Differentially Private Fingerprinting for Location Trajectories

Yuzhou Jiang  
yxj466@case.edu  
Case Western Reserve University

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

Erman Ayday  
exa208@case.edu  
Case Western Reserve University

ABSTRACT
Location-based services have brought significant convenience to people in their daily lives. Services like navigation, food delivery, and carpooling frequently ask for location data from users. On the other side, researchers and businesses are eager to acquire those data (that is collected by location-based service providers) for various purposes. However, directly releasing those data causes 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 solve this, we propose a system that protects users’ location data under differential privacy and prevents unauthorized redistribution at the same time. Observing high amount of noise introduced to achieve differential privacy, we implement a novel post-processing scheme to regain data utility. In addition, we also propose a novel fingerprinting scheme as a part of the post-processing (to detect unauthorized redistribution of data). Our proposed fingerprinting scheme considers correlations in location datasets and collusions among multiple parties, which makes it hard for the attackers to infer the fingerprinting codes and avoid accusation. Using the experiments on a real-life location dataset, we show that our system achieves high fingerprint robustness against state-of-the-art attacks. We also show the integrated fingerprinting scheme increases data utility for differentially private datasets, which is beneficial for data analysts in data mining.

1 INTRODUCTION
The rapid evolution of the internet and mobile technologies bring unprecedented opportunities to location-based applications. Most of such applications heavily rely on location-based information to ensure that services function correctly. For instance, Doordash [1] collects users’ approximate locations and displays nearby restaurants available for delivery/pickup. As a navigation application, Google Maps [2] uses real-time locations to determine optimal routes. Most individuals are subtly accustomed to the convenience of lifestyles using those location-based services, and hence they share their location patterns with such location-based service providers voluntarily (with consent).

On the other hand, location datasets (including check-ins or trajectories of individuals) that are constructed by location-based service providers bring vast benefits. Researchers can use such location datasets for various data mining tasks. Businesses can also utilize such location datasets in different ways, such as boosting user experience, adapting marketing strategies, or selecting optimal location for new facilities. However, location data, i.e., movement records combined with metadata, contains sensitive information, which leads to significant privacy concerns if service providers share the datasets directly with researchers or businesses. Service providers often share such location datasets with a limited number of parties, called data analyzers, after applying privacy-preserving data sharing techniques. Nevertheless, malicious data analyzers, motivated by profit, may leak their copies to unauthorized parties. To resolve this issue, service providers can embed a unique fingerprint in each shared copy of the original dataset. By utilizing the embedded fingerprint in a leaked dataset, service providers can identify the source of the unauthorized redistribution with high confidence. Thus, knowing that the leaked dataset will be traced back to them, malicious parties become less motivated to leak the copies of received datasets.

However, traditional digital fingerprinting schemes cannot be directly applied to the location datasets due to specific correlations in location datasets and their particular utility requirements. Typically, neighboring location points are highly correlated along the trajectory in several aspects. For instance, in a walking trajectory, recorded at every 10 seconds, it is not likely to have two contiguous location points one kilometer apart from each other. Also, one can precisely estimate/infer a given point in a location trajectory given the previous and following points. Therefore, 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. The attacker can then distort or remove such identified data points (i.e., distort the fingerprint), which makes it harder for the service providers to detect the source of a leaked dataset. To mitigate such vulnerabilities, a service provider may increase the number of fingerprinted data points in the shared dataset; however, this results in significant degradation in dataset utility. Therefore, a new, robust, and utility-preserving fingerprinting scheme is required to protect location datasets.

Fingerprinting only protects the dataset from redistribution. To protect the privacy of dataset participants’ (users’) location data, service providers still need to apply privacy-preserving techniques before sharing the dataset with data analyzers. It has been shown that users’ identities can be deanonymized with high confidence, given only a pattern of four location points [9]. Therefore, simple anonymization is not sufficient to provide privacy for the dataset participants. Differential privacy (DP) is a state-of-the-art concept for privacy preservation that quantifies and limits the information acquired from the attackers’ perspective. There exist DP-based techniques [14, 29] that aim to protect location datasets. However, to achieve DP, those schemes introduce too much noise, distorting the internal relations between points and losing useful features in the shared datasets. Also, these solutions do not provide liability guarantees (via fingerprinting) against dataset leakage (unauthorized redistribution). It is possible to achieve privacy protection together with robust fingerprinting in an ad-hoc way (e.g., achieving DP first and then fingerprinting the dataset), but, this ad-hoc approach results in more degradation in the utility of the shared dataset. Hence, new methods are in demand to provide both privacy and liability while preserving the utility of the shared location datasets.

To tackle these challenges, we introduce a framework that simultaneously provides robust fingerprinting, privacy preservation, and high utility when sharing location datasets. We first apply an existing privacy-preserving technique, i.e., the planar isotropic
mechanism (PIM) \cite{29} to the trajectories in the dataset. We model
the adversarial knowledge using a Hidden Markov Model. Instead of
hiding a location point among the entire area where the user possi-
bly occurs, we only protect it among highly probable regions, called
\( \delta \)-location set based on prior knowledge and auxiliary informa-
tion, i.e., a public correlation model. To solve the utility issue, we propose
a smoothing scheme as the post-processing step of the differentially
private mechanism and restore most correlations between consecutive
points. During this process, we check the 2-gram transitions in the
trajectory and replace each location that has a low probability
with a highly probable one by considering the directional informa-
tion of the transition. We also develop a robust fingerprinting
scheme using probabilistic sampling on location datasets and in-
tegrate it as a part of the aforementioned post-processing scheme
to minimize the utility degradation in the shared dataset. The pro-
posed scheme ensures that an attacker cannot avoid accusation (by
the service provider) by distorting the fingerprint even if using
publicly known correlations or colluding with other parties.

We implement our proposed scheme using a real-life dataset \cite{33}
and evaluate the fingerprint robustness against various attacks by
measuring the accuracy of the fingerprint detection algorithm. To
compare its detection accuracy, we implement a randomized ap-
proach that applies multivariate Laplacian noise to fingerprint the
data points. We also implement two baseline collusion resistant
schemes, i.e., the Tardos codes and Boneh-Shaw codes, to evalu-
ate the performance against collusion attacks. We show that our
proposed scheme achieves higher detection accuracy than all these
approaches against correlated distortion attacks and collusion at-
tacks. We also evaluate the data utility in terms of correlation fit
rate, accuracy of answering queries, and trajectory similarity. We
observe that our scheme provides significantly better utility than
the existing approaches. The main contributions of this work can
be summarized as follows:

- We design a system that provides both differential privacy
  and robust fingerprinting when sharing location trajectories.
- We propose a probabilistic fingerprinting scheme for loca-
tion trajectories that utilizes publicly known correlations to
counter various attacks against a fingerprinting scheme.
- We propose a scheme that restores the pairwise correlations
  in the trajectories as a post-processing step of differential pri-
  vacy, and then combine this post-processing scheme into the
  proposed fingerprinting scheme to minimize the degradation
  in the utility of the shared datasets.
- We evaluate the fingerprint robustness and data utility of
  our proposed scheme compared to the baseline approaches
  on a real-life location dataset.

The rest of the paper is organized as follows. We review the
existing works in Section 2 and provide the preliminaries in Sec-
tion 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. Section 7 concludes the paper.

