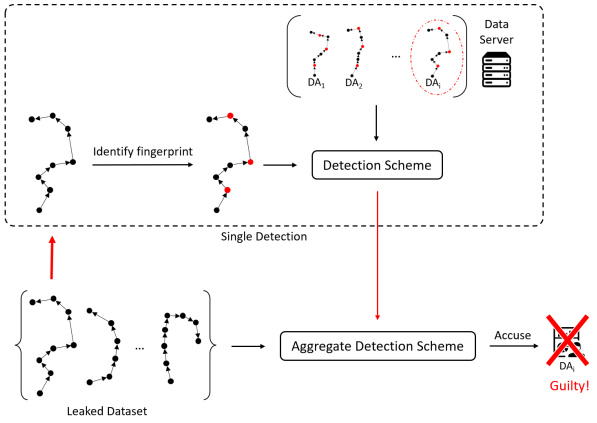


Figure 2: Detecting the source of the unauthorized redistribution.

an analyzer of leaking the dataset by majority voting. The details are given in Section 5.4.

4.3 Threat Model

In this section, we introduce the threat model considering the parties in our system. The service provider is the only entity that has access to the unperturbed data of the users. We assume that the service provider is trusted (i.e., it does not distribute user data to other unauthorized parties). The proposed scheme can be easily extended to provide privacy of users' data during the sharing process with the service provider (we discuss the practicality of a decentralized setting in Appendix C).

The analyzers can be malicious. An honest analyzer never shares the fingerprinted copy that is protected under a privacy-enhancing mechanism with unauthorized parties, and it does not want to know about the original dataset. An attacker, i.e., a malicious analyzer, is curious about the original (non-perturbed) data values in the received dataset and wants to break the location privacy guarantee. For this, they can utilize auxiliary information from public sources, e.g., correlations in the map area of interest. With the help of that information, they analyze the received trajectories and try to infer the original location points.

On the other hand, from the perspective of fingerprinting, the attacker may want to redistribute only one trajectory or a subset of the location dataset (i.e., multiple trajectories) to other parties, e.g., motivated by profit. To avoid tracking, the attacker tries to distort the fingerprint signature. They can exploit public correlations, collude with other analyzers, or even use both to hide their identities. In the rest of the section, we discuss all the attacks the analyzers can perform against the proposed fingerprinting scheme.

4.3.1 Random Flipping Attack. Random flip attacks are the baseline attack in which the attacker distorts the location points in the trajectory to distort the fingerprint. For each location point in the trajectory, the attacker chooses to report another point from the neighbors of the actual point with probability p_r. Otherwise, the attacker does not change the point and reports the actual point.

4.3.2 Correlation-Based Flipping Attack. The attacker can utilize public correlations to improve baseline distortion. This attack was first introduced in [34]. In this attack, the attacker analyzes the correlations between contiguous points along the trajectory from start to end. It checks the 2-gram transition from the previous point to the current one, i.e., $Pr(x_1 = x_j | x_0 = x_{j-1})$ at position j in the trajectory. If the transition probability is lower than a threshold τ, the attacker considers that the point is fingerprinted with a high probability. The attacker decides to distort the point with probability p_c. The attacker first constructs a set that contains all highly probable locations, i.e., the transition probability from the previous point x_{j-1} to each point in the set is at least τ. The attacker samples an output based on the transition probability from the last true point x_{j-1} to each point in the set. By doing so, the attacker distorts the suspicious positions and avoids being detected.

If multiple parties collude by sharing their copies with each other, they can perform more powerful attacks. We consider two types of collusion attacks in our setting, differing in whether the attackers take auxiliary information into account.

4.3.3 Majority Collusion Attack [5]. In the majority collusion attack, the attackers collude and analyze the merged dataset point by point. At each position, the attackers always choose the most frequent value as the output. The majority voting causes the trajectory to lose some fingerprint bits, which may mislead the fingerprint detection mechanism and result in accusing an innocent party.

4.3.4 Probabilistic Collusion Attack [34]. Similar to correlation-based flipping attacks, probabilistic collusion attacks [34] exploit the auxiliary information. The attackers share the datasets and analyze them using correlations, i.e., transition probabilities. They also set a probability p_e to approximate the actual fingerprinting probability p. Suppose that the attackers are deciding the output for the j-th position in a trajectory. The attackers collect all the location at position j to form an alphabet $G = \{g_1, g_2, \ldots, g_K\}$ at this position, where K is the number of distinct locations, and count the occurrence as $c_{j,k}$ for each location g_k, $k \in [1, K]$. The attackers filter those with low transition probabilities from the last released point y_{j-1}. Among the remaining set, they perform probabilistic sampling, where the probability is proportional to $(1-p_e)^{c_{j,k}} \cdot (\frac{p_e}{|G_j|-1})^{n-c_{j,k}} \cdot P(x_j = g_k | x_{j-1} = y'_{j-1})$, where G_j refers to the alphabet at position j. The first part $(1-p_e)^{c_{j,k}} \cdot (\frac{p_e}{|G_j|-1})^{n-c_{j,k}}$ is the probability of g_k being the original location at position j based on the assumed probability p_e, and the second part is the transition probability from the previous location. By combining the two parts, the attackers are able to calibrate such probability that a location with a very low probability is barely the true location even if it occurs multiple times, and a location with a high probability in the correlation model is more likely to be the true value although it occurs rarely. The attackers finally sample a location based on the weighted probability distribution and report that location at position j.

4.3.5 Re-Fingerprinting Attack. The attacker can execute the proposed fingerprinting scheme on the fingerprinted copy in order to perturb some embedded fingerprint points. We name this the re-fingerprinting attack. We consider that the attacker applies the fingerprinting scheme on the received dataset using a different fingerprinting ratio p_a.

5 METHODOLOGY

We follow the following steps for each trajectory in the dataset. First, we protect location datasets using a differentially private mechanism, i.e., the planar isotropic mechanism (PIM) [33]. After generating the differentially private dataset, we maximize the data utility of the shared dataset by applying a post-processing strategy and further integrate it into our probabilistic fingerprinting scheme. In the rest of this section, we provide the technical details of these mechanisms. In Section 5.1, we briefly explain the reason for choosing PIM as the building block and also compare it with other existing approaches. In Section 5.2, we introduce the post-processing scheme that regains pairwise correlations in the differentially private dataset. In Section 5.3, we propose our fingerprinting scheme and show how we integrate the post-processing scheme into our sampling process. In Section 5.4, we show how we detect an attacker. In addition, we prove that our scheme does not violate the privacy guarantee provided by the differentially private mechanism in Appendix A.

5.1 Privacy-Preserving Trajectory Data Sharing

We choose the planar isotropic mechanism [33] (PIM) as the building block to ensure trajectory privacy considering its three main advantages. First, PIM publishes trajectories with timestamps, while other approaches (e.g., [16]) do not. By preserving timestamps, PIM is able to provide more meaningful location trajectories, enhancing their overall value. Second, PIM and our proposed scheme share the same public information model, i.e., a correlation model generated from public sources. Third, as a perturbation-based method, PIM provides greater flexibility in selecting an appropriate noise level to balance privacy and utility. For instance, a user can either generate a noisy output with low data utility to services that have low utility requirements or release a less noisy one with high data utility to utility-sensitive services. Synthetic methods, in contrast, only preserve statistical features and omit other essential aspects (e.g., user-specific details), which leads to a significant loss of data utility even if a high privacy budget is allocated. Note that PIM ensures differential privacy in an area by eliminating low probable points (5% in total) at each timestamp, thus achieving a relaxation of differential privacy. We use this mechanism as our building block because other mechanisms suffer from significant limitations, e.g., lack of temporal information or trajectory-length restrictions, and thus are unable to be used for actual trajectory sharing in real-world scenarios. In addition, we show that our scheme is robust to a differentially private method (i.e., AdaTrace [16] after simple preprocessing) in Section 6.4.2.

Note that we do not generate a differentially private copy for each data analyzer. In our scheme, we apply the planar isotropic mechanism (PIM) only once for each trajectory in the dataset. After that, the same noisy dataset generated from the differentially private mechanism is used throughout the entire fingerprinting process. This is because sharing multiple outputs on the same input under differential privacy results in cumulative privacy loss [13], and this may be exploited if the attackers collude and perform averaging attacks to recover the original dataset. As a result, we choose to apply PIM once for each trajectory and then use the same noisy copy in our proposed fingerprinting scheme.

Similar to other perturbation-based approaches that ensure event-level differential privacy, PIM generates a large amount of noise for each location point under high privacy protection. This leads to significant utility loss in the shared location dataset, and the pairwise correlations inside are mostly very low for common ϵ values. Influenced by the two aforementioned factors, the data utility of the entire trajectory decreases significantly. In other words, the trajectories before and after perturbations differ considerably in terms of shape and point-wise relations. As a result, the dataset is almost unusable for the data analyzers as they can hardly infer meaningful pieces of information, e.g., moving trends and statistics, from the trajectories. To solve this problem, we propose our post-processing scheme, called utility-focused post-processing.

5.2 Utility-Focused Post-Processing

The utility-focused post-processing scheme utilizes the auxiliary information that is also used in PIM and from public sources to boost the data utility of the released trajectory data. We start with the definition of the τ-probable set in Definition 5.1.

