TrajGuard: A Comprehensive Trajectory Copyright Protection Scheme

Zheyi Pan[1], Jie Bao[2,3], Weinan Zhang[1], Yong Yu[1], Yu Zheng[2,3,4*]
[1]Department of Computer Science and Engineering, Shanghai Jiaotong University, China
[2]JD Intelligent Cities Research, China; [3]JD Intelligent Cities Business Unit, China
[4]School of Computer Science and Technology, Xidian University, China
{zhpan,wnzhang,yyu}@apex.sjtu.edu.cn;baojie@jd.com;msyuzheng@outlook.com

ABSTRACT

Trajectory data has been widely used in many urban applications. Sharing trajectory data with effective supervision is a vital task, as it contains private information of moving objects. However, malicious data users can modify trajectories in various ways to avoid data distribution tracking by the hashing-based data signatures, e.g., MD5. Moreover, the existing trajectory data protection scheme can only protect trajectories from either spatial or temporal modifications. Finally, so far there is no authoritative third party for trajectory data sharing process, as trajectory data is too sensitive.

To this end, we propose a novel trajectory copyright protection scheme, which can protect trajectory data from comprehensive types of data modifications/attacks. Three main techniques are employed to effectively guarantee the robustness and comprehensiveness of the proposed data sharing scheme: 1) the identity information is embedded distributively across a set of sub-trajectories partitioned based on the spatio-temporal regions; 2) the centroid distance of the sub-trajectories is served as a stable trajectory attribute to embed the information; and 3) the blockchain technique is used as a trusted third party to log all data transaction history for data distribution tracking in a decentralized manner. Extensive experiments were conducted based on two real-world trajectory datasets to demonstrate the effectiveness of our proposed scheme.

CCS CONCEPTS

• **Information systems** → **Spatial-temporal systems**.

KEYWORDS

Trajectory data; Copyright protection; Urban computing

ACM Reference Format:
Zheyi Pan, Jie Bao, Weinan Zhang, Yong Yu, Yu Zheng. 2019. TrajGuard: A Comprehensive Trajectory Copyright Protection Scheme. In *The 25th ACM SIGKDD Conference on Knowledge Discovery & Data Mining (KDD'19), August 4–8, 2019, Anchorage, AK, USA.* ACM, NY, NY, USA, 11 pages.
https://doi.org/10.1145/3292500.3330685

*Yu Zheng is the corresponding author.

KDD '19, August 4–8, 2019, Anchorage, AK, USA

ACM ISBN 978-1-4503-6201-6/19/08…$15.00
https://doi.org/10.1145/3292500.3330685

1 INTRODUCTION

With the advances in location-acquisition technology and mobile computing, massive trajectory data has been generated from vehicles, people and animals. Trajectory data is very valuable, as it is used in many urban applications, e.g., travel prediction [19] and recommendation [23], moving relationship mining [6, 11, 18], and urban planning [1, 20]. Many organizations have released trajectory data for profits, research credits, or community services. However, trajectory data is sensitive, as it contains much private information of moving objects [3, 4]. As a result, the access to trajectory data should be well regularized. However, currently, most of trajectory data is shared or exchanged without any supervision, i.e., the data is shared freely without tracking its ownership and the redistribution relations, which leads to many potential problems [2, 17].

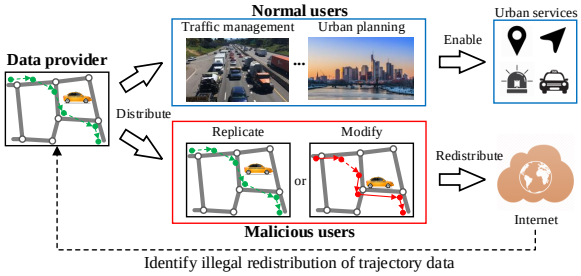


Figure 1: Plagiary of trajectory data.

To this end, a well regulated and supervised data sharing and redistribution scheme is vital for enabling the prosperous usage of trajectory data. In another word, it is important and necessary to identify the illegal redistribution of trajectory data by malicious data users (or data plagiarists), shown as the bottom part in Figure 1. However, malicious users can modify the original data easily by adding noises or shifting GPS points to avoid hashing-based data verification schemes, such as MD5. Moreover, the conventional ID embedding based verification schemes, which are widely used in protecting copyrights of images [13], audio [16], and videos [5], cannot be applied directly in trajectories for the following reasons.