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

2.1 Location Privacy
Location data contain sensitive information such as moving pat-
terns and preferred locations. Traditional privacy enhancing tech-
niques, e.g., k-anonymity \cite{26} and l-diversity \cite{22}, have been adapted
to the location setting. However, for a location dataset, those tech-
niques have their limitations in dealing with data streams with
various lengths. Some works \cite{3, 13} split the trajectories into equal-
length fragments and achieve privacy on the fragment, which is
not sufficient for privacy protection on trajectories. Differential
privacy \cite{10} as a popular privacy definition has also been
used to protect location datasets in recent years \cite{6, 24, 25, 31}.
Geo-indistinguishability \cite{4} defines a variant of differential pri-
vacy based on the distance between the points of interests, but it
only works on location points instead of trajectories. Several meth-
ods \cite{7, 14, 16} provide differential privacy to the statistics extracted
from original location datasets. Some of those works \cite{14, 16} also
generate a synthetic dataset instead of the statistics. Those works
completely eliminate moving features of any specific user while pre-
serving statistics, which significantly decreases the usability of the
dataset from certain services, e.g., map navigation and carpooling.
\cite{18} releases differentially private trajectories by sampling and in-
terpolating them. Still, the scheme has an additional restriction that
the start and end locations are revealed in advance. \cite{29} generates
perturbed trajectories by considering point-wise correlations and
prior knowledge from previous released points. The method fits our
objective and provides enough privacy protection to trajectories.
We use it to achieve privacy guarantees for location datasets and
provide the details in Section 5.1.

2.2 Digital Fingerprinting
Digital fingerprinting embeds a unique identifier, e.g., a sequence
of marks, to the data by adding, removing or editing partial values
of the data. Several works have been proposed to enable digital
fingerprinting for data distribution \cite{5, 8, 27}. Boneh and Shaw de-
design a fingerprint code and prevent the receivers from colluding \cite{5}.
Tardos et al. propose a probability-based fingerprinting scheme that
can catch all suspicious individuals simultaneously \cite{27}. 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 the data
with various types, i.e., relational databases \cite{17, 20, 21, 28} or mul-
timedia \cite{19}. These approaches are not applicable as well since
 trajectories in location datasets contain high correlations between
neighboring location points, which is a different setting. \cite{30} intro-
duces a fingerprinting scheme for sequential data that considers
correlations between data points. However, it only works when the
states are limited and inter-transitible. We improve this work and
implement it in our system.

3 PRELIMINARIES
In this section, we first introduce the definition of differential pri-
vacy 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). [10] For any neighboring datasets $D, D'$ that only differ in one data record, a randomized algorithm $\mathcal{M}$ satisfies $\epsilon$-differential privacy if for all possible outputs $S \subseteq \text{Range}(\mathcal{M})$

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

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). [11] Let $\mathcal{M}$ be a randomized algorithm that is $(\epsilon, \delta)$-differentially private. For any arbitrary randomized mapping $f : \mathbb{R}^q \rightarrow \mathbb{R}'$ where $p, q \in \mathbb{N}^+$, $f \circ \mathcal{M}$ is $(\epsilon, \delta)$-differentially private.

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

3.2 Planar Isotropic Mechanism (PIM)
The planar isotropic mechanism [29] provides differential privacy guarantee to the location points in a trajectory. It builds the attacker’s knowledge using a Markov chain according to the previously released points in the trajectory and constructs a set called $\delta$-location set that contains the highly probable locations. Based on the location set, the method generates a noisy location at each timestamp using the simplified $k$-norm mechanism [15], which is done in an isotropic space. To better define the sensitivity, the mechanism designs a sensitivity metric, namely sensitivity hull, which is stricter than the $l_1$ norm in the location setting. A brief description of the mechanism is given in Appendix A.

3.3 The Boneh-Shaw Codes
The Boneh-Shaw codes [5] is a kind of collusion-resistant fingerprinting binary codes. It catches one of the colluders with only a probability $\omega$ of incorrect accusation with $c$ service providers colluding ($\omega$-secure) under the marking assumption [5], $\Gamma(n, d)$-codes serve $n$ users and consist of $n \times d$ digits. Each user $i \in [1, n]$ gets the first $(i - 1) \times d$ bits as 1’s and the rest $(n - i) \times d$ 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 $i$ as 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.

3.4 The Tardos Codes
The Tardos codes [27] are another binary fingerprint technique under the marking assumption. The codes utilize randomization in construction and provide similar security against colluding attacks while requiring a shorter code length than the Boneh-Shaw codes.

The construction of the Tardos codes requires the number of sharings $n$, the number of colluding units $c$, and the expected security $\omega$. The minimal binary code length to ensure $\omega$-security is $m = 100c^2k$, where $k = \lceil \log(1/\omega) \rceil$. Let $t = 1/(300c)$ and $\sin^2t' = t, 0 < t < \pi/4$. $p_i$ denotes the probability of 1 at position $i$, i.e., $Pr(X_i = 1) = p_i$, and is independently calculated. To select the probability for each position $i$, we sample $r_i \in [t', \pi/2 - t']$ uniformly and then acquire $p_i = \sin^2r_i$. Let $X_{ji}$ denotes the $i$-th digit of the user $j$ and $\mathcal{Y} = \{y_1, y_2, ..., y_m\}$ is the leaked data. While accusing the colluders, the codes use a scoring function as

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

and accuse the user $j$ if $\sum_{i=1}^{m} y_i U_{ji} \geq 20c_k$.

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 symbols 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 $\mathcal{X} = [x_1, x_2, ..., x_{|\mathcal{X}|}]$ 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 pre-process the trajectories to share the same time interval and omit the timestamps. Although some secondary metadata can occur such as velocities and directions, we leave these to future work.

4.1.2 Map Discretization. In location settings, a map area is often discretized into cells for simplicity [7, 14, 16, 29]. Following those works, we divide the continuous two dimensional space using a uniform grid of $N_{lat} \times N_{lng}$. Throughout the rest of the paper, we still use “points” to represent a discretized area for generalization.

4.1.3 Correlations. We build our correlations using the Markov chain. For each location $y \in \mathcal{X}$, the transition probability is modeled as $Pr[X_{g} | \mathcal{X}]$, where $X$ is a prefix sequence of $|\mathcal{X}|$ points and can be empty. We use the 2-gram model in our scheme ($|\mathcal{X}| = 1$). We provide a discussion about the correlation model in Appendix B.1.

4.2 System Model
The system workflow is shown in Figure 1. There are two parties in our setting: the service provider and the data analyzers. A 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 in their data server and is willing to share them with other parties. Meanwhile, researchers and businesses, categorized as data analyzers, want to access such location datasets. As discussed, releasing location data may raise privacy concerns. Therefore, the service provider aims to ensure users’ location privacy before sharing. More
specifically, it applies 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 users’ privacy intact simultaneously, we propose post-processing techniques at the service provider to regain partial data utility.

In this section, we introduce the threat model considering the parties in our system. The service provider is the only entity that accesses raw data from the users. Thus, we assume the service provider is honest (i.e., it does not distribute unauthorized copies of users’ data to other unauthorized parties). We discuss the practicality of a decentralized setting in Appendix B.2. However, in this work, since we focus on the privacy and unauthorized redistribution of the location datasets that are shared by a service provider, we do not consider that part. Nevertheless, what we proposed can be easily extended to provide privacy of users’ data during the sharing process with the service provider.

The analyzers are considered to be potentially malicious. The analyzers understand that the dataset has been perturbed to protect the privacy of dataset participants, but they may be curious about the original (non-perturbed) data values in the dataset. For this, they can utilize auxiliary information from public sources, e.g., the correlations in the map area of interest. They build their knowledge along each trajectory using a hidden Markov model, where the original values along the trajectory are states and the reported values at those positions are the observations (details are in Section 5.1. With the help of those auxiliary information, they can analyze the received trajectories and try to infer the true location points.