Definition 5.1 (τ-Probable Set). Let $\tau \in [0, 1]$ and $\mathcal{G}$ be the set of discrete map areas. $\mathcal{M}$ is the 2-gram Markov model. Given a location point $g^* \in \mathcal{G}$, the τ-probable set of g^* is defined as

$$prob_\tau(g^*) \leftarrow \{g | Pr[x_1 = g | x_0 = g^*] \geq \tau\}, g \in \mathcal{G} \quad (1)$$

, where $Pr[x_1 = g | x_0 = g^*]$ is the transition probability obtained from the correlation model $\mathcal{M}$.

The idea of τ-probable set origins from [34], where the authors only consider pairwise data points with a transition probability larger than or equal to τ. We build the correlations using the 2-gram Markov chain (following [34]) and consider the transitions based on the previous locations in the trajectory.

In the post-processing scheme, we iterate the location points in the trajectory in sequential order. While post-processing the j-th location point of a differentially private trajectory, named $\hat{x}_j$, we first obtain the $(j-1)$-th output x^*_{j-1} that is generated from the post-processing scheme and calculate its τ-probable set $prob_\tau(x^*_{j-1})$. If $\hat{x}_j$ is in $prob_\tau(x^*_{j-1})$, the correlations are preserved between the two points, and thus we do not modify the points. Otherwise, the correlations do not exist. In this case, we choose the closest one to $\hat{x}_j$ within the τ-probable set as the output. The new point x^*_j is treated as the original value of the corresponding data point during the fingerprinting process.

Note that selecting the points in the τ-probable set depends on the transition probability from the correlation model. Thus, it is not guaranteed that the τ-probable set is a circle-like shape that covers all the directions of the previous location $\hat{x}$. Due to the insufficiency of the correlations generated from publicly available datasets, in some extreme cases, there exist no suitable location points in the set that are closer to $\hat{x}_j$ compared with the previous location x^*_{j-1}. If this happens and the trajectory trend continues, i.e., no turning back, the following outputs will fall into a pit. Figure 3 is an example of pit falling. x^*_{j-1} is the post-processed output at position $j-1$, and the τ probable set is marked using a dashed square. When deciding x^*_j, the scheme finds that $\hat{x}_j$ is outside of the τ-probable set, and thus it should choose the closest point to report. As the closest location is identical to the previous release x^*_{j-1}, the algorithm still reports

$x_j^* = x_{j-1}^*$. x_j^* and the remaining points $x_{j+1}^*, x_{j+2}^*, \cdots$ remain in the same position following the same process, causing the trajectory to fall into a pit. Our solution is to let $x_j^* = \hat{x}_j$ in this case. By doing so, we force the scheme to jump out of the pit so that the generation still follows the temporary trend of the trajectory. The complete algorithm of the post-processing scheme is shown in Algorithm 2 in Appendix D.

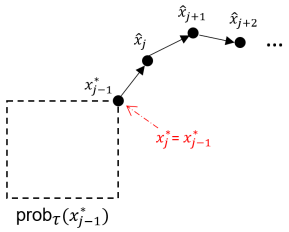


Figure 3: *Pit falling.* x_{j-1}^* is the last smoothed point. The following outputs $x_j^*, x_{j+1^*}, \cdots$ will stay at the same position as x_{j-1}^*, forming a pit.

5.3 Robust Fingerprinting

Traditional fingerprinting approaches [5, 31] do not consider spatial/temporal correlations and treat each point independently. However, the location points in a trajectory are highly correlated, especially the neighboring ones. Thus, modifying a location point without following the correlation model will make a point far away from its neighboring points such that the attacker can easily identify most of the fingerprint bits by checking pairwise correlations. The probabilistic fingerprinting scheme (PFS) [34] is the only existing approach that takes correlations into account during fingerprinting. However, the scheme in [34] requires that the states of the data be limited and intertransitable. If the number of states is large and they have sparse correlations, i.e., transitions only exist between a small portion of the state pairs, [34] starts having limitations. Additionally, PFS does not consider the privacy of shared data streams. In the following, we first briefly introduce PFS.

5.3.1 The Probabilistic Fingerprinting Scheme (PFS). PFS embeds fingerprint codes from the start to the end of a data stream, i.e., x_0 to $x_{|\mathcal{X}|-1}$. Suppose that we are generating the j-th position in a data stream $\mathcal{X}$, and the fingerprinting ratio, i.e., probability of a point being fingerprinted (perturbed), is p. While determining the output x'_j, PFS checks the transition probability $Pr[x_j = g | x_{j-1} = x'_{j-1}]$ for each g in the alphabet and filters those with low probability (i.e., less than a threshold σ). PFS then forms a probability distribution among the remaining values. If the original value is not eliminated, $Pr[x_j]$ is set to $1-p$ with the remaining p proportionally assigned to the rest according to their transition probabilities. If the original value of the corresponding data point at position j is eliminated, the scheme only generates the output proportionally from the remaining values.

However, PFS cannot be applied to location datasets even without privacy protection. The most critical problem is forced deviation. PFS process normally works in location fingerprinting, but when the correlations are low between the data points, it starts to show limitations. According to PFS, the scheme eliminates the original value of the corresponding data point if the correlations do not hold. Then, the scheme proportionally samples a point from the remaining τ-probable set consisting of highly probable points and reports that one. In trajectory fingerprinting, once the sampled output appears outside of the next point's τ-probable set, the rest of the points will wander around the τ-probable set forever. We show this in Figure 4 as an example. Here, PFS fingerprints the j-th position in the trajectory, while x'_{j-1} is the last fingerprinted location and the dashed circular area in black is the τ-probable set of x'_{j-1}. x_j^* is the actual location at position j, and it is in the τ-probable set of x'_{j-1}. PFS wants to sample a point among the τ-probable set and releases that point. If the sampled point is located as x'_j in Figure 4, we realize that the next original value of the corresponding data point x_{j+1}^* is not in the τ-probable set of x'_j. In this case, the scheme will sample a location only among the set, regardless of the distance from the original value. The next original value x_{j+2}^* will be more likely to occur outside of the τ-probable set (marked by a red dashed circle) as well since the actual trajectory moves forward and the sampled output sticks to the area close to the first separation, i.e., x'_j. If the generation continues, the fingerprinted locations will be sampled around the first deviated location x'_j, and this will finally result in a forced deviation. We show some examples for this scenario for better clarification by applying PFS and our proposed scheme on two trajectory samples in Figure 5. As shown, PFS falls into forced deviation at the very beginning for each sample, while our approach generates fingerprints along the trajectory (i.e., the right figures).

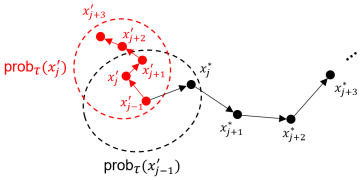


Figure 4: *Forced deviation.* The generated point x'_j at timestamp j is sampled inside the τ-probable set of the previous release x'_{j-1}. However, the next original value of the corresponding data point x_{j+1}^* is outside its τ-probable set. Following FPS, the next points will be sampled among $prob_\tau(x'_j)$ only.

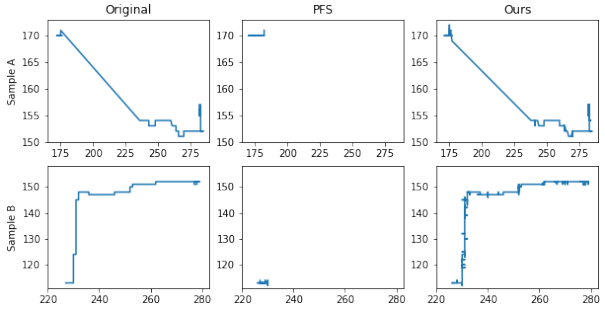


Figure 5: Visualization of two fingerprinting schemes, i.e., (i) PFS [34] and (ii) our scheme, on two trajectory samples. Forced deviation is clearly shown in the copies using PFS.

5.3.2 Direction-Sensitive Fingerprinting Scheme For Location Trajectories. To solve the aforementioned challenges, we propose a new sampling scheme, called the direction-sensitive fingerprinting scheme (see Algorithm 1 for details). For a released point x'_{j-1}, we first form a set containing all locations closer to or equal to x^*_j than x'_{j-1} in the τ-probable set, called the **τ-closer set**, which can be expressed as $prob^c_\tau(x'_{j-1}) \leftarrow \{g \big| \|g, x^*_j\|_2 \leq \|x'_{j-1}, x^*_j\|_2, g \in prob_\tau(x'_{j-1})\}$. Normally, if the original value of the corresponding data point x^*_j is in the τ-closer set, we sample the output among it by setting the probability of choosing the original value as $1-p$ and the rest are assigned proportionally based on the transition probability to the destination. We improve the sampling process to avoid forced deviation during generation. There are four cases in which the original value is selected at the j-th position. If the original value x^*_j is in the τ-closer set of the previously released location x'_{j-1}, there is no difference between ours and in PFS. If x^*_j is not in the τ-closer set, we check its membership in the τ-probable set and sample from the same distribution as above, but among the τ-probable set instead. If not, we check the closest point $\tilde{x}$ to the original value x^*_j in $prob_\tau(x'_{j-1})$. If $\tilde{x}$ is the same as x'_{j-1}, which means that there is no such location closer to x^*_j, we let the true temporary value be x^*_j. Otherwise, we choose $\tilde{x}$ as the temporary original value at this timestamp and perform the proportional sampling scheme. For the first location x^*_0 in the trajectory, we do not have conditional probabilities. Instead, we use the emission probability of x^*_0's neighboring locations, i.e., $Pr(g) = \frac{(\#\text{ of points at } g)}{\sum_{g' (\#\text{ of points at } g'), g' \in neigh(x^*_0)}}, g \in neigh(x^*_0)$, where $neigh(x)$ denotes a set of all neighbors of x^*_0 (including x^*_0 itself), in the sampling process.