- *Data Utility Preservation.* To make trajectory data usable in urban applications, the data utilities, i.e., spatio-temporal properties, of trajectory should be preserved. However, trajectory data contains semantic meanings about moving objects, such as routes and passing by landmarks. Thus every single point of a trajectory is sensitive to modifications. As the ID embedding based verification schemes for traditional multimedia do not consider the spatio-temporal properties of trajectory data, they fail to preserve data utility.
- *Various Attack Protection.* Data plagiarists can alter the representation of a trajectory (i.e., the attack) to avoid the hashing based

and ID embedding based verification schemes without sacrificing the data utility, e.g., resampling points on a trajectory and adding some noises. A robust and comprehensive scheme is necessary to protect the copyright of the data against all possible attacks.

• *Data Distribution Tracking.* When illegal usage of data is detected, the data plagiarist can still disavow it since the exchange of data is not supervised. As trajectory data is very valuable and contains sensitive personal information, to the best of our knowledge, there is still no authoritative third party company for trajectory exchange. Thus, it is necessary to have a protocol to form a decentralized community to maintain and validate all transactions.

The existing trajectory data protection schemes [8–10] consider the preservation of spatial property and embed data owner information into coordinates of GPS points. [22] further utilizes the preservation of temporal properties to find some pairs of feature points and then embeds information into them. However, all of them do not consider compositional situations and are vulnerable to some complex attacks, such as resampling the trajectory and then adding noise on each point. Moreover, all previous schemes do not consider the real-world situation that no trusted third-party company is available to maintain the transactions of trajectory data.

To tackle the aforementioned challenges, we propose an innovative scheme, which is a process towards protecting the security of spatio-temporal data through a knowledge (specifically, identity information) discovery process. The main intuition of the scheme is that the attacks applied by plagiarists should still preserve the spatio-temporal utility of the data. To this end, we first partition trajectories and distribute the identity information across all the sub-trajectories. Moreover, we use the centroid distance (i.e., a stable trajectory spatio-temporal attribute) to embedded information. Finally, a blockchain based transaction logger is maintained, such that all data transactions can be verified and the illegal usage of data can be proved. The main contributions of our work are:

• We summarize a comprehensive set of possible attacks on the trajectory dataset, including spatial attacks, temporal attacks, spatio-temporal attacks, and transformation attacks.

• We propose a novel and comprehensive scheme, named TrajGuard, to protect trajectory dataset from the illegal data redistribution. The scheme distributes the identity information based on the centroid distance of a large number of sub-trajectories.

• We employ a blockchain to maintain a transaction logger for trajectory data distribution, which decentralizes the authority for validating the trajectory data transaction.

• We evaluate our scheme on two real-world datasets: *Geolife* and *T-Drive*. The experimental results verify that the embedded information is robust against various attacks and demonstrate the superiority of our scheme beyond the baselines.

2 PRELIMINARIES

2.1 Basic Concepts

Definition 1. ***Trajectory.*** *A trajectory $\mathcal{T}$ is a sequence of GPS points, denoted as $\mathcal{T} = \{\langle t_i, p_i\rangle \mid t_i < t_{i+1}\}$, where t_i is the timestamp and p_i is a two-dimensional vector (x_i, y_i) which represents the longitude and latitude of a moving object.*

Definition 2. ***Trajectory Location Estimation.*** *A trajectory is continuous spatially and temporally in the real world, which can be approximated by a function as $P_{\mathcal{T}}(t)$. The function returns a location of the moving object at time t. For example, in this paper, we use linear interpolation for trajectory estimation:*

$$P_{\mathcal{T}}(t) = p_i + \frac{t - t_i}{t_{i+1} - t_i}(p_{i+1} - p_i) \qquad t \in [t_i, t_{i+1}]. \tag{1}$$

Definition 3. ***Trajectory Modification Bound.*** *Given an original trajectory $\mathcal{T}$, a modified trajectory $\mathcal{T}'$ is usable, if two trajectory can be aligned on the time axis and the spatial modification is bounded by a threshold τ. Formally, the modification bound is expressed as: $\max_t |P_{\mathcal{T}}(t) - P_{\mathcal{T}'}(t)| \le \tau$, where $|a - b|$ represents the distance between point a and b. In this paper, we adopt Euclidean distance.*

Definition 4. ***Trajectory Centroid.*** *The centroid of a trajectory $\mathcal{T} = \{\langle t_i, p_i\rangle | i = 0 \ldots n-1\}$ is defined as the average coordinates of points on the trajectory, which can be expressed as:*

$$c_{\mathcal{T}} = \frac{1}{t_{n-1} - t_0} \sum_{i=0}^{n-2} \int_{t_i}^{t_{i+1}} P_{\mathcal{T}}(t)\, dt. \tag{2}$$

Definition 5. ***Trajectory Centroid Distance.*** *The centroid distance of a trajectory $\mathcal{T} = \{\langle t_i, p_i\rangle | i = 0 \ldots n-1\}$ is the average distance between estimated point and the trajectory centroid:*

$$d_{\mathcal{T}} = \frac{1}{t_{n-1} - t_0} \sum_{i=0}^{n-2} \int_{t_i}^{t_{i+1}} |P_{\mathcal{T}}(t) - c_{\mathcal{T}}|\, dt. \tag{3}$$