From the perspective of fingerprinting, a malicious analyzer (i.e., 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, without being accused. To avoid being tracked, the analyzer tries to distort the fingerprint signature. They can exploit the public correlations, collude with other analyzers, or even use both to hide their identity. In the rest of the section, we discuss all the attacks the analyzers can perform against the proposed fingerprinting scheme.

4.3.1 Random Distortion Attack. The random distortion attack is the baseline attack in which the attacker distorts the location points in the trajectory in order to distort the fingerprint. For each location point in the trajectory, the attacker adds a 2-dimensional multivariate Laplacian noise with a probability $p_r$.

4.3.2 Correlated Distortion Attack. The attacker can utilize the public correlations to improve the baseline distortion. This attack was first introduced in [30]. In this attack, the attacker analyzes the correlations between consecutive points along the trajectory from the start to the end. It checks the 2-gram transition from the previous point to the current one, i.e., $Pr(x_j|x_{j-1})$ at location $j$. If the transition probability is lower than a threshold $\tau$, the attacker considers the point is fingerprinted with high probability. The attacker decides to distort the point with a probability $p_c$. The attacker first constructs a set containing 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 $\tau$. 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 thus 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 makes the trajectory lose some fingerprint bits, which may mislead the fingerprint detection mechanism and result in accusing an innocent party.
4.3.4 Probabilistic Collusion Attack [30]. Similar to the correlated distortion attack, the correlated collusion attack [30] exploits the auxiliary information. The attackers share the datasets and analyze them using correlations, i.e., the transition probabilities. They also set a probability $p_k$ to approximate the actual fingerprinting probability $p$. Suppose 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 the 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 the probabilistic sampling, where the probability is proportional to $(1-p_e)c_{j,k} \cdot \left(\frac{p_e}{|G|-1}\right)^{n-c_{j,k}} \cdot Pr(x_j = g_k | x_{j-1} = x'_{j-1})$, where $G_j$ refers to the alphabet at position $j$. The first part $(1-p_e)c_{j,k} \cdot \left(\frac{p_e}{|G|-1}\right)^{n-c_{j,k}}$ is the probability of $g_k$ being the original location at position $j$ based on the assumed probability $p_e$, and the latter 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 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 $j$.

5 METHODOLOGY

Here, we introduce the details of our solution to achieve both differential privacy and robust fingerprinting while sharing location trajectories. For example, the method in [18] requires that the starting and finishing points of all the trajectories should be fixed, which only works on specific types such as ship or flight trajectories. [14] and [16] require a dataset as input and output a synthetic one that only preserves selected statistics instead of trajectories. In addition, trajectory addition and removal is one of the most common requests from the users as they become more concerned about their data privacy. However, [14] and [16] cannot handle those operations. Meanwhile, PIM is executed on each trajectory and shares a noisy copy of the input trajectory. Among the dataset constructed by the output of PIM, trajectories can be selectively inserted or deleted. PIM is parameterized by the privacy budget $\epsilon$. When $\epsilon$ is small, the points are mostly deviated from the actual locations to location points. When $\epsilon$ is large, the output trajectory looks like the original one. Using PIM, we construct the attacker’s knowledge using the hidden Markov model at each timestamp and hide the actual position among all the highly probable points. Given the noisy trajectory and prior knowledge, the attackers cannot reveal a user’s actual position at any timestamp. We note that this scheme is not our contribution; we only use PIM to achieve differential privacy of location points in trajectories. More details can be found in [29].

However, PIM still suffers from high data utility loss. While generating each location point along a trajectory, PIM introduces noise and makes the output location far from the actual one without considering the moving trend of the trajectory, which results in zigzag lines occurring everywhere in the trajectory. Zigzag patterns break regional correlations in that area, and they are not realistic for human and automobile. Also, due to the introduced noise, the distance between two contiguous released points varies significantly and may be much larger than it could be concerning the maximum possible velocity of the moving object. The two factors above finally exhaust data utility and make the shared trajectories useless for the analyzers. To solve this problem, we utilize the auxiliary information that is also used in PIM and from public sources to boost data utility of the released trajectory data.

5.2 Utility-Focused Post-Processing

Similar to other perturbation-based approaches that ensure event-level differential privacy, PIM generates high amount of noise for each location point, leading to significant utility loss in the shared location dataset. Since there are no solid constraints for the neighboring locations in the released trajectory that guarantee the moving patterns are realistic, the pairwise correlations inside are mostly very low for common $\epsilon$s. Influenced by the above two factors, the data utility of the whole trajectory drops 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. Our task is to generate a usable trajectory from the noisy release and use it as the true value in fingerprinting.

To solve this problem, we propose a utility-focused post-processing scheme to improve the dataset’s utility. We start with the definition of the $\tau$-probable set in Definition 5.1. We build the correlations using the 2-gram Markov chain and consider the transition based on the previous location in the trajectory.

Definition 5.1 ($\tau$-Probable Set). Let $\tau \in [0, 1]$ and $G$ be the map area. Given the location $g^*$, for any $j \in \mathcal{Z}^*$, the $\tau$-probable set of $g^*$ is defined as $\text{prob}_\tau(g^*) = \{g | Pr[x_{j+1} = g | x_j = g^*] \geq \tau\}$, $g \in G$.

In the post-processing scheme, we iterate the location points in the trajectory in sequential order. For each location point $\hat{x}_j$, we calculate the $\tau$-probable set of the location $x'_{j-1}$ generated from the post-processing scheme. If $\hat{x}_j$ is not in $\text{prob}_\tau(x'_{j-1})$, i.e., $Pr[x_{j} = \hat{x}_j | x_{j-1} = x'_{j-1}] \geq \tau$, the correlations are preserved between the two points, and we do not need to perform the utility boost. Otherwise, if $Pr[x_{j} = \hat{x}_j | x_{j-1} = x'_{j-1}] < \tau$, the correlations barely exist. In this case, we replace $x'_j$ with a new one from our
selection. Among the $\tau$-probable set, we generally choose the one closest to $\hat{x}_j$ as the final output. The new point $x_j^\ast$ is treated as the true value during the following fingerprinting process.

![Diagram](Figure 3: Pit falling. $x_{j-1}^\ast$ is the last smoothed point. The following outputs $x_j^\ast, x_{j+1}^\ast, \ldots$ will stay at the same position as $x_j^\ast$, forming a pit.)

Notice that selecting the points into the $\tau$-probable set depends on the transition probability. Thus, it is not guaranteed that the $\tau$-probable set is a circle-like shape covering all the directions of the previous location $\hat{x}$. Due to the insufficiency of the correlations generated from the publicly available datasets, in some extreme cases, there exists no suitable location in the set getting close to $\hat{x}_j$ compared with the previous location $x_{j-1}^\ast$. 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. $x_{j-1}^\ast$ is the smoothed location at the $(j-1)$-th position, and the $\tau$-probable set of it is marked using the dashed circle. Since the closest location is identical to the previous release $x_{j-1}^\ast$, the algorithm still reports $x_j^\ast = x_{j-1}^\ast$. Then, the following locations will still be in the same position as $x_{j-1}^\ast$, making the trajectory fall into a pit. Our solution is to let $x_j^\ast = \hat{x}_j$ in this case. By doing so, we force the scheme to jump out of the pit while 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 C.

According to Definition 3.2, any post-processing on the differentially private outputs do not violate the differential privacy guarantee if the actual values are unknown. The scheme modifies each location point only based on auxiliary information, i.e., the correlation model, without considering the true values of the data points before the differentially private generation. Thus, the post-processing scheme satisfies the property of immunity to post-processing and does not violate the DP guarantee of PIM.