In order to offer fingerprint robustness and data usability at the same time, we integrate the proposed post-processing scheme in Section 5.2 into our fingerprinting. In particular, if the next original value x^*_j is not in the τ-probable set, we follow the post-processing scheme to choose the closest point as the surrogate, and assume it to be the original value. This post-processing integration does not take effect if we work on not differentially private trajectories, as pairwise correlations are preserved along those trajectories. If dealing with noisy trajectories, i.e., protected under differential privacy, the post-processing step will regain pairwise correlations and thus improve data utility for location datasets.

In addition, we follow [34] and use the balancing strategy. During the fingerprint generation, some positions are perturbed, while some remain the same as the original values. We use *FP*s and *NoFP*s to represent them, respectively. PFS balances the distribution of the *FP*s by using the balancing factor θ. The scheme checks the *FP* count every $\lceil \frac{1}{p} \rceil$ points. If the actual *FP* count is larger than expected, then the temporary fingerprint ratio is changed to $p*(1-\theta)$. If the number of *FP* is not enough, the ratio becomes $p*(1+\theta)$. The complete algorithm is shown in Algorithm 1.

Algorithm 1: Direction-Sensitive Fingerprinting Scheme

input : Trajectory $\mathcal{X}^* = [x^*_1, x^*_2, \ldots, x^*_m]$, location alphabet $\mathcal{G}$, emission probability $Pr[g_p]$ and transition probability $Pr[g_q|g_r]$ for any locations $g_p, g_q, g_r \in \mathcal{G}$, probability threshold τ, fingerprinting ratio p, ratio balancing factor θ, the first fingerprinted trajectory $\mathcal{X}^\iota = [x^o_1, x^o_2, \ldots, x^o_m]$

output : Fingerprinted trajectory $\mathcal{X}' = [x'_1, x'_2, \ldots, x'_m]$

$PD \leftarrow Pr[x'_1 = x^*_1] = 1 - p_{current}, Pr[x'_1 = g] = \frac{Pr[g]}{\sum_{g' \in \mathcal{G} \setminus x^*_1} Pr[g']}, g \in \mathcal{G}$;

$x'_1 \leftarrow$ sample from PD;

$p_{current} = p$;

forall $j \in 2, 3, \ldots, m$ **do**

  $prob_\tau(x'_{j-1}) \leftarrow \tau$-probable set of x'_{j-1};

  $prob^c_\tau(x'_{j-1}) \leftarrow \{g | \|g, x^*_j\|_2 \leq \|x'_{j-1}, x^*_j\|_2, g \in prob_\tau(x'_{j-1})\}$;

  if $x^*_j \in prob^c_\tau(x'_{j-1})$ *and* $|prob^c_\tau(x'_{j-1})| > 1$ **then**

    $PD \leftarrow Pr[x'_j = x^*_j] = 1 - p, Pr[x'_j = g] = \frac{Pr[x'_j = g | x'_{j-1}]}{\sum_{g' \in prob_\tau(x'_{j-1}) \setminus x^*_j} Pr[x'_j = g' | x'_{j-1}]} * p, g \in prob^c_\tau(x'_{j-1})$;

    $x'_j \leftarrow$ sample from PD;

  else if $x^*_j \in prob_\tau(x'_{j-1})$ *and* $|prob^c_\tau(x'_{j-1})| == 1$ **then**

    $PD \leftarrow Pr[x'_j = x^*_j] = 1 - p, Pr[x'_j = g] = \frac{Pr[x'_j = g | x'_{j-1}]}{\sum_{g' \in prob_\tau(x'_{j-1}) \setminus x^*_j} Pr[x'_j = g' | x'_{j-1}]} * p, g \in prob_\tau(x'_{j-1})$;

    $x'_j \leftarrow$ sample from PD;

  else

    $x_{closest} \leftarrow$ closet point to x^*_j in $prob_\tau(x'_{j-1})$;

    if $|prob_\tau(x'_{j-1})| <= 1$ **then**

      $x'_j \leftarrow x^*_j$;

    else

      $PD \leftarrow Pr[x'_j = x_{closest}] = 1 - p, Pr[x'_j = g] = \frac{Pr[x'_j = g | x'_{j-1}]}{\sum_{g' \in prob_\tau(x'_{j-1}) \setminus x_{closest}} Pr[x'_j = g' | x'_{j-1}]} * p, g \in prob_\tau(x'_{j-1})$;

      $x'_j \leftarrow$ sample from PD;

  if $j \ mod \lceil \frac{1}{p} \rceil == 0$ **then**

    $count \leftarrow$ # of fingerprinted positions;

    if $count > p * j$ **then**

      $p_{current} \leftarrow p * (1 - \theta)$;

    else if $count < p * j$ **then**

      $p_{current} \leftarrow p * (1 + \theta)$;

    else

      $p_{current} \leftarrow p$;

end

5.4 Detecting the Source of the Unauthorized Redistribution

We use similarity-based detection [34] with our improvement. During traditional similarity-based detection, data points in leaked data are compared with distributed copies. At each position, if the leaked data point matches some data analyzers, each of them will be assigned a score $\frac{1}{|X|}$, where $|X|$ is the length of the data. After inspecting all data points, the analyzer with the highest cumulative score is considered malicious. In location data, slight perturbation is enough to invalidate those exact matches and thus influence the detecting accuracy. Thus, we replace it with a distance-based match in similarity-based detection. For each location point in the trajectory, we assign $\frac{1}{|X|}$ to all points that have the shortest distance from the

leaked location point instead of exact matches, which significantly improves our detection.

The described detection works on a single trajectory leakage from a shared trajectory dataset. For a multi-trajectory leakage, we implement an aggregate detection scheme to identify the source of the unauthorized redistribution. We first use distance-based detection to analyze leaked trajectories one by one in the leaked dataset and accuse one of being malicious for each leaked trajectory. Among all the accused data analyzers, we do majority voting on them and choose the most frequent one as the final malicious data analyzers. The evaluation results of multi-trajectory leakage are in Appendix E.2.

6 EVALUATION

We implemented the proposed fingerprinting scheme and provide the experimental results. We first evaluate the fingerprint robustness of our scheme against multiple attacks. After that, we evaluate it on the datasets protected by an alternative privacy-preserving method (i.e., AdaTrace [16]) and show that we still achieve similar performance against the considered attacks. In terms of data utility, we evaluate fingerprinted datasets using five utility metrics mentioned in Section 6.3.2. Furthermore, we performed parametric experiments on trajectory length (in Section 6.4.4) and time complexity (in Section 6.5.1). For the experiments, we used a rack server with 64GB memory (DDR4, 2666Mhz) and an Intel Xeon E5-2650 @ 2.20GHz with 40 cores. We run all experiments more than $1,000$ times with 20 dataset shuffles and take the average, and the 95% confidence intervals represented as shaded areas in the figures.

6.1 Datasets

We used 4 datasets during evaluation: 1) the GeoLife dataset (Version 1.3) [37], 2) the Taxi dataset [25], 3) the Oldenburg dataset [7], and 4) the San Joaquin dataset [7]. The GeoLife and Taxi datasets are real-life ones, and the Oldenburg and San Joaquin datasets are synthetic ones from the Brinkhoff generator. The GeoLife dataset contains $17,621$ trajectories generated by 182 users using different GPS devices over five years (April 2007-August 2012), including $1,292,951$ kilometers in distance and $50,176$ hours in time, where most of the locations are in Beijing, China. The Taxi dataset is used in the Taxi Service Prediction Challenge at ECML-PKDD 2015 [25], including $1,710,670$ taxi trajectories in Porto, Portugal. The remaining two datasets are synthesized from the Brinkhoff generator for moving objects [7] in the cities of Oldenburg and San Joaquin, respectively. We generate $5,000$ trajectories for each dataset.

6.1.1 Data Pre-Processing. We preprocessed the trajectories to avoid various data intervals. We smoothed the trajectories to have similar time intervals, i.e., around 60 seconds. For each dataset, we defined an area of interest that covers most of the trajectories and cut and filtered out the trajectory fragments outside the area. We picked $1,000$ trajectories as our fingerprinting targets and used the remaining ones to build public correlations.

6.2 Experimental Settings

We compare our fingerprinting scheme with two traditional ones, i.e., the Boneh-Shaw codes and the Tardos codes. We evaluate detection accuracy of the three schemes on both non-differentially private and differentially private datasets. The Boneh-Shaw codes and the Tardos codes do not support detection of multiple trajectories, so we use the same detection logic as ours, i.e., working on trajectories one by one and then majority voting, to fit our experiments.

The following experiments assume that the attacker(s) will only leak one trajectory from the entire dataset. As we mentioned in Section 5.4, we perform detection one by one on each leaked trajectory and do majority voting for the final accusation. The detection processes of leaked trajectories are independent from each other, which makes the problem become a combination problem (i.e., given detection accuracy for a single trajectory equal to p, what is the detection accuracy of k trajectories using majority voting?). As we will show in the following sections, our approach significantly outperforms existing schemes and maintains 90% detection accuracy in most cases. If multiple trajectories are leaked, the overall detection accuracy increases and reaches 99.99%. We show this in Appendix E.2. For simplicity, we consider only the leakage of one trajectory in the following.