2.2 Problem Formulation

Assume that a data provider needs to distribute a trajectory dataset $\mathcal{T}$ with a required preservation of data utility:

$$\max_t |P_{\mathcal{T}}(t) - P_{\mathcal{T}'}(t)| \le \tau,$$

where τ is the maximum permissible excursion. In this paper, we propose a comprehensive scheme which embeds identity information into the trajectory dataset, and the embedded identity information is recoverable to confirm the ownership of the data provider and the identity of the plagiarist from an attacked dataset, as long as the attacks preserve the data utility.

2.3 System Overview

Figure 2 gives an overview of TrajGuard. It is resistant to a comprehensive set of trajectory modifications/attacks, detailed in Section 3. There are three main processes in the scheme: 1) *identity embedding* (detailed in Section 4), 2) *ownership detection* (detailed in Section 5), and 3) *ownership tracking* (detailed in Section 6). We further present the pseudo code of the system in Appendix Section A.

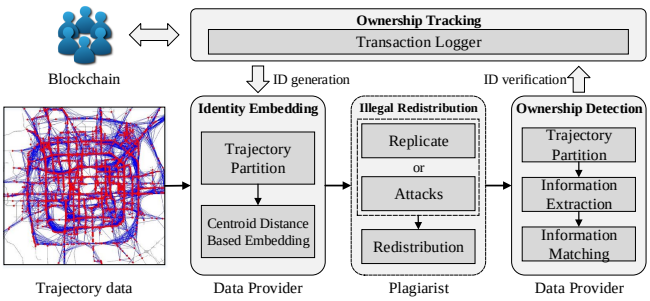


Figure 2: Overview of TrajGuard

• **Identity Embedding.** It embeds identity information into trajectory dataset. We first partition the original trajectory into sub-trajectories to distribute the identity information across the entire dataset. In each sub-trajectory, the trajectory centroid distance is used to embed information. Finally, all the embedded sub-trajectories are concatenated to form a completed trajectory dataset.
• **Ownership Detection.** It identifies the ownership of trajectory data, when the data provider obtains a suspicious dataset. First, it partitions the trajectories with the same settings as the embedding process. Then, the information extraction module extracts the embedded information from each sub-trajectory. If the extracted information matches the embedded information, the data ownership is confirmed and the illegal redistribution activity can be proved.
• **Ownership Tracking.** The ownership tracking process records data transaction history, which serves as a trusted third party to prove the existence of data transactions. In this way, data plagiarists cannot deny the illegal data redistribution activity, when identity information is detected. To make it equitable and endorsed by all individuals, the transaction logger is maintained in a decentralized mode by a blockchain, such that no one can dominate the logger.

3 ATTACKS ON TRAJECTORY DATA

3.1 Spatial Attacks

Attacks on the spatial domain modify the coordinates of trajectories. Figure 3 depicts two types of such attacks.

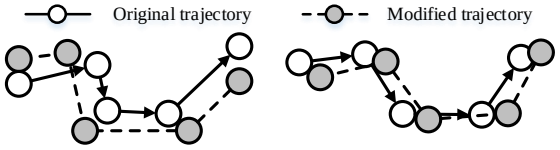


Figure 3: Attacks on the spatial domain

• **Additive Spatial Noise**. It's a popular attack for trajectory data [8, 10, 22]. The data plagiarist disturbs the identity information by adding noise on the coordinates of points. In this paper, we consider the additive gaussian noise on coordinates (AGNC).
• **Rigid Transformation (RT)**. It contains two types of transformations: shift and rotation. In RT attack, the shape of the trajectory and the distance between any pairs of points are preserved.

3.2 Temporal Attacks

The attacks on the temporal domain modify the timestamps. As shown in Figure 4, each timestamp t_i is modified by Δt_i respectively. There are two types of attacks on the temporal domain.

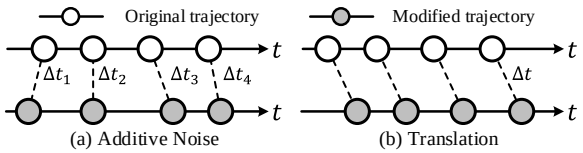


Figure 4: Attacks on the temporal domain

• **Additive Temporal Noise**. Similar to the additive noise added on the coordinates, data plagiarist can also add small noise Δt_i on each timestamp. In this paper, we consider the additive gaussian noise on timestamps (AGNT).
• **Temporal Shift (TS)**. For the applications which are not sensitive to the occurring time of a trajectory, the data plagiarist can shift the temporal information of the trajectory, e.g., subtracting all the timestamps of GPS points by one day. Formally, $\Delta t_1 = \Delta t_2 = \ldots = \Delta t$ where Δt is a constant chosen by the data plagiarist. In this way, the temporal relationship between any pairs of points is invariant, so the utility of data is preserved.