5.3 Robust Fingerprinting

Traditional fingerprinting approaches are not robust against the attacker who inspects the correlations in the fingerprinted data. In order to mitigate such attacks, we use the probabilistic fingerprinting scheme (PFS) from [30]. The probabilistic fingerprinting scheme introduces correlations into traditional fingerprinting, and it is able to protect the generated fingerprints from correlation-based distortions. [30] focuses on pair-wise correlations and preserves them while fingerprinting, which fits our needs exactly. Nevertheless, PFS only works on datasets with limited states for each data point (e.g., 3 states in [30]), while location datasets have infinite states in reality. Thus, we are unable to apply PFS directly to location datasets. We redesign the sampling process in PFS to solve this issue. In the remaining paragraphs of this section, we first illustrate how PFS works. After that, we identify the problem of using PFS in the location setting and propose our solution.

5.3.1 The Probabilistic Fingerprinting Scheme

PFS embeds the fingerprint codes from the start to the end, i.e., $x_0$ to $x_{|\mathbf{X}|-1}$. Suppose we are generating the $j$-th position in the genomic sequence $\mathbf{X}$, and the fingerprinting ratio is $p$. While determining the output $y_j$, PFS checks the transition probability $Pr[x_j = q|x_{j-1} = y_{j-1}]$ for each $q$ in the alphabet. The scheme first eliminates all the states with a transition probability less than a given threshold $\tau$. PFS then forms a probability distribution for sampling the output among the remaining states. If the original value is not eliminated, $Pr[x_j]$ is set to $1-p$ with the rest $p$ proportionally assigned to the remaining states according to their transition probabilities. If the true value at position $j$ is eliminated, the scheme only generates the output proportionally from the remaining states.

During a generation, some positions are perturbed while some remain the same as the original true values. We use $\text{FPs}$ and $\text{NoFPs}$ to represent them, respectively. PFS balances the distribution of the $\text{FPs}$ by using the balancing factor $\theta$. The scheme checks the $\text{FP}$ count every $\lceil \frac{\tau}{\theta} \rceil$ points. If the actual $\text{FP}$ count is larger than expected, then the temporary fingerprinting ratio is changed to $p = (1-\theta)$. If the $\text{FP}$ are not enough, the ratio becomes $p = (1+\theta)$. PFS also integrates the Boneh-Shaw codes [5] into the fingerprinting scheme for collusion resistance. In the first fingerprinted copy of the genomic sequence, the authors arbitrarily pick $bc + bs \text{ FP}$ data points to represent the $\Gamma(bc, bs)$ Boneh-Shaw codes. The codes consist of $bc$ blocks, and each block has $bs$ data points. For the $i$-th copy, the scheme reports the original values for the first $i-1$ blocks and releases the ‘FP’ values for the rest blocks. For instance, if $bc = 3$, the scheme will output 4 different Boneh-Shaw codewords, i.e., $[B_{TP}^1, B_{TP}^2, B_{TP}^3], [B_{TR}^1, B_{TR}^2, B_{TP}^3], [B_{TR}^1, B_{TR}^2, B_{TR}^3]$, and $[B_{TR}^1, B_{TR}^2, B_{TR}^3]$, where $B_{FP}$ means the block has the same values as in the first fingerprinted trajectory and $B_{TR}$ refers to the true values of the block. Since there are at most $bc + 1$ different codewords for the Boneh-Shaw codes, they loop the codewords and assign them to the rest copies in turn. Since PFS changes some positions from $\text{FP}$ to $\text{NoFP}$, the number of $\text{FP}$ decreases. To maintain a stable number of $\text{FP}$ bits, the scheme calculates the fingerprinting ratio for the rest points by excluding the Boneh-Shaw codes before the start of each generation to make sure the aggregate fingerprinting ratio is similar.

5.3.2 Challenges for Using PFS for Location Data

There are several challenges to apply the PFS directly onto the location trajectories. First of all, the scheme itself is designed for data with limited states [30]. For location data, the states are infinite in reality. Even if we discretize the map area of interest using a grid, the states are still excessive, which potentially results in a complexity issue. Moreover, the logic inside PFS causes a vulnerability while used in the trajectories. PFS checks the correlations between the previously released point and the current value for each position. After filtering the low probable states, if the correlations hold, the scheme assigns a probability of $1-p$ to share the true value and the rest is proportionally distributed to the rest states according to the transition probability. This process normally works in location fingerprinting, but when the correlations are low between those points, things change. According to PFS, the scheme eliminates the true value since the correlations do not hold. Then, the scheme proportionally samples a point from the remaining $\tau$-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 $\tau$-probable set, the rest of the points will wander among
the $\tau$-probable set forever. For instance, in Figure 4, we show how
forced deviation is formed. Here, PFS is fingerprinting the $j$-th po-
bition in the trajectory, while $x'_{j-1}$ is the last fingerprinted location
and the dashed circular area in black is the $\tau$-probable set of $x'_{j-1}$,
$x_j$ is the actual location at position $j$, and it is in the $\tau$-probable set
of $x'_{j-1}$, PFS wants to sample a point among the $\tau$-probable set and
release that point. If the sampled point is located as $x'_j$ in Figure 4,
we realize that the next true value $x'_{j+1}$ is not in the $\tau$-probable set
of $x'_j$. In this case, the scheme will sample a location only among
the set without considering the true value. The next true value will
be more likely to occur outside of the $\tau$-probable set (marked by
red dashed circle) as well since the actual trajectory moves forward
and the sampled output sticks to the area close to the first separa-
tion, i.e., $x'_j$. If the generation continues, the fingerprinted locations
will stay around the first deviated location $x'_j$ and finally result in
forced deviation. We resolve this issue by considering directional
transitions between points during the sampling process in PFS.

5.3.3 Adaptive Fingerprinting Scheme. To solve the aforemen-
tioned challenges, we apply a direction-sensitive sampling to PFS
instead of the vanilla approach, called the adaptive fingerprinting
scheme (see Algorithm 1 for details). For a released point $x'_{j-1}$, we
first form a set containing all locations closer or equal to $x'_j$ than
$x'_{j-1}$ in the $\tau$-prob set, called $\tau$-closer set, which can be expressed
as $\text{prob}_\tau(x'_j) \leftarrow \{g | g \in x'_j \leq \|x'_{j-1} - x'_j\|_2, g \in \text{prob}_\tau(x'_{j-1})\}$.

Normally, if the true value $x^*_j$ is in the $\tau$-closer set, we sample
the output among it by setting the probability of choosing the true
value as $1 - p$ and the rest is proportionally assigned based on the
transition probability to the destination. We improve the sampling
process using temporary true values to avoid death loops in the
generation. There are four cases while selecting the true value at
the $j$-th position. If the true value $x^*_j$ is in the $\tau$-closer set of the
previously released location $x'_{j-1}$, there is no difference between
ours and in PFS. If the true value $x^*_j$ is not in the $\tau$-closer set, we
check its membership in the $\tau$-probable set and sample from the
same distribution as above but among the $\tau$-probable set instead.
If not, we check the closest point $\hat{x}$ to the true value in $\text{prob}_\tau(x'_{j-1})$.
If $\hat{x}$ is the same as $x'_{j-1}$, which means there exists no such location
closer than the true value, we let the temporary true value be the
original true value $x^*_j$. Otherwise, we choose $\hat{x}$ as temporary true
value at this timestamp and perform the proportional sampling
scheme. For the first location in the trajectory, we do not have the
conditional probabilities. Instead, we use the emission probability.