6.2.1 Parameter Settings. If not specified, we use the following parameter setting throughout the experiments. An original dataset contains 100 randomly selected trajectories, and each has 100 locations. We assume that 100 SPs get the copies by default. We set $\tau = 0.005$ as the correlation threshold concluded from our experiments and the fingerprint balancing factor $\theta = 0.5$. The Tardos codes use $\omega = 0.01$ as the error probability. The Boneh-Shaw codeword consists of $|X|$ blocks and 1 location points in each block. For PIM, we follow [33] and set $\delta = 0.01$ for the δ-location set. The fingerprint ratio is set to 0.4. We assume that the attacker(s) uses $p_c = 0.8$ and $p_r = 0.8$ in random and correlation-based flipping attacks, respectively, and 3 service providers collude by default.

6.3 Evaluation Metrics

6.3.1 Fingerprint Robustness Metric. We define a successful accusation as correctly identifying the attacker who leaks the data. Our evaluation metric of fingerprint robustness is then represented as $Accuracy = \frac{(\#\text{ of successful accusation})}{(\#\text{ of trials})}$. If multiple attackers collude, we consider catching one of the colluding attackers. Since the Tardos codes focus on catching all those who leak the data, we adjust the accusation process for alignment. More specifically, we only consider the one with the highest scores in the Tardos detection instead of using the threshold $20ck$ (in Appendix B.2).

6.3.2 Utility Metrics. Following the existing work [16, 18, 33], we introduce our utility metrics as follows.

Query Answering of Location Points. The count query is one of the most frequent usages for location datasets.

Let $Q_t(D, g)$ denote the query "how many trajectories pass a circular area represented by a center c and a radius r in the dataset D". Then, we define the relative error as

$$AvRE = \frac{|Q_t(D, g) - Q_t(D', g)|}{max(Q_t(D, g), b)}$$

, where D is the original dataset and D' is the output of our scheme. We set $b = 0.01 \times |D|$ according to [9, 16, 26, 36].

Query Answering of Patterns. We also implement another query answering metric for patterns. As discussed in Section 4.1.3, we only focus on the 2-gram patterns. Given a 2-gram pattern P, the count query on P is $Q_p(P, D)$ that counts P in the dataset D. We also evaluate the utility using relative error.

Area Popularity. We follow [16] and evaluate the divergence of the popularity rankings by area. Based on the number of location points within each area, we generate the popularity ranking for each fingerprinting scheme. We compare the ranking with that of the original data set and calculate the Kendall-tau coefficient, which is defined as $KT = \frac{(\#\text{ of concordant pairs}) - (\#\text{ of discordant pairs})}{(\#\text{ of pairs})}$. The Kendall-Tau coefficient measures the ordinal association between sequences. A higher coefficient represents a better utility.

Trip Error. Trip error [16] measures trip length. We calculate the lengths of all the trajectories in the dataset and put them into 11 bins, i.e., $[0, \frac{L}{10}), [\frac{L}{10}, \frac{2L}{10}), \cdots, [\frac{9L}{10}, L)$, and $[L, \infty)$, where L is the maximum trip length in the original dataset. We calculate the Jensen-Shannon divergence (JSD) between the fingerprinted dataset and the original dataset.

Diameter Error. Diameter error [16] is similar to the trip error, but considers the distances between contiguous location points along the trajectories. We use the 11 bins and then evaluate the Jensen-Shannon divergence.

Trajectory Similarity. In services like carpooling, the shape of the trajectory is an important feature that can be used by the service to design an optimal strategy. We use 2-dimensional dynamic time wrapping (DTW) [27] to evaluate the similarity between the original and fingerprinted datasets.

6.4 Fingerprint Robustness

We show the experiment results of fingerprint robustness against five attacks in Section 4.3, i.e., random flipping attacks, correlation-based flipping attacks, majority collusion attacks, probabilistic collusion attacks, and re-fingerprinting attacks. Here, we represent four of the attacks using abbreviations for simplicity. In particular, "RF" denotes random flipping attacks, and "CF" represents correlation-based flipping attacks. "MJR" and "PROB" are majority collusion attacks and probabilistic collusion attacks, respectively. Due to page limitation, we defer several experiments, i.e., 1) fingerprint robustness on three datasets (i.e., Taxi [25], OldenBurg [7], and San Joaquin [7]), 2) fingerprint robustness when an alternative differentially private method (i.e., AdaTrace][16]) is used, and 3) parameterized experiments regarding the number of leaked trajectories to the appendix E.

6.4.1 Fingerprint Robustness on Datasets Protected by PIM. Our scheme performs significantly better (shown in Figure 6) compared with the existing methods. The proposed scheme achieves around 99.9% detection accuracy against random flipping attacks, majority collusion attacks (with a slight drop for a larger collusion count), and probabilistic collusion attacks. In terms of correlation-based flipping attacks, our scheme achieves 99.9% accuracy when $p_c < 0.8$, and it achieves 98% accuracy if $p_c = 0.8$. Note that a large fingerprint ratio does not mean high detection accuracy for our scheme. This is because we leverage pairwise correlations. If we choose to embed a fingerprint digit at a position, we distort the original point. However, the new point has lower pairwise correlations in most cases, which eventually influences the internal correlations among the trajectory and thus decreases the detection accuracy. Based on our experiments, a fingerprint ratio around 0.4 is optimal. Meanwhile, Boneh-Shaw codes and Tardos codes are not as robust as our proposed scheme. In conclusion, our scheme outperform those methods.

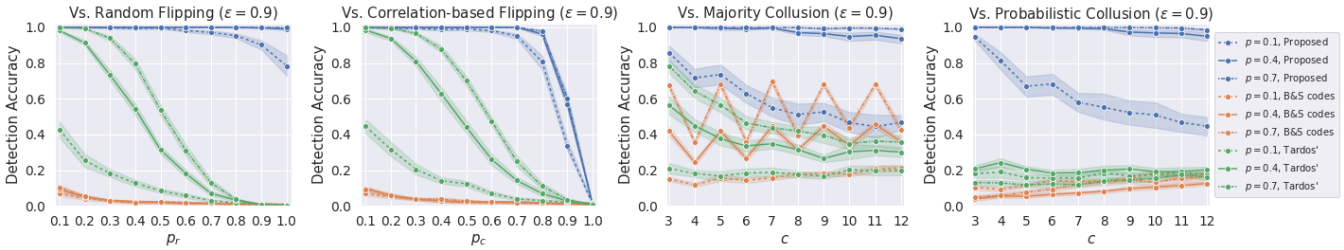
6.4.2 Fingerprint Robustness on Datasets Protected by an Alternative Method. In order to show that our framework works with different differentially private mechanisms on location datasets, we implement an alternative DP mechanism, i.e., AdaTrace [16]. However, AdaTrace is a synthetic mechanism that does not preserve any temporal information in the released dataset. Thus, we post-process the output dataset from AdaTrace using the Bresenham's algorithm, a line drawing algorithm, to traverse all passed points between two points and add time-sequenced indexes to each point. By using the Bresenham's algorithm and then assigning timestamps manually, we generated a synthetic dataset with high pairwise correlations but a fake version. As shown in Figure 8, we achieve similar results compared to those in the original datasets (i.e., in Figure 13) in general, which proves that our scheme can work on other differentially private mechanisms.

6.4.3 Fingerprint robustness against re-fingerprinting attack. The attacker can distort the embedded fingerprint by applying the proposed fingerprinting scheme on the received dataset, namely re-fingerprinting attacks. To evaluate our scheme against such attacks, we design the experiment as follows. We build small data sets of different sizes for evaluation. We assume that the attacker, based on the experiment results in Sections 6.4.1, and 6.4.2, chooses the optimal parameters, i.e., $\tau = 0.005, \theta = 0.5$, and applies the proposed fingerprinting scheme to the received dataset (which is also fingerprinted by the data owner). The attack ratio in this attack refers to the fingerprint ratio that the attacker uses. As shown in Figure 7, our scheme offers high fingerprint robustness against re-fingerprinting attacks regardless of the number of trajectories in the dataset, while larger datasets (with more trajectories) lead to higher detection accuracy. When the trajectory count exceeds 10, the detection accuracy reaches 98% for any fingerprint ratio that is not greater than 0.8 and remains above 60% even if the attack ratio reaches 0.9. For higher attack ratios, similar to correlation-based attacks discussed in Section 6.4.1, the resulting low data utility limits the attacker from executing such attacks.

6.4.4 Fingerprinting Robustness on Differentially Private Datasets for Trajectories with Different Lengths. We evaluate the performance of fingerprint robustness on trajectories of different lengths and show the results in Figure 9. Our scheme significantly outperforms existing methods in all aspects, except for a high fingerprint ratio $p = 0.7$.

6.5 Utility Evaluation

Table 2 shows the data utility of the proposed scheme and compares it with the original dataset. For $\epsilon = 0.9, 1.7,$ and 2.5, our proposed method is better than Boneh-Shaw codes and Tardos codes in most cases. Meanwhile, our scheme is not the best for query answering (when $\epsilon = 2.5$) on the Taxi and San Joaquin datasets and for popularity analysis (when $\epsilon = 0.9$) on the Oldenburg dataset. On the Taxi



(a) vs. RF **(b) vs. CF** **(c) vs. MJR** **(d) vs. PROB**

Figure 6: Fingerprint robustness of the proposed scheme on the differentially private GeoLife [37] dataset by PIM [33] compared with two existing methods, i.e., the Boneh-Shaw codes [5] and the Tardos codes [31], under different fingerprint ratio p.