3.3 Spatio-Temporal Attacks

The spatio-temporal attacks modify trajectories on both the spatial and the temporal domains. Figure 5 depicts three attacks on the spatio-temporal domain.

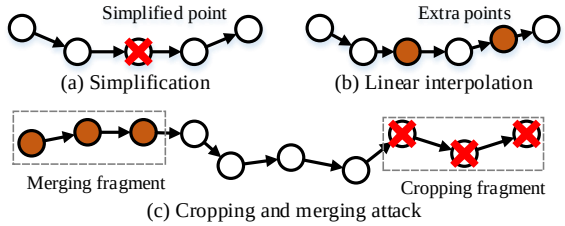


Figure 5: Attacks on the spatio-temporal domain

• **Simplification (SIMP)**. Trajectory simplification [22] removes the redundant points in a trajectory, which is mainly used for improving the efficiency of trajectory data processing.
• **Linear Interpolation (LI)**. It adds some extra points by linear interpolation (see Eq. 1) between some adjacent points.
• **Cropping and Merging Attack (CM)**.The cropping attack removes some meaningless parts of trajectory. For example, when estimating the average speed of real-time traffic, we can remove some pieces of a trajectory which would not affect the estimation. While the merging attack [7] fuses the pieces of trajectories from different sources. Both attacks can keep the data utility.
• **Hybrid Attack**. Furthermore, the data plagiarist can simultaneously apply the aforementioned spatial, temporal and spatio-temporal attacks, leading to a complex alteration of the trajectory.

3.4 Transformation Attacks

We also discuss complex transformation of trajectories, which can wipe out identity information while keeping the utility of data.
• **Double Embedding (DE)**. The data provider can embed information into trajectory data, while the data plagiarist can also embed his own information to disturb the detection process. Although the attacked dataset contains the information of the plagiarist, the information embedded by the data provider should still be detected.
• **Map Matching (MM)**. For the trajectories collected in the city (e.g. taxi trajectory), a common preprocessing stage is map matching [24]. Map matching technique maps GPS points to the points on road networks, so the coordinates are modified without breaking the utility. Thus the embedded information should be preserved if the data plagiarist applies such modification.

4 IDENTITY EMBEDDING PROCESS

In this process, the identity information is inserted into trajectory data to show the ownership of the data provider. Since many attacks can crop, modify and shift the trajectories spatio-temporally, there are two main challenges in embedding the identity information into

trajectory data: 1) the embedded information needs to be distributed across the dataset in case of being lost during attacks; and 2) the identity information needs to be embedded with a robust trajectory attribute, to be more stable during the attack. To this end, in the identity embedding process, two main modules are employed: 1) trajectory partition; and 2) centroid distance based embedding.

The trajectory partition module aims to broadcast the identity information everywhere in trajectory data. Thus, the information can be recovered, even if the data is cropped or randomly sampled.

After partitioning trajectories, the centroid distance based embedding module is employed to insert the information into each sub-trajectory. As any potential attack needs to preserve a certain degree of spatio-temporal property to guarantee the data utility, centroid distance is a very stable attribute. Thus, we embed the information by modulating the centroid distance of each sub-trajectory.

4.1 Trajectory Partition

Main Idea. In this module, each trajectory in the dataset is partitioned into pieces to distribute the identity information. More specifically, the embedding information for each sub-trajectory is generated by the identity information and the sub-trajectory index. To guarantee the accuracy of recovered information, there should be sufficient points in each sub-trajectory. The main insight is from *Central Limit Theorem*, where more points in a polygon increase the robustness of its centroid distance.

One intuitive way to perform trajectory partition, is to divide them by using spatial properties, e.g., using spatial grids or R-trees. As the distribution of points on the spatial domain is imbalanced, partitioning by spatial grids cannot guarantee the number of points in each sub-trajectory. As a result, some sub-trajectories may contain few points, which decreases the robustness of the centroid distance and the embedded information. Moreover, simply enlarging the partition granularity (e.g., grid size) does not address the problem, as it decreases the number of sub-trajectories, and makes the scheme vulnerable. On the other hand, self-adapting spatial partition method, e.g., R-tree, also cannot be employed directly here, as the data plagiarists can change the number of GPS points in a trajectory by interpolation or simplification, which makes the trajectory partition module inconsistent after being attacked.