| Algorithm 1: Adaptive Fingerprinting Scheme |
|---------------------------------------------|
| **input** : Trajectory $X^* = [x^*_1, x^*_2, \ldots, x^*_m]$, location alphabet $G$, conditional probability in the correlations $\Pr(x_j | x_{j-1})$ for any $j \in [1, m]$, probability threshold $r$, fingerprinting ratio $p$ |
| **output** : Fingerprinted trajectory $X' = [x'_1, x'_2, \ldots, x'_m]$ |
| for all $j = 2, 3, \ldots, m$ do |
| 1. $\text{prob}_\tau(x'_{j-1}) \leftarrow \tau$-probable set of $x'_{j-1}$; |
| 2. $\text{prob}_\tau(x'_{j-1}) \leftarrow \{g | g \in x'_j \leq \|x'_{j-1} - x'_j\|_2, g \in \text{prob}_\tau(x'_{j-1})\}$; |
| 3. if $x'_j \in \text{prob}_\tau(x'_{j-1})$ and $|\text{prob}_\tau(x'_{j-1})| > 1$ then |
| 4. $\Pr(x'_j | x_{j-1}) = 1 - p, \Pr(x'_j | g) =$ |
| 5. $\text{prob}_\tau(x'_{j-1}) \leftarrow \{g | g \in \text{prob}_\tau(x'_{j-1})\}$; |
| 6. $x''_j \leftarrow$ sample from $\text{prob}_\tau(x'_{j-1})$; |
| 7. else if $x'_j \in \text{prob}_\tau(x'_{j-1})$ and $|\text{prob}_\tau(x'_{j-1})| = 1$ then |
| 8. $\Pr(x'_j | x_{j-1}) = 1 - p, \Pr(x'_j | g) =$ |
| 9. $\text{prob}_\tau(x'_{j-1}) \leftarrow \{g | g \in \text{prob}_\tau(x'_{j-1})\}$; |
| 10. $x''_j \leftarrow$ sample from $\text{prob}_\tau(x'_{j-1})$; |
| 11. $x_{\text{closest}} \leftarrow$ closest point to $x''_j$ in $\text{prob}_\tau(x'_{j-1})$; |
| 12. else if $|\text{prob}_\tau(x'_{j-1})| = 1$ then |
| 13. $x'_j \leftarrow x''_j$; |
| 14. else |
| 15. $\Pr(x'_j | x_{\text{closest}}) = 1 - p, \Pr(x'_j | g) =$ |
| 16. $\text{prob}_\tau(x'_{j-1}) \leftarrow \{g | g \in \text{prob}_\tau(x'_{j-1})\}$; |
| 17. $x'_j \leftarrow$ sample from $\text{prob}_\tau(x'_{j-1})$; |

5.4 Integration of the Fingerprinting Scheme with the Post-Processing

Observing that the post-processing (in Section 5.2) includes the
similar operations with the improved PFS considering the correla-
tions in the data, we integrate the post-processing scheme into the
probabilistic sampling step. One notable benefit of this integration
is the increase of the detection accuracy against collusion attacks in
the obfuscated datasets. Originally, we receive a trajectory $\hat{X}$ from
the post-processing and assume it to be the ground truth while
fingerprinting. We perturb around $p \ast |\hat{X}|$ location points and only
use them to identify the source of the unauthorized redistribution.
However, in reality, some other points in $\hat{X}$ have also been per-
turbed in the post-processing while choosing the surrogates, but
they are not involved in the detection process. To solve the issue,
we propose a hybrid scheme by combining the post-processing and
robust fingerprinting simultaneously for each point. By doing
so, we are able to make all the perturbed points from either steps
contribute to the detection, which increases the fingerprint code
length and finally increase the detection accuracy.

In the hybrid scheme, if a location point has low correlation with
the previous released point, we replace the true value with the one
closest to it among the $\tau$-probable set. If there is at least 1 location
in the $\tau$-probable set, we force the true value to be the output. If 2 or
more locations exist, we perform the proportional sampling similar to 5.3.3. When the location point has high correlation instead, we sample from the r-closer set if the r-closer set contains at least 2 points or from the r-probable set if not. The full algorithm (which we call “Hybrid Probabilistic Fingerprinting Scheme”) is shown in Algorithm 3 in Appendix D.

PIM provides differential privacy, and the following utility-focused post-processing and fingerprinting scheme does not violate the differential privacy guarantee. In the beginning, PIM offers initial guarantee differential privacy for each trajectory in the dataset [29]. According to the Proposition 3.2, any mapping function does not violate the guarantee of a differentially private method if the function does not utilize the actual values from the original dataset. Thus, the utility-focused post-processing in Section 5.2 does not break differential privacy since it only utilizes the noisy trajectories generated from PIM and the auxiliary information (i.e., correlations) from public sources. Meanwhile, the fingerprinting scheme also preserves the guarantee, which can be proved using the similar logic as above since it does not utilize any information from the original dataset. Therefore, we prove that our entire approach satisfies differential privacy.

5.5 Detecting the Source of the Unauthorized Redistribution

A probability-based and a similarity-based detection mechanisms have been proposed in [30]. Both detections calculate the accumulated scores and consider the one with the highest score as the guilty. Similar to [30], our elementary experiments show that the similarity-based detection works better in the location data, as the probability-based one is much more sensitive and often mis-charges an innocent analyzer if distortion attacks are involved. Both similarity-based and probability-based mechanisms rely on the exact match. 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 consider the analyzers with the shortest distance to the leaked one as suspicious and use the aggregate scores for the detection.

In the Hybrid Probabilistic Fingerprinting Scheme (in Section 5.4), we follow [30] and implement the detection of the Boneh-Shaw codes. While analyzing such trajectories, we first run the similarity-based detection and sort the scores in decreasing order. We pick the top \( \left\lfloor \frac{1}{\max \text{scores}} \right\rfloor \) candidates and determine the final source using the Boneh-Shaw inspection. For the remaining candidates, we traverse all the Boneh-Shaw blocks and find the first block with more FPs than NoFPs. Suppose we get the k’s block. Then, we check if there is a candidate that belongs to the k-th group in the Boneh-Shaw generation. If so, we accuse that candidate. If there are multiple candidates in the k-th group, we report the one with the highest score among them. If none of them is in the group, we accuse the one with the highest score.

When a dataset instead of a single trajectory is leaked, we implement an aggregate detection scheme to identify the source of the unauthorized redistribution. We first use the similarity-based detection to analyze each leaked trajectory in the dataset and calculate a similarity score for each analyzer. Then, we get the average for each analyzer and finally accuse an analyzer using the aforementioned Boneh-Shaw code inspection.

6 EVALUATION

We provide our evaluation results using a real dataset in this section. We run all experiments on a Dell PowerEdge R640 server with 64GB Memory (DDR4, 2666Mhz) and Intel Xeon processors with 40 cores. We run all experiments for more than 2,000 times with 20 dataset shuffles and report the average.

6.1 Experimental Setting

6.1.1 Baseline Approaches. For comparison, we implement a randomized fingerprinting scheme (RFS) as the baseline approach. RFS embeds a fingerprint value at each position with a uniform probability \( p \). At those selected positions, RFS generates a multivariate Laplacian noise to the original coordinates. The scales along the longitude and latitude are the same and aligned with our proposed scheme by the average deviation between corresponding points. We also implement the vanilla Boneh-Shaw codes and the vanilla Tardos codes to compare their performance with our scheme.

6.1.2 Dataset. We use the Geolife [33] dataset (Version 1.3) for evaluation. Geolife 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 the distance and 50,176 hours in time, where most of the locations are in Beijing, China. The dataset also covers a variety of outdoor activities, providing generality for the evaluation.

6.1.3 Pre-Processing. We pre-process the trajectories to avoid various data intervals. In the dataset, the time intervals vary from 1 second to more than 60 seconds. To build a reasonable correlation model for our system, we pre-process the trajectories to limit the time interval uniformly that approximates to 3 seconds to simulate the services such as Google Maps and carpooling.