Table 2: *Utility Evaluation.* "DSFS" is the proposed scheme in this paper, "BS" denotes the Boneh-Shaw codes, and "Tardos" refers to the Tardos codes. Better results are marked in bold. For Popularity KT coefficient, higher values are better. For the rest of the metrics, lower is better. We show 95% confidence intervals for all results.

		$\epsilon = 0.9$			$\epsilon = 1.7$			$\epsilon = 2.5$		
		DSFS	BS [5]	Tardos [31]	DSFS	BS [5]	Tardos [31]	DSFS	BS [5]	Tardos [31]
GeoLife [37]	QA Area AvRE	**9.6 ± 3.6**	12.2 ± 3.3	18.7 ± 5.3	**2.8 ± 0.9**	3.9 ± 1.6	3.3 ± 1.4	**0.9 ± 0.4**	1.6 ± 0.7	1.3 ± 0.3
	QA Pattern AvRE	**2.5 ± 0.4**	4.3 ± 0.5	4.8 ± 0.5	**1.0 ± 0.2**	2.0 ± 0.3	1.8 ± 0.4	**0.5 ± 0.1**	1.0 ± 0.2	0.9 ± 0.2
	Popularity KT [16]	**0.62 ± 0.01**	0.56 ± 0.01	0.57 ± 0.02	**0.74 ± 0.01**	0.68 ± 0.02	0.69 ± 0.01	**0.83 ± 0.02**	0.79 ± 0.02	0.78 ± 0.01
	Trip Error [16]	**0.75 ± 0.01**	0.81 ± 0.01	0.81 ± 0.01	**0.66 ± 0.01**	0.78 ± 0.01	0.79 ± 0.01	**0.54 ± 0.02**	0.71 ± 0.01	0.69 ± 0.01
	Diameter Error [16]	**0.14 ± 0.00**	0.31 ± 0.00	0.31 ± 0.00	**0.12 ± 0.00**	0.24 ± 0.00	0.24 ± 0.00	**0.11 ± 0.00**	0.21 ± 0.00	0.20 ± 0.00
	DTW Distance	**308 ± 10**	409 ± 9	400 ± 11	**146 ± 5**	182 ± 6	180 ± 7	**78 ± 3**	101 ± 3	101 ± 4
Taxi [25]	QA Area AvRE	**8.4 ± 3.5**	9.6 ± 4.0	13.7 ± 5.1	**0.7 ± 0.4**	2.1 ± 1.4	1.8 ± 1.3	0.34 ± 0.25	**0.33 ± 0.23**	0.53 ± 0.29
	QA Pattern AvRE	**7.5 ± 2.0**	9.3 ± 1.7	9.2 ± 1.4	**0.85 ± 0.44**	1.83 ± 0.69	2.99 ± 1.39	**0.25 ± 0.07**	0.74 ± 0.22	0.66 ± 0.20
	Popularity KT [16]	**0.54 ± 0.03**	0.53 ± 0.02	0.51 ± 0.03	**0.69 ± 0.04**	0.68 ± 0.03	0.68 ± 0.02	**0.83 ± 0.05**	0.80 ± 0.03	0.77 ± 0.04
	Trip Error [16]	**0.69 ± 0.01**	0.80 ± 0.01	0.80 ± 0.01	**0.45 ± 0.02**	0.77 ± 0.01	0.77 ± 0.01	**0.36 ± 0.02**	0.64 ± 0.01	0.63 ± 0.02
	Diameter Error [16]	**0.11 ± 0.00**	0.30 ± 0.00	0.29 ± 0.00	**0.07 ± 0.00**	0.21 ± 0.00	0.21 ± 0.00	**0.06 ± 0.00**	0.17 ± 0.00	0.17 ± 0.00
	DTW Distance	**196 ± 4**	257 ± 6	249 ± 5	**75 ± 3**	98 ± 2	100 ± 3	**42 ± 1**	54 ± 2	56 ± 2
Oldenburg [7]	QA Area AvRE	**1.4 ± 0.3**	2.0 ± 0.6	2.7 ± 0.6	**0.34 ± 0.13**	0.37 ± 0.10	0.41 ± 0.11	**0.13 ± 0.04**	0.18 ± 0.07	0.16 ± 0.07
	QA Pattern AvRE	**3.4 ± 0.5**	6.3 ± 0.3	6.5 ± 0.5	**1.6 ± 0.2**	3.3 ± 0.2	3.0 ± 0.2	**0.8 ± 0.1**	2.0 ± 0.2	1.9 ± 0.1
	Popularity KT [16]	0.69 ± 0.01	**0.70 ± 0.01**	**0.70 ± 0.01**	**0.84 ± 0.01**	0.83 ± 0.01	0.83 ± 0.01	**0.90 ± 0.01**	0.89 ± 0.01	0.89 ± 0.01
	Trip Error [16]	**0.70 ± 0.01**	0.80 ± 0.01	0.80 ± 0.01	**0.53 ± 0.02**	0.76 ± 0.01	0.76 ± 0.01	**0.44 ± 0.02**	0.67 ± 0.02	0.66 ± 0.01
	Diameter Error [16]	**0.11 ± 0.00**	0.28 ± 0.00	0.28 ± 0.00	**0.08 ± 0.00**	0.19 ± 0.00	0.19 ± 0.00	**0.07 ± 0.00**	0.15 ± 0.00	0.14 ± 0.00
	DTW Distance	**234 ± 6**	264 ± 7	265 ± 6	**86 ± 3**	97 ± 1	96 ± 2	**48 ± 1**	55 ± 1	56 ± 1
San Joaquin [7]	QA Area AvRE	**2.0 ± 0.6**	2.3 ± 0.6	2.12 ± 0.60	**0.4 ± 0.1**	0.5 ± 0.2	0.6 ± 0.2	0.17 ± 0.07	**0.14 ± 0.04**	0.21 ± 0.06
	QA Pattern AvRE	**3.4 ± 0.5**	6.6 ± 0.5	6.2 ± 0.3	**1.1 ± 0.2**	3.1 ± 0.3	2.8 ± 0.4	**0.7 ± 0.1**	1.7 ± 0.2	1.5 ± 0.1
	Popularity KT [16]	**0.68 ± 0.01**	0.65 ± 0.02	0.65 ± 0.02	**0.81 ± 0.01**	0.80 ± 0.01	0.79 ± 0.01	**0.89 ± 0.01**	0.87 ± 0.01	0.87 ± 0.01
	Trip Error [16]	**0.65 ± 0.02**	0.81 ± 0.01	0.81 ± 0.01	**0.47 ± 0.01**	0.75 ± 0.01	0.76 ± 0.01	**0.39 ± 0.02**	0.66 ± 0.01	0.65 ± 0.01
	Diameter Error [16]	**0.09 ± 0.00**	0.28 ± 0.00	0.28 ± 0.00	**0.07 ± 0.00**	0.18 ± 0.00	0.18 ± 0.00	**0.05 ± 0.00**	0.14 ± 0.00	0.14 ± 0.00
	DTW Distance	**238 ± 8**	277 ± 6	271 ± 5	**97 ± 2**	107 ± 3	110 ± 4	**53 ± 2**	60 ± 1	59 ± 2

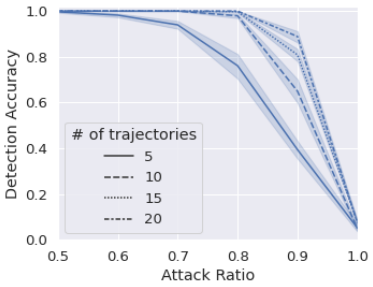


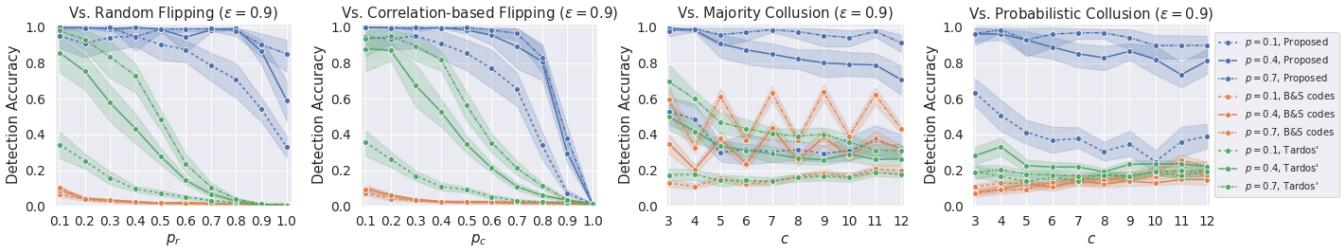
Figure 7: Fingerprint robustness against re-fingerprinting attacks on differentially private datasets of different sizes. The attack ratio denotes the fingerprint ratio that the attacker uses during the attack.

dataset when $\epsilon = 2.5$, the error for query answering on location points is 0.34, which is slightly higher than 0.33 for the Boneh-Shaw codes. Similarly, the performance of our scheme when $\epsilon = 0.9$ is 0.01 worse than the two existing methods for popularity analysis on the Oldenburg dataset. On the San Joaquin dataset, the query answering error on location points of our scheme is 0.17, while the error is 0.14 if the Boneh-Shaw codes are used. For the majority of the metrics, our scheme outperforms the Boneh-Shaw codes and the Tardos' code. For a few metrics, our scheme is slightly worse but still comparable with the existing methods.