Another intuitive way to perform the partition, is to divide the trajectories based on the temporal domain, as in most cases, the trajectories in the same dataset have similar sampling rate, which makes it easier to control the number of GPS points in each sub-trajectory. However, partitioning trajectories temporally is vulnerable to the attack on the temporal domain, i.e., temporal shifts, as we may not be able to generate the same sub-trajectories during the detection process.

To this end, both spatial and temporal domains are used to overcome the drawbacks from the spatial or temporal based partition. The main insight here is to find a reference point for each sub-trajectory with a reference timestamp Δt, such that $\forall i, t_i - \Delta t$ is invariant to Temporal Shift. The spatial partition, e.g., grid, is used here to identify the reference points, which is selected as the intersection points between the spatial grid and the trajectory.

Algorithm. To prevent others inverting the embedding process to remove the embedded information, two secret keys ω_s and ω_t are chosen by the data provider to partition trajectory. There are two main steps in the trajectory partition module:

- **Spatial Partition.** The spatial plane is divided into uniform grids with the size of ω_s. The trajectories are partitioned spatially and bounded by the points that intersect spatial grids.
- **Temporal Partition.** For the sub-trajectory inside a grid with index (x, y), it is further partitioned using a temporal interval $[\Delta t + k\omega_t, \Delta t + (k+1)\omega_t]$, where Δt is the timestamp when the object first passing the boundary of the grid and k is the temporal index. Finally, the embedded information of sub-trajectory $\mathcal{T}_k^{x,y}$ is generated by a hash function, i.e., $\text{hash}(IDs, x, y, k)$, whose input is the combination of the identity information (both the data provider and data user), the grid index (x, y), and the temporal index k of the sub-trajectory.

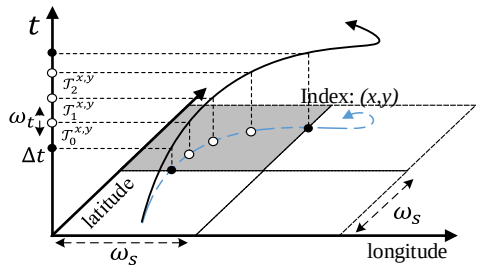


Figure 6: Partition trajectory on spatio-temporal domain.

Example. Figure 6 gives an example, where the size of grid is ω_s and the temporal interval is ω_t. The trajectory is firstly partitioned spatially, as demonstrated as the line segment between the black points that intersect the grid boundary. After that, the sub-trajectory in the grid (x, y) with gray color is further partitioned into $\{\mathcal{T}_k^{x,y}\}$ by white points, where k is the temporal index and the timespan of them is ω_t. Each of the sub-trajectory between two white points will be embedded with the information generated by the hash function.

Analysis. To guarantee enough points in each sub-trajectory for more robust centroid distance, we can select ω_t according to the sampling rate of the trajectory. To produce more sub-trajectory for more embedding information, selecting a suitable ω_s is vital for the data provider, as more embedding information indicates more reliability of the scheme. If ω_s is set to a very small value, the sub-trajectories produced by the spatial partition may fail to further partition on the temporal domain. On the other hand, if ω_s is set as a very large value (e.g., $\omega_s \to \infty$), the trajectory is hard to pass through the boundary of grids to find a reference timestamp Δt for initial points in the trajectory. Both scenarios decrease the number of sub-trajectories for embedding, which would decrease the reliability of the ownership detection process. Detailed relations between the selection of different ω_t, ω_s and the robustness are presented in the experiment section, i.e., Section 7.4.

4.2 Centriod Distance based Embedding

After the trajectories are partitioned into sub-trajectories, we embed the information in this module. The module mainly contains two procedures: 1) calculate the hash function to generate embedding information; 2) embed information by modulating centroid distance.

Main Idea. To keep the utility of trajectory data, the capacity of embedding information for each sub-trajectory should be limited.

Since we have many sub-trajectories (e.g., more than ten thousand sub-trajectories in a dataset), embedding a single bit of information into each sub-trajectory is sufficient to show the ownership of the data provider, if most of the bits (e.g., 90% bits) can be recovered correctly by the detection process. Moreover, the conflict problem caused by the single-bit hash function, i.e., different identity information resulting in the same embedded information, can be significantly alleviated because of large amounts of embedded bits. As a result, hash(IDs, x, y, k) returning a single bit, i.e., 0 or 1, is used to generate the embedded bits.