Moreover, we remove the altitude and construct a 2-dimensional geographical area and define the area of interest by the latitude \( [39.6797, 40.1280] \) and the longitude \( [116.0287, 116.7064] \). Since there are some location points far from Beijing, we remove those fragments from the dataset. We discretize the area into cells uniformly using a 1000 \times 1000 grid both in the PIM and in the HPFS. The different selections of the grid size do not affect the performance since the two processes (PIM and HPFS) are independent, but we match the two grids for simplicity.

We randomly select 80% of the trajectories in the dataset to form the correlation model, while the rest are used for the evaluation on fingerprint robustness and data utility.

6.1.4 Parameter Settings. If not specified, we use the following parameter setting throughout the experiment. We assume 100 SPs get the copies and each trajectory which contains 200 – 1200 locations. We set \( \tau = 0.005 \) as the correlation threshold concluded from our preliminary 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 10 blocks and 5 location points in each block. For PIM, we follow [29] and set \( \delta = 0.01 \) for the \( \Delta \)-location set and the privacy budget \( \epsilon = 0.5 \) if we mention an obfuscated dataset. The fingerprinting ratio is set to 0.3. We suppose the attacker(s) use \( p_c = 0.5 \) and \( p_r = 0.5 \) in random and correlated distortion attacks, respectively, and 5 SPs collude by default.

6.1.5 Fingerprint Robustness Metric. A successful accusation is to catch the analyzer who leaks the data. If multiple analyzers collude, we consider catching one of the colluders. Since the Tardos
codes focus on catching all who leak the data, we adjust the accumulation process for alignment. More specifically, we only consider the one with the highest scores in the Tardos detection instead of using the threshold 20ck.

6.1.6 Utility Metrics. Following the existing works [14, 16, 29], we introduce our utility metrics as follows.

Correlation Fit Rate It indicates how accurately location pairs conform to the correlations, i.e., \( \Pr(x_{j+1} \mid x_j) \geq \tau \) for a location point \( x_j \). If the rate is high, most pairs follow the correlations; otherwise, the public correlations do not hold in the trajectories.

Query Answering of Trajectories The count query is one of the most common 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 of \( Q_t \) as

\[
RE_t = \frac{|Q_t(D, g) - Q_t(D', g)|}{\max(Q_t(D, g), b)}
\]

, where \( D \) is the original dataset and \( D' \) is the output of our scheme. We set \( b = 0.01 \times |D| \) according to [7, 14, 23, 32]. We calculate the average of the errors as \( AsRE_t \).

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 \( Q_p(D, g) \) that counts \( P \) in the dataset \( D \). The relative error \( AsRE_p \) is similarly defined as \( AsRE_t \).

Trajectory Similarity In the services like carpooling, the trajectory shape is an important feature that can be used for the service to design an optimal strategy. We use Discrete Fréchet Distance [12] to calculate the similarity between the trajectories from PIM and HPFS, respectively.

6.2 Experimental Results

We implement the proposed fingerprinting scheme and provide the experimental results here. We first show the fingerprinted dataset preserves high correlation fit rate compared with the baseline method. Then, we evaluate the fingerprint robustness in the worst case, i.e., leaking only one trajectory, since the fingerprint code length may be insufficient for a successful accusation. In Section 6.2.5, we provide the results when the attacker leaks the whole dataset instead of a single trajectory. Moreover, we show the data utility of the fingerprinted dataset. We also evaluate the time efficiency of our scheme. In Appendix E, we display additional results when fingerprinting is applied directly on the original dataset if the privacy is not considered.

6.2.1 Fingerprint Robustness Against Random Distortion Attacks (Single Trajectory). The detection accuracy against random distortion attacks (\( p_c = 0.5 \)) reaches 1.0 when the fingerprinting ratio \( \rho > 0.05 \) from our experimental results. Figure 5a shows the fingerprint robustness of our scheme against random distortion attacks compared under different random distortion ratio \( \rho_c \). Both our scheme and RFS show high robustness, but RFS loses the data utility (measured by \( AsRE_t \)) significantly compared with our scheme. In Figure 5b where we set \( \rho_c = 0.5 \), our scheme also outperform RFS in data utility. The reason that RFS has similar accuracy with APFS is that RFS has more candidates to choose for each point. Unlike APFS that considers correlations and samples the output from a limited set, RFS can freely choose any point to report from the multivariate Laplacian distribution. Thus, the candidates at each position are far more than our scheme and the binary codes, i.e., the Tardos codes and the Boneh-Shaw codes. The large alphabet results in a higher accuracy for RFS, but the utility loss is too high (See the red dashed line in all three subfigures). In conclusion, APFS shows high fingerprint robustness and data utility simultaneously.

6.2.2 Fingerprint Robustness Against Correlated Distortion Attacks (Single Trajectory). Our scheme achieves 100% detection accuracy against correlated distortion attacks when \( \rho > 0.05 \) and \( \rho_c = 0.5 \). If we let \( \rho = 0.3 \) and change \( \rho_c \), the accuracy only decreases from 100% once \( \rho_c \) reaches 0.98. In Figure 5c, our scheme keeps around 100% detection accuracy for all privacy budgets when the fingerprinting ratio \( \rho = 0.3 \). On the other side, the Boneh-Shaw codes show very low accuracy in all datasets no matter how we choose the parameters of the codes (\( bs \) and \( bc \)). In the obfuscated datasets where \( \rho_c = 0.98 \), all the methods are not influenced by the change of the privacy budget \( \epsilon \). In those experiments, our proposed scheme APFS achieves better accuracy compared with all the baseline approaches and also provides high data utility.

6.2.3 Fingerprint Robustness Against Majority Collusion Attacks (Single Trajectory). Figure 6a and 6b shows the performance of different fingerprinting schemes against majority collusion attacks. We implement the Boneh-Shaw codes and the Tardos codes on top of the random scheme. HPFS is almost 100% accurate and outperforms all other methods in Figure 6a. Figure 6b shows the fingerprint robustness with the change of the fingerprinting ratio \( \rho \) in the obfuscated datasets. The detection accuracy of HPFS is close to 100% if \( \rho \) reaches 0.3. Other methods require greater fingerprinting ratios for a similar performance, while the Boneh-Shaw codes only get 95% even if \( \rho = 1.0 \). Our results in Appendix G show that privacy level \( \epsilon \) does not show a significant impact on detection accuracy for both majority and probabilistic collusion attacks.

6.2.4 Fingerprint Robustness Against Probabilistic Collusion Attacks (Single Trajectory). In probabilistic collusion attacks, HPFS dominates in all experiments significantly (see Figure 6c and 6d). Our scheme maintains a high detection accuracy regardless of the colluding count \( c \), and it keeps > 70% accuracy after \( \rho = 0.3 \) in both DP and non-DP settings. The Boneh-Shaw codes and Tardos codes cannot effectively counter such attacks, thus showing low detection accuracy.

6.2.5 Fingerprint Robustness Against Multiple Attacks (Location Datasets). Figure 7 shows the performance against dataset redistribution when a dataset of 100 trajectories is leaked by the attacker. If the trajectory length is greater than 400 and the fingerprinting ratio is at least 20%, the detection accuracy reaches 90% in most cases, which proves our scheme are robust against those attacks if a subset or even an entire dataset is leaked.

6.2.6 Utility Evaluation. Figure 8 shows the data utility of the proposed scheme and compare it with the released dataset from PIM. Our method provides better correlation fit rate than other ones, and we provide the experiment results in Appendix F. Comparing the output from PIM and from the whole system (including PIM and HPFS), we can conclude that our fingerprinting scheme provides high data utility for all metrics.