Table 3: Execution time of generating a fingerprinted dataset ($n = 100$)

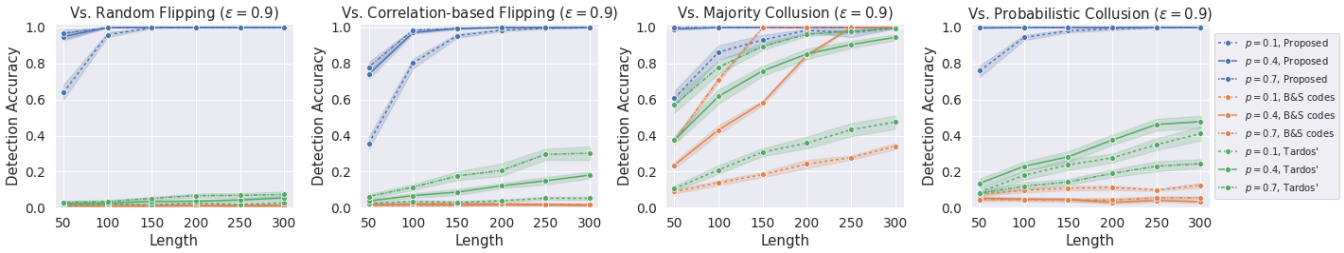
l	100	200	300	400	500
time(s)	0.5177	1.0229	1.5268	2.0407	2.5779

6.5.1 Computation Time. We present the computation time of the proposed scheme in Table 3. For a dataset of 100 trajectories with a length equal to 500, the proposed scheme only takes 2.5779 seconds to generate one fingerprinted copy. We observe that the computation time increases linearly with increasing trajectory length. In conclusion, our scheme shows practical time efficiency for fingerprint generation and scales well for large datasets.



(a) vs. RF (b) vs. CF (c) vs. MJR (d) vs. PROB

Figure 8: Fingerprint robustness on the differentially private Geolife [37] dataset protected by an alternative method (AdaTrace [16]) compared with two existing methods, i.e., Boneh-Shaw codes [5] and Tardos codes [31] with different fingerprint ratio p.



(a) vs. RF (b) vs. CF (c) vs. MJR (d) vs. PROB

Figure 9: Fingerprint robustness on the differentially private Geolife [37] dataset protected by PIM [33] with different lengths of leaked trajectories compared with two existing methods, i.e., Boneh-Shaw codes [5] and Tardos codes [31] with different fingerprint ratio p.

7 DISCUSSION

Here, we compare PIM with other differentially private mechanisms and discuss the correlation model.

7.1 Comparison Between PIM and Other Differentially Private Mechanisms

Compared with PIM, other existing works have more or less their limitations for realistic location dataset sharing. Jiang et al.'s approach [20] requires that the starting and finishing points of all the trajectories should be fixed, making it only work on specific types such as ship or flight trajectories. [16] and [18] need accurate statistical features of the input datasets. Thus, the size of the dataset should be comparably large. In other words, they cannot handle datasets with only a few trajectories. In addition, adding and removing trajectory is one of the most common requests from users as they become more concerned with their data privacy [6]. Synthetic methods [16, 18] cannot perform such operations simply by working on the protected dataset and they have to regenerate the entire dataset. Meanwhile, PIM is executed on each trajectory instead of on the whole data set. It can easily achieve this by adding or removing generated copies of a specific trajectory to/from the shared dataset.

7.2 Correlation Model

In this work, we use 2-gram Markov chain to model correlations. If we use a higher-order model, each pattern X's occurrence will decrease significantly since longer prefixes are harder to find intuitively. Therefore, we cannot collect enough patterns Xg to form a reliable transition distribution for a prefix X, which results in an inaccurate transition matrix. For instance, GeoLife dataset [37] consists of 17,621 trajectories in Beijing. However, we can hardly construct a reliable 3-gram model out of it, especially if we use a dense grid for services like Google Maps that collects location data frequently. Some approaches use a sparse grid to overcome this problem [9, 33] (around $400 * 400m^2$), but the location points are too general for analytical purposes. On the other hand, our target applications, e.g., Google Maps and outdoor exercises, cannot bear such general locations. As a result, we compromise with the 2-gram Markov chain.

8 CONCLUSION AND FUTURE WORK

In this paper, we design a system that achieves both privacy preservation and robust fingerprinting for location datasets. We first apply a differentially private mechanism to the dataset and then implement a fingerprinting scheme that considers pairwise correlations in the location data and prevents the attackers from unauthorized leakage of the dataset. With the integration of a utility-boosting post-processing, our proposed direction-sensitive fingerprinting scheme provides high data utility for data analyzers.

There are several directions for further research. First, we plan to improve our correlation model to a higher-order model (e.g., using road structures) and analyze the performance of the scheme. In addition, a non-uniform grid in discretization can be used and different types of collusion attacks can be defined and studied. Moreover, our approach provides differential privacy and fingerprint robustness in two separate steps. Combining those two steps is another potential future work.

REFERENCES

[1] 2023. *Doordash*. Retrieved June 13, 2023 from https://www.doordash.com/
[2] 2023. *Google Maps*. Retrieved June 13, 2023 from https://maps.google.com/
[3] Osman Abul, Francesco Bonchi, and Mirco Nanni. 2008. Never walk alone: Uncertainty for anonymity in moving objects databases. In *2008 IEEE 24th international conference on data engineering*. Ieee, 376–385.
[4] Miguel E Andrés, Nicolás E Bordenabe, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. 2013. Geo-indistinguishability: Differential privacy for location-based systems. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. 901–914.
[5] Dan Boneh and James Shaw. 1998. Collusion-secure fingerprinting for digital data. *IEEE Transactions on Information Theory* 44, 5 (1998), 1897–1905.
[6] Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. 2021. Machine unlearning. In *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 141–159.
[7] Thomas Brinkhoff. 2002. A framework for generating network-based moving objects. *GeoInformatica* 6, 2 (2002), 153–180.
[8] Yang Cao, Masatoshi Yoshikawa, Yonghui Xiao, and Li Xiong. 2017. Quantifying differential privacy under temporal correlations. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*. IEEE, 821–832.
[9] Rui Chen, Gergely Acs, and Claude Castelluccia. 2012. Differentially private sequential data publication via variable-length n-grams. In *Proceedings of the 2012 ACM conference on Computer and communications security*. 638–649.
[10] Minquan Cheng and Ying Miao. 2011. On anti-collusion codes and detection algorithms for multimedia fingerprinting. *IEEE transactions on information theory* 57, 7 (2011), 4843–4851.
[11] Yves-Alexandre De Montjoye, César A Hidalgo, Michel Verleysen, and Vincent D Blondel. 2013. Unique in the crowd: The privacy bounds of human mobility. *Scientific reports* 3, 1 (2013), 1–5.
[12] Cynthia Dwork. 2008. Differential privacy: A survey of results. In *International conference on theory and applications of models of computation*. Springer, 1–19.
[13] Cynthia Dwork, Aaron Roth, et al. 2014. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.* 9, 3-4 (2014), 211–407.
[14] Benjamin CM Fung, Ke Wang, Ada Wai-Chee Fu, and Jian Pei. 2008. Anonymity for continuous data publishing. In *Proceedings of the 11th international conference on Extending database technology: Advances in database technology*. 264–275.
[15] Mehmet Emre Gursoy, Ling Liu, Stacey Truex, and Lei Yu. 2018. Differentially private and utility preserving publication of trajectory data. *IEEE Transactions on Mobile Computing* 18, 10 (2018), 2315–2329.
[16] Mehmet Emre Gursoy, Ling Liu, Stacey Truex, Lei Yu, and Wenqi Wei. 2018. Utility-aware synthesis of differentially private and attack-resilient location traces. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*. 196–211.
[17] Moritz Hardt and Kunal Talwar. 2010. On the geometry of differential privacy. In *Proceedings of the forty-second ACM symposium on Theory of computing*. 705–714.
[18] Xi He, Graham Cormode, Ashwin Machanavajjhala, Cecilia M Procopiuc, and Divesh Srivastava. 2015. DPT: differentially private trajectory synthesis using hierarchical reference systems. *Proceedings of the VLDB Endowment* 8, 11 (2015), 1154–1165.
[19] Tianxi Ji, Emre Yilmaz, Erman Ayday, and Pan Li. 2021. The Curse of Correlations for Robust Fingerprinting of Relational Databases. In *24th International Symposium on Research in Attacks, Intrusions and Defenses*. 412–427.
[20] Kaifeng Jiang, Dongxu Shao, Stéphane Bressan, Thomas Kister, and Kian-Lee Tan. 2013. Publishing trajectories with differential privacy guarantees. In *Proceedings of the 25th International Conference on Scientific and Statistical Database Management*. 1–12.
[21] Julien Lafaye, David Gross-Amblard, Camelia Constantin, and Meryem Guerrouani. 2008. Watermill: An optimized fingerprinting system for databases under constraints. *IEEE Transactions on Knowledge and Data Engineering* 20, 4 (2008), 532–546.
[22] Yingjiu Li, Vipin Swarup, and Sushil Jajodia. 2005. Fingerprinting relational databases: Schemes and specialties. *IEEE Transactions on Dependable and Secure Computing* 2, 1 (2005), 34–45.
[23] Siyuan Liu, Shuhong Wang, Robert H Deng, and Weizhong Shao. 2004. A block oriented fingerprinting scheme in relational database. In *International conference on information security and cryptology*. Springer, 455–466.
[24] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. 2007. l-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 1, 1 (2007), 3–es.
[25] Luis Moreira-Matias, Joao Gama, Michel Ferreira, Joao Mendes-Moreira, and Luis Damas. 2013. Predicting taxi–passenger demand using streaming data. *IEEE Transactions on Intelligent Transportation Systems* 14, 3 (2013), 1393–1402.
[26] Mehmet Ercan Nergiz, Maurizio Atzori, and Yucel Saygin. 2008. Towards trajectory anonymization: a generalization-based approach. In *Proceedings of the SIGSPATIAL ACM GIS 2008 International Workshop on Security and Privacy in GIS and LBS*. 52–61.
[27] Mohammad Shokoohi-Yekta, Bing Hu, Hongxia Jin, Jun Wang, and Eamonn Keogh. 2017. Generalizing DTW to the multi-dimensional case requires an adaptive approach. *Data mining and knowledge discovery* 31, 1 (2017), 1–31.
[28] Reza Shokri. 2014. Privacy games: Optimal user-centric data obfuscation. *arXiv preprint arXiv:1402.3426* (2014).
[29] Reza Shokri, George Theodorakopoulos, Jean-Yves Le Boudec, and Jean-Pierre Hubaux. 2011. Quantifying location privacy. In *2011 IEEE symposium on security and privacy*. IEEE, 247–262.
[30] Latanya Sweeney. 2002. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 10, 05 (2002), 557–570.
[31] Gábor Tardos. 2008. Optimal probabilistic fingerprint codes. *Journal of the ACM (JACM)* 55, 2 (2008), 1–24.
[32] Min Wu, Wade Trappe, Z Jane Wang, and KJ Ray Liu. 2004. Collusion-resistant fingerprinting for multimedia. *IEEE Signal Processing Magazine* 21, 2 (2004), 15–27.
[33] Yonghui Xiao and Li Xiong. 2015. Protecting locations with differential privacy under temporal correlations. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. 1298–1309.
[34] Emre Yilmaz and Erman Ayday. 2020. Collusion-Resilient Probabilistic Fingerprinting Scheme for Correlated Data. *arXiv preprint arXiv:2001.09555* (2020).
[35] Lei Yu, Ling Liu, and Calton Pu. 2017. Dynamic Differential Location Privacy with Personalized Error Bounds.. In *NDSS*.
[36] Jun Zhang, Xiaokui Xiao, and Xing Xie. 2016. Privtree: A differentially private algorithm for hierarchical decompositions. In *Proceedings of the 2016 International Conference on Management of Data*. 155–170.
[37] Yu Zheng, Xing Xie, Wei-Ying Ma, et al. 2010. GeoLife: A collaborative social networking service among user, location and trajectory. *IEEE Data Eng. Bull.* 33, 2 (2010), 32–39.