Next, we mainly describe how we embed a single bit into the centroid distance. The main insight here is to take advantages of the data utility requirement (i.e., preservation of the spatio-temporal properties of sub-trajectories), which is guaranteed in any type of attack, and select a robust attribute for embedding. Thus, the centroid distance of a sub-trajectory is a very robust attribute, as it reflects an overall spatio-temporal distribution of the points. In this module, we modulate the centroid distance, by scaling the distance from each point on a trajectory to its centroid. To guarantee the trajectory modification bound τ, the modification on the centroid distance should be less than τ. As modification under different τ is different, we firstly normalize the centroid distance by dividing τ, such that the modification of it should be less than 1. Since we embed one bit, i.e., 0 or 1, we classify centroid distance into two categories, where the decimal part of it in $[0, 0.5)$ indicates bit 0 and in $[0.5, 1.0)$ indicates bit 1. In this way, we can alter the decimal part of the centroid distance to embed information. More specifically, according to the embedded bit, the decimal part of centroid distance is set as one middle value of the two scopes, i.e., 0.25 (for bit 0) or 0.75 (for bit 1), such that attacks cannot easily change the scope of the centroid distance because of its robustness.

A five-step process is employed here: 1) calculate the embedding bit, 2) calculate and normalize the centroid distance; 3) embed the bit at the decimal part of the centroid distance; 4) modify points by scaling the distance from them to the centroid; and 5) refine the change at each point to guarantee the data utility.

Algorithm. Given an original sub-trajectory $\mathcal{T}$, the first step is calculating the embedding bit b by the hash function. Then we calculate the centroid $c_{\mathcal{T}}$ and centroid distance $d_{\mathcal{T}}$ by Eq. 2 and Eq. 3, and normalize $d_{\mathcal{T}}$ by $d_{\mathcal{T}} \leftarrow d_{\mathcal{T}}/\tau$. Thirdly, a new centroid distance $d_{\mathcal{T}'}$ is calculated by embedding a bit b into $d_{\mathcal{T}}$:

$$d_{\mathcal{T}'} \leftarrow \lfloor d_{\mathcal{T}} \rfloor + \begin{cases} 0.25 & \text{if} \quad b = 0 \\ 0.75 & \text{if} \quad b = 1. \end{cases} \tag{4}$$

Fourthly, calculate the new point by $p_i' = \frac{d_{\mathcal{T}'}}{d_{\mathcal{T}}}(p_i - c_{\mathcal{T}}) + c_{\mathcal{T}}$, such that the new trajectory has the same centroid $c_{\mathcal{T}}$ while the normalized centroid distance becomes metric $d_{\mathcal{T}'}$ (the proof is simply substituting p_i for p_i' in Eq. 2 and Eq. 3). However, $|p_i' - p_i|$ may be larger than modification bound τ. Finally set $p_i' = \frac{p_i' - p_i}{|p_i' - p_i|}\tau + p_i$ when $|p_i' - p_i| > \tau$, to make sure $|p_i' - p_i| = \tau$.

Example. An example of the embedding algorithm is shown in Figure 7. At first the centroid distance of the original trajectory is 4.99. To embed 0 into this trajectory, the centroid distance is modulated to $d_{\mathcal{T}'} = 4.25$. Then the coordinates p_i' are calculated by

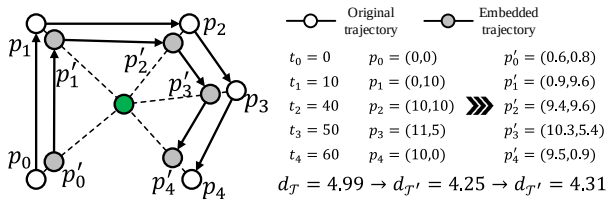


Figure 7: Example of embedding 0 into trajectory ($\tau = 1$).

scaling the distance from centroid to p_i by $\frac{d_{\mathcal{T}'}}{d_{\mathcal{T}}}$. Finally p_i' is modified such that $|p_i - p_i'| \leq 1.0$, and the centroid distance becomes 4.31.

Analysis. Since data plagiarists do not have the secret keys of the trajectory partition module, they do not know the detailed information of each sub-trajectory, such as the number of points, the length, the shape, and etc. To ensure the data utility, data plagiarists have two choice to modify sub-trajectories: 1) adding uniform shift to all points while keeping the shape of the trajectory, e.g., rigid transformation; 2) introducing random noise on each point such that $|P_{\mathcal{T}'}(t) - P_{\mathcal{T}''}(t)| \leq \tau$. However, none of these two approaches can be used effectively against our scheme.

For the case of adding uniform shifts, the shape of a trajectory is kept, which means the distance between any pair of points is the same. As a result, the centroid distance does not change. For the case of introducing random noises, the change of centroid distance is limited as long as the data utility is preserved. We will show the modification of centroid distance in Section 7.4.