6.2.7 Evaluation on Computation Time. We present the computation time of the proposed scheme in Table 2. For a dataset of
Figure 5: The fingerprint robustness against random distortion attacks and correlated distortion attacks for different values of distortion ratio $p_r$ and privacy level $\epsilon$.

(a) Random distortion with $p_r$  
(b) Random distortion with $\epsilon$  
(c) Correlated distortion with $\epsilon$

Figure 6: The fingerprint robustness against majority collusion attacks (MJR) and probabilistic collusion attacks (PROB) for different values of the number of colluding parties $c$ and fingerprinting ratio $p$.

(a) $c$, MJR  
(b) $p$, MJR  
(c) $c$, PROB  
(d) $p$, PROB

Figure 7: Experiment results of leaking a dataset of 100 trajectories against majority and probabilistic collusion attacks with different (a) colluding count $c$, (b) fingerprinting ratio $p$, and (c) trajectory length $l$.

(a) Accuracy vs. $c$  
(b) Accuracy vs. $p$  
(c) Accuracy vs. $l$

Figure 8: Data utility experiments of (a) query answering of trajectories $Q_t$, (b) query answering of patterns $Q_p$, and (c) trajectory similarity using discrete Fréchet distance.

(a) Query Answering of Trajectories  
(b) Query Answering of Patterns  
(c) Trajectory Similarity

100 trajectories with length equal to 500, the scheme only spends 2.5779 seconds generating one fingerprinted copy. Also, we observe the computation time increases linearly when the trajectory length grows. In conclusion, our scheme shows excellent time efficiency in generation and scales well for large datasets.

Table 2: Execution time of generating a fingerprinted dataset ($n = 100$)

| $l$   | 100  | 200  | 300  | 400  | 500  |
|-------|------|------|------|------|------|
| time (s) | 0.517 | 1.022 | 1.527 | 2.041 | 2.578 |
7 CONCLUSION AND FUTURE WORK

In this paper, we design a system that achieves both privacy preservation and robust fingerprinting while achieving high utility at the same time. We first apply the differentially private mechanism PIM to the dataset to protect the location privacy. Then, we implement a fingerprinting scheme that considers pairwise correlations in the location data and prevent the attackers from distortions and collusions. With the integration of a utility-boosting post-processing, our scheme HPFS 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 or a more realistic 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 collision 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] 2022. Doordash. https://www.doordash.com/. [2022]. [Online; accessed 29-March-2022].

[2] 2022. Google Maps. https://maps.google.com/. [2022]. [Online; accessed 29-March-2022].

[3] Osman Abul, Francesco Bonchi, and Miro 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. Unique in the crowd: The privacy bounds of human mobility. IEEE Transactions on Information Theory 59, 3 (2013), 1531–1557.

[5] Min Wu, Wade Trappe, J. Zane Wang, and KJ Ray Liu. 2003. Location privacy in the presence of prior knowledge. IEEE Transactions on Information Theory 49, 12 (2003), 3231–3241.

[6] Reza Shokri, Rastko Soric, and Srdjan Capkun. 2010. Robust fingerprinting for multimedia. IEEE Transactions on Dependable and Secure Computing 7, 3 (2010), 323–336.

[7] Tanxi Ji, Emre Yilmaz, Erman Ayday, and Pan Li. 2021. The Curse of Correlations on Robust Fingerprinting of Relational Databases. In 24th International Symposium on Research in Attacks, Intrusions and Defenses. 412–427.

[8] 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.

[9] Yang Cao, Masatoshi Yoshikawa, Yonghui Xiao, and Li Xiong. 2017. Quantifying uncertainty for anonymity in moving objects databases. In Proceedings of the 25th International Conference on Scientific and Statistical Database Management. IEEE, 247–262.

[10] Zhiyuan 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 cryptography. Springer, 455–466.

Appendices

A PLANAR ISOTROPIC MECHANISM

The planar isotropic mechanism aims to protect each location points 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 the adversarial knowledge, i.e., the probability distribution of the location at timestamp \( t \), the methods add calibrated noise to the actual location and shares the perturbed location.

At timestamp \( t \), PIM calculates the prior probability distribution \( P_t^\pi \) by \( P_t^\pi = P_{t-1}^\pi M \), where \( P_{t-1}^\pi \) is the posterior distribution at timestamp \( t-1 \), and \( M \) denotes the transition matrix. Based on the prior probabilities, it builds a \( \delta \)-location set \( \Delta_t \), which contains minimum number of locations with the probability sum larger or equal to \( 1 - \delta \), i.e., \( \Delta_t = \min \{ s \mid \sum_j P_j^\pi \geq 1 - \delta \} \), which means a subset of locations with the total probability less than \( \delta \) is omitted. After that, PIM releases the perturbed location given \( \Delta_t \) (details in the next paragraph) at timestamp \( t \) and call it \( s_t \), the posterior probability distribution is then updated as \( \sum_j P_j^\pi = \frac{1}{\sum_j P_j^\pi | u_j^t |} \frac{P_j^\pi}{P_j^\pi | u_j^t |} \frac{| u_j^t |}{P_j^\pi | u_j^t |} \) for each location \( s_j \), where \( u_j^t \) is the true location at timestamp \( t \).

The generation of the perturbed output contains 6 steps:

1. Generate a convex hull \( K \) from \( \Delta_t \):

2. Build a set \( \Delta_{V_k} \) by

\[ \Delta_{V_{k}} = \bigcup_{i=1}^{n} \{ v_i \} \]
We discuss several topics related to our approach including the potential privacy (e.g., using PIM) and send the noisy locations to the protected data are transmitted to the centralized server. Every time exists, we can alternatively set up a decentralized system, where the provider) has direct access to the collected dataset. If no such party into the system since the centralized data server (i.e., the service

B.2 Decentralized Setting

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. Thus, this mechanism achieve differential privacy for the trajectories in the location datasets.

B DISCUSSION

We discuss several topics related to our approach including the correlation model and an alternative setting of the system model that does not require a trusted third party.

B.1 Correlation Model

In this work, we use 2-gram Markov chain to model correlations. If we use a higher-order model, each 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, thus resulting in an inaccurate transition matrix. In our implementation, GeoLife [33] 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 [7, 29] (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.

B.2 Decentralized Setting

We build our system in the centralized setting, i.e., users’ location points are collected by a centralized data server and then processed by our scheme. This relies on a completely honest party involved into 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. Users can apply the DP protection locally on their devices by setting the desired privacy level they want to achieve. Then, the protected data are transmitted to the centralized server. Every time when the service provider collects real-time location information from users, the users instantly protect their locations under differential privacy (e.g., using PIM) and send the noisy locations to the centralized server. The server collects those locations sequentially and apply our 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, we definitely do not want to report incorrect locations. But if we use Google Maps to find nearby restaurants, we often accept a vague or slightly deviated localization. Service providers can choose the either setting based on the services they provide.

C TRAJECTORY POST-PROCESSING SCHEME

Algorithm 2 shows the steps of the post-processing scheme described in Section 5.2, where ||·||_2 denotes the l2-norm.