Appendices

A PROOF OF IMMUNITY TO POST-PROCESSING OF OUR SCHEME

Claim. *The utility-focused post-processing scheme and the direction-sensitive fingerprinting scheme do not violate differential privacy.*

Proof. First, we assume that the input data are released under an arbitrary privacy-preserving mechanism that satisfies ϵ-differential privacy. According to Proposition 3.2, any mapping function does not violate the guarantee of a differentially private method if the function does not utilize the values of the original dataset. During utility-focused post-processing in Section 5.2 and direction-sensitive fingerprinting in Section 5.3.2, we only use public correlations instead of any information from the original dataset, which satisfies the conditions of immunity to post-processing. Thus, post-processing and fingerprinting do not violate ϵ-differential privacy that is provided by the privacy-preserving mechanism.

□

B EXISTING FINGERPRINTING SCHEMES

In this section, we introduce the two existing fingerprinting schemes that we use for comparison.

B.1 The Boneh-Shaw Codes

The Boneh-Shaw codes [5] are collusion-resistant fingerprinting codes. It captures one of the colluding parties with only a probability ω of incorrect accusation with c service providers colluding (ω-secure) under the marking assumption [5]. $\Gamma(n, d)$-codes serve n users and consist of $n * d$ digits. Each user $i \in [1, n]$ gets the first $(i-1) * d$ bits as 1's and the rest $(n-i) * d$ as 0's. An example of the $\Gamma(4, 3)$ code is: {111-111-111, 000-111-111, 000-000-111, 000-000-000}, and each user receives one of the codewords. By identifying the first block with a majority of 1's in the leaked data, e.g., the i-th block, the algorithm considers the user i guilty. In the above

example, if 001-011-111 is leaked, the 2nd user who owns 000-111-111 will be accused of leaking the data since the 2nd block is the first block with a majority of 1's.

B.2 The Tardos Codes

The Tardos codes [31] are another binary fingerprinting technique under the marking assumption. The codes utilize randomization in construction and provide similar security against majority collusion attacks while requiring a shorter code length than the Boneh-Shaw codes. The construction of Tardos codes requires the number of sharings n, the number of colluding units c, and the expected security ω. The minimum length of binary code to ensure ω-security is $m = 100c^2k$, where $k = \lceil \log(1/\omega) \rceil$. Let $t = 1/(300c)$ and $sin^2 t' = t, 0 < t' < \pi/4$. p_i denotes the probability of 1 at position i, i.e., $Pr(X_i = 1) = p_i$, and is calculated independently. To select the probability for each position i, we sample $r_i \in [t', \pi/2 - t']$ uniformly and then acquire $p_i = sin^2 r_i$. Let X_{ji} denote the i-th digit of the user j and $\mathcal{Y} = \{y_1, y_2, \ldots, y_m\}$ denote the leaked data. While accusing the colluders, the codes use a scoring function as

$$U_{ji} = \begin{cases} \sqrt{\frac{1-p_i}{p_i}} & \text{if } X_{ji} = 1 \\ -\sqrt{\frac{p_i}{1-p_i}} & \text{if } X_{ji} = 0 \end{cases} \tag{2}$$

and accuse the user j if

$$\sum_{i=1}^{m} y_i U_{ji} \geq 20ck$$

.

C DECENTRALIZED SETTING

We build our system in the centralized setting, i.e., users' location points are collected by a centralized data server (service provider) and then processed by our scheme. This relies on an honest party involved in the system, since the centralized data server (i.e., the service provider) has direct access to the collected dataset. If no such party exists, we can alternatively set up a decentralized system, where the privacy is protected before sending location data to the centralized server. In such a decentralized setting, users can apply DP protection locally on their devices by setting the desired privacy level they want to achieve. The protected data are then transmitted to the centralized server. Every time the service provider collects real-time location information from the users, users immediately protect their locations under differential privacy (e.g., using PIM) and send the noisy locations to the centralized server. The server collects these locations sequentially and applies our proposed fingerprinting scheme to the location data. In this case, real locations are not exposed to any party, including the centralized server, thus protecting users' location privacy in a better way. However, this setting sacrifices users' experience while using location-based servers, and thus some service providers may offer poor services due to the inaccuracy of the location information. While using Google Maps for navigation, one does not want to report incorrect locations. But if one uses Google Maps to find nearby restaurants, they often accept a vague or slightly deviated localization. Service providers can choose either setting based on the services they provide.

D TRAJECTORY POST-PROCESSING SCHEME

Algorithm 2 shows the steps of the post-processing scheme described in Section 5.2, where $\|\cdot\|_2$ denotes the l_2-norm.

Algorithm 2: Trajectory Post-Processing Scheme

input : Noisy trajectory $\hat{\mathcal{X}} = [\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_m]$, location alphabet $\mathcal{G}$, conditional probability in the correlations $Pr(x_j|x_{j-1})$ for any $j \in [1, m]$, probability threshold τ

output: Smoothed trajectory $\mathcal{X}^* = [x_1^*, x_2^*, \ldots, x_m^*]$

$x_1^* \leftarrow x_1$
forall $j \in \{2, 3, \ldots, m\}$ **do**
  $prob_\tau(x_{j-1}^*) \leftarrow \tau$-probable set of x_{j-1}^*;
  $x_{closest} \leftarrow$ closet point to $\hat{x}_j$ in $prob_\tau(x_{j-1}^*)$;
  if $\hat{x}_j \notin prob_\tau(x_{j-1}^*)$ **then**
    if $\|x_{j-1}^*, \hat{x}_j\|_2 \leq \|x_{j-1}^*, x_{closest}\|_2$ **then**
      $x_j^* \leftarrow \hat{x}_j$
    else
      $x_j^* \leftarrow x_{closest}$
  else
    $x_j^* \leftarrow \hat{x}_j$
end

E ADDITIONAL EXPERIMENTAL RESULTS

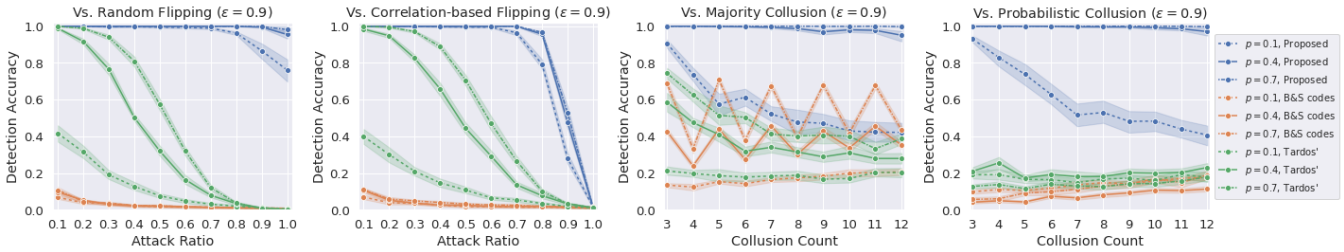
In this section, we show additional experimental results. First, we evaluate fingerprint robustness on other datasets apart from GeoLife [37]. Then we show how length impacts fingerprinting performance. Also, we extend Section 6.4.1 and evaluate our scheme's performance when multiple trajectories are leaked.