5 OWNERSHIP DETECTION PROCESS

When a suspicious trajectory dataset is obtained by the data owner, the ownership detection process extracts the embedded information from each sub-trajectory and verifies it. It is essentially a reversed procedure of the identity embedding process, which consists of three main steps: 1) adopting a trajectory partition module to split the trajectories with exactly the same setting as the embedding process; 2) extracting information from the sub-trajectories; 3) judging whether the extracted information can be matched to the embedded one. If the embedded information is detected from a redistributed dataset, the data provider can ask the transaction logger to verify the identity information and then claims the ownership of the data. In this section, we will elaborate on each module in detail.

5.1 Trajectory Partition

After acquiring the suspicious trajectory dataset, this module partitions the trajectory by using the same ω_s and ω_t. The same index, i.e., spatial index (x, y) and temporal index k, can be generated during partition process. After that, the embedded information can be calculated in the same way by the hash function using the spatio-temporal indexes, i.e., x, y, k, and the identity information.

5.2 Information Extraction

This module extracts the embedded information from each sub-trajectory by examining the centroid distance. The insight is that the modification on centroid distance is small and stable. In the embedding algorithm, the embedded bit is 0, when the decimal part of a normalized centroid distance is near 0.25. On the other side, the embedded bit is 1, when it is near 0.75. Thus, the extraction algorithm calculates and normalizes the centroid distance $d_{\mathcal{T}''}$ of an

attacked sub-trajectory $\mathcal{T}''$, and then extract the bit by the formula:

$$b'' = \begin{cases} 0 & \text{if} \quad d_{\mathcal{T}''} - \lfloor d_{\mathcal{T}''} \rfloor < 0.5 \\ 1 & \text{if} \quad d_{\mathcal{T}''} - \lfloor d_{\mathcal{T}''} \rfloor \geq 0.5. \end{cases} \tag{5}$$

In Figure 7, as the centroid distance $d_{\mathcal{T}''} = 4.31$, the extracted bit is 0 (i.e., the same as what we embedded).

5.3 Information Matching

As the ground truth of embedded bit for each sub-trajectory can be calculated by the identity information and its spatio-temporal index, i.e., hash(IDs, x, y, k) (see Section 4.2), and the extracted bit can be calculated by Eq. 5, the last step is judging whether the extracted bits can be matched to the ground truth.

We define a binary sequence $M = \langle 0, 1, 1, 0, 1, 1, 1, \ldots \rangle$, where the i-th number indicates whether the extracted bit of i-th sub-trajectory is equal to the embedded bit. If there is a subsequence $M' = \{M_k | i + 1 \leq k \leq j\}$ whose length is longer than a constant L and its accuracy (the number of correct bits divided by the total number of bits) is greater than a threshold α, then those sub-trajectories form a fragment which has a copyright issue.

If we enumerate all possible index i, j, the time complexity is $O(|M|^2)$ which is unacceptable as datasets contain large amount of sub-trajectories. We can further optimize it to find M' linearly. First, we compute a new sequence: $S_i = M_1 + \ldots + M_i$. Then the accuracy of a subsequence whose index is from $i + 1$ to j can be expressed as $\frac{S_j - S_i}{j-i}$. Suppose $\beta_i = S_i - \alpha i$. If $i < j$ and $\beta_i < \beta_j$, we can find a subsequence M' that $\frac{S_j - S_i}{j-i} = \frac{\alpha(j-i)+(\beta_j-\beta_i)}{j-i} > \alpha$. Therefore the algorithm can be implemented by simply maintaining the minimum value of the prefix of $\{\beta_i\}$. if $min(\beta_1, \beta_2, \ldots, \beta_{i-L-1}) < \beta_i$, then we find a subsequence with the length $> L$ and accuracy $> \alpha$.

6 OWNERSHIP TRACKING PROCESS

Traditionally, trusted third-party organizations act an intermediate to embed information and transmit data, such that the whole process is under supervision to prevent the scenario that the plagiarist disavows the data transaction. However, since the great value and crucial privacy of trajectory data, centralizing the authority and data to a single organization is highly risky. Thus, a decentralizing protocol is needed to supervise trajectory distribution transaction.

To deal with such problem, we employ a *Transaction Logger*, which leverages the blockchain [12, 15] to record data transactions in a decentralized mode. First, the data provider issues a transaction with identity information, i.e., the unique certificated names of the data provider and the user. All the names are preliminarily approved by all users on the blockchain. Then, the data user validates the transaction. Both the data provider and the user add their own digital signatures to confirm the transaction. Next, the transaction is broadcast to all nodes on the blockchain. After that, the transaction information can be validated by signatures and then appended to the blockchain. Finally, the data provider embeds the identity information, i.e., IDs of data provider, user, and transaction, into trajectory data and transmits it to the user.

As long as the blockchain is reliable, the information on it is retrievable and unalterable. So it can be directly used to verify the existence of transactions, and the individuals involving in them.