Algorithm 2: Trajectory Post-Processing Scheme

input: Noisy trajectory X = [x_1, x_2, ..., 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 \(r\)

output: Smoothed trajectory \(X' = [x'_1, x'_2, ..., x'_m]\)

1. \(x'_1 \leftarrow x_1\)
2. for all \(j \in [2, 3, ..., m]\) do
3. \(prob_t(x'_j|x_{j-1}) \leftarrow \tau\)-probable set of \(x'_j|x_{j-1}\);
4. \(x_{closest} \leftarrow\) closest point to \(x_j\) in \(prob_t(x'_j|x_{j-1})\);
5. if \(x_j \notin prob_t(x'_j|x_{j-1})\) then
6. if ||\(x'_j, \hat{x}_j\)|| \leq ||x'_j|x_{closest}||_2\) then
7. \(x'_j \leftarrow \hat{x}_j\)
8. else
9. \(x'_j \leftarrow x_{closest}\)
10. else
11. \(x'_j \leftarrow \hat{x}_j\)
12. end

D HYBRID PROBABILISTIC FINGERPRINTING SCHEME

We provide the Hybrid Probabilistic Fingerprinting Scheme in Algorithm 3.

E EXPERIMENT RESULTS WHEN FINGERPRINTING IS APPLIED ON THE ORIGINAL DATASET

In Section 6.2, we provide the experimental results of the proposed fingerprinting algorithm, when fingerprinting is applied after a perturbation-based differentially private mechanism planar isotropic mechanism (PIM). Therefore, we showed that the proposed scheme provide robust fingerprinting along with differential privacy. Here, we provide our results when the fingerprinting scheme is applied on original dataset without applying a differentially private data sharing mechanism. In Figure 9, the proposed scheme shows high accuracy and high data utility for all random distortion ratio \(p_r\)-s. In Figure 10, we give that our scheme maintains 100% detection accuracy for all correlated distortion attacks with distortion ratio \(p_r\in [0, 0.99]\), and it slightly decreases if the attacker distort all suspicious points with low correlations. We also find that a trajectory with 200 location points is enough to support
Hence, we can conclude that the proposed fingerprinting scheme also works well on original data (i.e., provides high robustness and utility) when privacy is not considered.

**Algorithm 3: Hybrid Probabilistic Fingerprinting Scheme**

```
Algorithm 3: Hybrid Probabilistic Fingerprinting Scheme

input: Trajectory $X' = \{x'_1, x'_2, \ldots, x'_n\}$, location alphabet $G$, occurrence probability $Pr[\{g\}]$ and transition probability $Pr[\{g\}\{g'\}]$ for any locations $g, g', g' \in G$; probability threshold $\theta$; fingerprinting ratio $p$; ratio balancing factor $\theta$; the first fingerprinted trajectory $X = \{x_1, x_2, \ldots, x_m\}$; the Boneh-Shaw codeword indexes $BSI = \{B_1, B_2, \ldots, B_{\theta \cdot \epsilon}\}$ and their corresponding fingerprinting boolean flags $BSF = \{F_1, F_2, \ldots, F_{\theta \cdot \epsilon}\}$

output: Fingerprinted trajectory $X' = \{x'_1, x'_2, \ldots, x'_n\}$

1. if $i \in BSI$ then
   2. if $F_i = \text{true}$ then
      3. $x'_i \leftarrow x_i$;
   4. else
      5. $x'_i \leftarrow x_i$;
   6. else
      7. $PD \leftarrow Pr[x'_i = x_i] = 1 - p_{\text{current}}, Pr[x'_i = g] = \frac{Pr[g]}{\sum_{g \in G} Pr[\{g\}\{g\}]} , g \in G$;
      8. $x'_i \leftarrow \text{sample from} \, PD$;
      9. $p_{\text{current}} \leftarrow p$;
   10. for all $j = 2, 3, \ldots, m$ do
      11. if $j \in BSI$ then
         12. if $F_j = \text{true}$ then
            13. $x'_j \leftarrow x_j$;
         14. else
            15. $x'_j \leftarrow x_j$;
         16. continue;
      17. $prob_{b}(x_{j-1}) \leftarrow \text{r-probable set of} \, x_{j-1}$;
      18. $prob_{b}(x_{j-1}) \leftarrow \{|g| x_g \| x_{j} \| \leq ||x_{j-1} - x_j||, g \in prob_{b}(x_{j-1})\}$;
      19. if $x_j \in prob_{b}(x_{j-1})$ and $|prob_{b}(x_{j-1})| > 1$ then
         20. $PD \leftarrow Pr[x'_j = x_j] = 1 - p, Pr[x'_j = g] = Pr[x'_j = \text{closest}] = 1 - p, Pr[x'_j = g] =$ $\sum_{g \in prob_{b}(x_{j-1})} Pr[\{g\}\{g\}] \times p, g \in prob_{b}(x_{j-1})$;
         21. $x'_j \leftarrow \text{sample from} \, PD$;
      22. else if $x_j \in prob_{b}(x_{j-1})$ and $|prob_{b}(x_{j-1})| = 1$ then
         23. $PD \leftarrow Pr[x'_j = x_j] = 1 - p, Pr[x'_j = g] =$ $\sum_{g \in prob_{b}(x_{j-1})} Pr[\{g\}\{g\}] \times p, g \in prob_{b}(x_{j-1})$;
         24. $x'_j \leftarrow \text{sample from} \, PD$;
      25. else
         26. $x_{\text{closest}} \leftarrow \text{closest point to} \, x_j \in prob_{b}(x_{j-1})$;
         27. if $|prob_{b}(x_{j-1})| < 1$ then
            28. $x'_j \leftarrow x_{\text{closest}}$;
         29. else
            30. $PD \leftarrow Pr[x'_j = x_{\text{closest}}] = 1 - p, Pr[x'_j = g] =$ $\sum_{g \in prob_{b}(x_{j-1})} Pr[\{g\}\{g\}] \times p, g \in prob_{b}(x_{j-1})$;
            31. $x'_j \leftarrow \text{sample from} \, PD$;
      32. if $j \mod \frac{1}{\epsilon} = 0$ then
         33. $\text{count} \leftarrow \# \text{of fingerprinted positions}$;
         34. if $\text{count} > p \cdot j$ then
            35. $p_{\text{current}} \leftarrow p \times (1 - \theta)$;
         36. else if $\text{count} < p \cdot j$ then
            37. $p_{\text{current}} \leftarrow p \times (1 + \theta)$;
         38. else
            39. $p_{\text{current}} \leftarrow p$;
   40. end
```

Figure 9: The fingerprint robustness against random distortion attacks in the original dataset with different distortion ratio $p_r$ ($p = 0.3$)

Figure 10: The fingerprint robustness against correlated distortion attacks in the original dataset with different distortion ratio $p_c$ ($p = 0.3$)

**F CORRELATION FIT RATE**

Figure 13 shows the correlation fit rate for RFS and our scheme when $p$ is set to 0.3. RFS only maintains 0.05 fit rate for all $\epsilon$. Our scheme gets 80% correlation fit rate if $\epsilon = 0.1$, and the rate approaches the one in the original dataset (marked by the blue dashed line) when $\epsilon$ reaches 0.7. The proposed scheme outperforms RFS and preserves most of the pairwise correlations.

**G EXPERIMENTS ON DIFFERENT PRIVACY BUDGETS**

Figure 14 shows detection accuracy v.s. different privacy budgets against majority collusion attacks, where the colluding count is set to 7 for better representation. HPFS gets the highest accuracy in all scenarios and it is stable with various privacy budgets. Similar results are shown in Figure 15 for probabilistic collusion attacks.
Figure 11: The fingerprint robustness against majority collusion attacks in the original dataset with different (a) colluding count $c$, (b) trajectory length, and (c) fingerprinting ratio $p$.

Figure 12: The fingerprint robustness against probabilistic collusion attacks in the original dataset with different (a) colluding count $c$, (b) trajectory length, and (c) fingerprinting ratio $p$.

Figure 13: The correlation fit rate with different privacy budget $\epsilon$.

Figure 14: $\epsilon$ v.s. accuracy against majority collusion.

Figure 15: $\epsilon$ v.s. accuracy against probabilistic collusion.