E.1 Fingerprinting Robustness on Other Datasets Under Differential Privacy

As is shown in Figure 10, 11, and 12, the results are almost identical to GeoLife [37]'s (in Section 6.4.1. It proves that our fingerprinting scheme is robust and consistent for all location datasets.

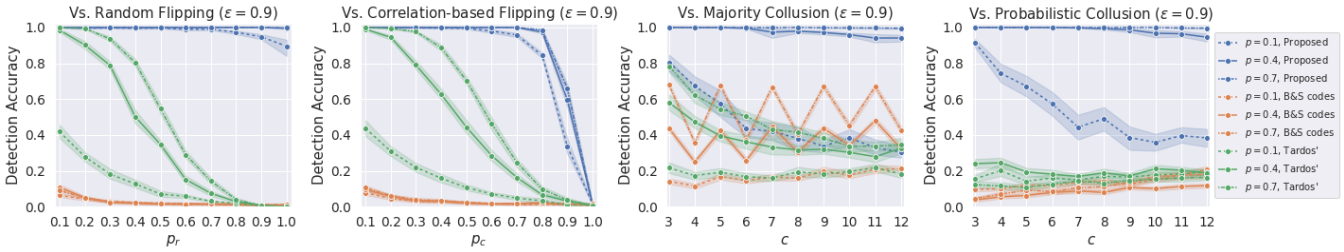
E.1.1 Fingerprint Robustness on Datasets Without Differential Privacy. We evaluate the fingerprint robustness of our proposed scheme without privacy-preserving mechanisms. Figure 13 shows the performance of the proposed scheme against multiple attacks. For random flipping attacks, our scheme achieves almost 100% accuracy if the attacker does not perturb more than 60% of the location points, and it decreases to 90% if the attacker distorts 80% of the location points. In terms of correlation-based flipping attacks, the scheme has high accuracy when the flipping ratio p_c is less than or equal to 0.6, and the accuracy drops significantly for larger p_c. The reason is almost the same as why the probabilistic fingerprinting scheme (PFS) [34] does not work on location datasets, i.e., the forced deviation (shown in Figure 4). For an acceptable data utility, the attacker does not prefer a large p_c in practice. For majority collusion attacks, the detection accuracy of our scheme is greater than 80%. In terms of probabilistic collusion attacks, our scheme achieves 99% detection accuracy if $c = 3$ and still gets around 60% if c increases to 12.

Note that the scheme does not benefit from higher fingerprint ratio against two correlation-based attacks (i.e., correlation-based



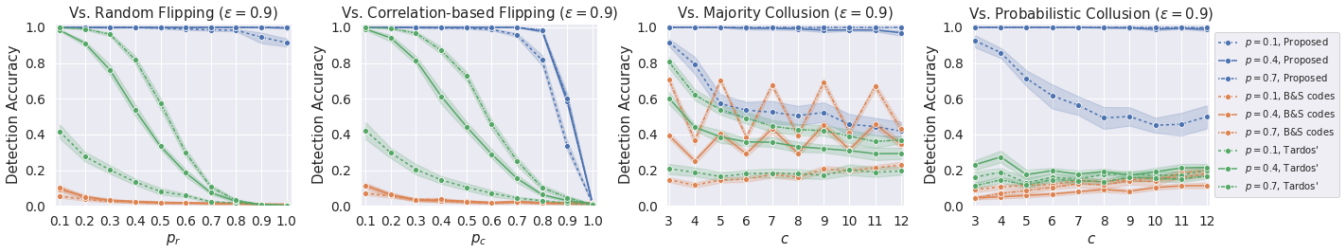
(a) vs. RF (b) vs. CF (c) vs. MJR (d) vs. PROB

Figure 10: Fingerprint robustness of the proposed scheme on the differentially private Taxi [25] dataset by PIM [33] compared with two existing methods, i.e., Boneh-Shaw codes [5] and Tardos codes [31], with different fingerprint ratio p.



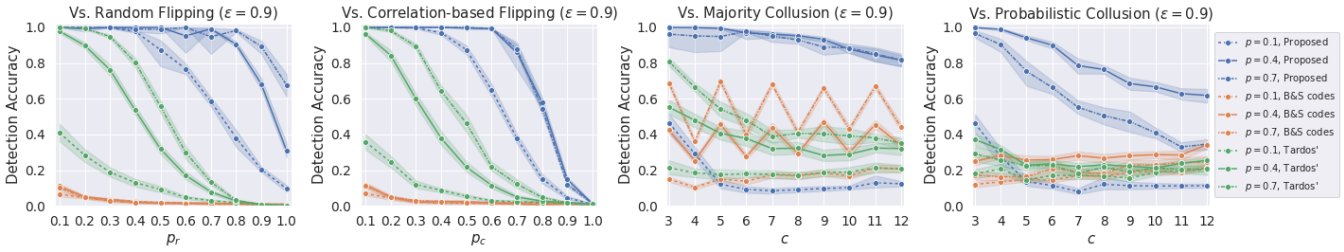
(a) vs. RF (b) vs. CF (c) vs. MJR (d) vs. PROB

Figure 11: Fingerprint robustness of the proposed scheme on the differentially private OldenBurg [7] dataset by PIM [33] compared with two existing methods, i.e., Boneh-Shaw codes [5] and Tardos codes [31], with different fingerprint ratio p.



(a) vs. RF (b) vs. CF (c) vs. MJR (d) vs. PROB

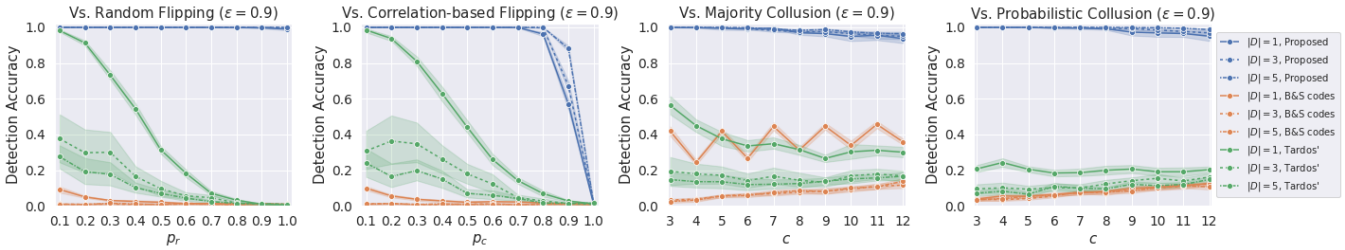
Figure 12: Fingerprint robustness of the proposed scheme on the differentially private Joaquin [37] dataset by PIM [33] compared with two existing methods, i.e., Boneh-Shaw codes [5] and Tardos codes [31], with different fingerprint ratio p.



(a) vs. RF (b) vs. CF (c) vs. MJR (d) vs. PROB

Figure 13: Fingerprint robustness of the proposed scheme on the non-differentially private dataset (the GeoLife dataset [37]) compared with two existing methods, i.e., Boneh-Shaw codes [5] and Tardos codes [31], with different fingerprint ratio p.

flipping attacks and probabilistic collusion attacks) for non-noisy datasets, and the accuracy becomes even worse for probabilistic collusion attacks, which can be explained as follows. In a non-noisy trajectory, pairwise correlations mostly hold, i.e., the transition probability from the previous point to the current point remains high. In that case, if we fingerprint (modify) two consecutive points, the pairwise correlation between the modified values mostly decreases. When fingerprint ratio, i.e., probability of a point being fingerprinted (perturbed), is high (i.e., $p > 0.5$), such scenarios occur more commonly and can be exploited by the attacker, resulting



(a) vs. RF **(b) vs. CF** **(c) vs. MJR** **(d) vs. PROB**

Figure 14: Fingerprint robustness on the differentially private Geolife [37] dataset protected by PIM [33] with different number of trajectories in the leaked dataset compared with two existing methods, i.e., Boneh-Shaw codes [5] and Tardos codes [31], with different fingerprint ratio p.

in a decrease in detection accuracy. On the other hand, when the fingerprint ratio is low, e.g., $p = 0.1$, our scheme does not have too few fingerprinted points to provide fingerprint robustness against collusion attacks, resulting in a degradation in accuracy. This ensures high data utility and high fingerprint robustness in the shared dataset simultaneously.

Compared to our scheme, the two existing methods are not equally robust. Boneh-Shaw codes achieve around 50% detection accuracy in majority collusion attacks and have at most 20% chance to identify the attacker against other attacks, where the wavy style in Figure 13c results from its own design. The detection accuracy of the Tardos codes is 100% against the two flipping attacks if the flipping ratio is 0.1, but it quickly drops to below 40% and 20% for random flipping attacks and correlation-based flipping attacks, respectively. For collusion attacks, their detection accuracy is at most 70% if 3 attackers collude and 40% when 12 are involved. Overall, our scheme achieves better performance against all the attacks considered.

E.2 Fingerprinting Robustness on Differentially Private Datasets While Multiple Trajectories are Leaked

Figure 14 shows the fingerprint robustness on differentially private datasets while multiple trajectories are leaked.