7 EVALUATION

7.1 Settings

7.1.1 Datasets. We employ two real-world datasets for evaluation:

- **GeoLife** [25–27]. It records many types of human outdoor movements. GeoLife has 182 trajectories, containing 24,876,978 sampling points. Most trajectories in GeoLife are logged as dense representation, e.g., 1-5 seconds per point.
- **T-Drive**. It records the movements of taxicabs in Beijing. The dataset has 8,874 trajectories, containing 10,088,335 points in total. Most of trajectories in T-Drive are logged as sparse representation which has a much larger sampling rate (e.g., 10 minutes).

To keep the utility of these two datasets, the modification bound τ is set as $1e-5$ and $1e-4$ for GeoLife and T-Drive, respectively.

7.1.2 Baseline Methods.

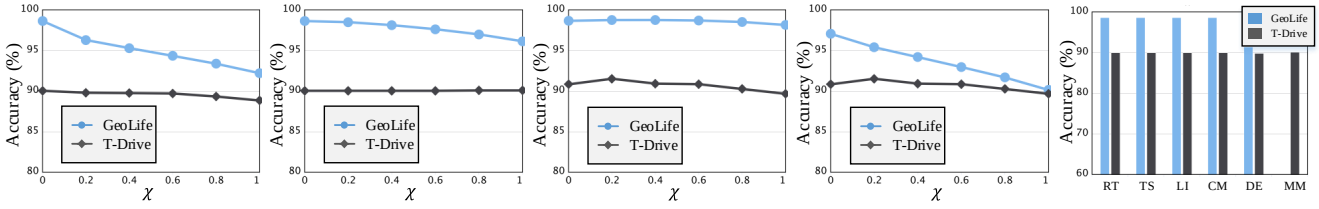
- **Fourier Descriptor Modulation [10, 14] (FDM)**. It leverages the robustness of trajectory data on the frequency domain. The embedding process mainly consists of three steps: 1) transform trajectory into the frequency domain; 2) embed identity information into the Fourier descriptors; 3) transform the trajectory back to the spatial domain. The illegal usage of trajectory data can be judged by extracting bits from the attacked trajectory on the frequency domain and then matching them to the embedded bits.
- **Distance Modulation between Feature Points [22] (DMFP)**. In the embedding process, it finds some pairs of feature points, and embeds one bit into each of them by modulating the pairwise distance. In the detection process, it extracts bits from the pairwise feature points and matches them to the embeddit bits.

7.1.3 Attacks. To keep data utility, the original trajectory $\mathcal{T}$, the embedded trajectory $\mathcal{T}'$ and the attacked trajectory $\mathcal{T}''$ satisfy: $\max_t |P_{\mathcal{T}}(t) - P_{\mathcal{T}'}(t)| \leq \tau$ and $\exists \Delta t, \max_t |P_{\mathcal{T}'}(t) - P_{\mathcal{T}''}(t + \Delta t)| \leq \tau$ (plagiarist can apply TS attack with Δt). To further verify the robustness against attacks with different strength, we introduce a coefficient $\chi \in (0, 1]$ to bound the attacks by $\exists \Delta t, \max_t |P_{\mathcal{T}'}(t) - P_{\mathcal{T}''}(t + \Delta t)| \leq \chi\tau$. The details of the attacks are as follow:

- **AGNC**. For any point p_i in a trajectory, first we sample a value c_i from $\mathcal{N}(\mu = 0, \sigma^2 = 1)$ on $[-1, 1]$ and a random direction d_i (2D vector) with $|d_i| = 1$. Then modify p_i as $p_i + c_i \chi\tau \cdot d_i$.
- **RT**. Rotate and shift trajectory spatially with random values.
- **AGNT**. For each timestamp t_i, first we sample a value c_i from $\mathcal{N}(\mu = 0, \sigma^2 = 1)$ on $[-1, 1]$. Second we calculate the maximum modification on this timestamp: $\Delta t_i = \tau / max(\frac{p_{i+1}-p_i}{t_{i+1}-t_i}, \frac{p_i-p_{i-1}}{t_i-t_{i-1}})$. Finally, the timestamp is set as $t_i + c_i \chi \cdot \Delta t_i$.
- **TS**. Shift all timestamps with the same random value.
- **SIMP**. We use Douglas-Peucker algorithm [24] to simplify a trajectory $\{p_1, p_1, \ldots, p_n\}$. If for all i the time synchronized Euclidean distance from p_i to the line $\overline{p_1 p_n}$ is less than $\chi\tau$, then we can replace the original trajectory by $\overline{p_1 p_n}$. Otherwise, it recursively partitions the original trajectory into two trajectories by selecting the splitting point contributing the biggest time synchronized distance to $\overline{p_1 p_n}$.
- **LI**. We randomly add some new points by Eq. 1.
- **CM**. We randomly crop a fragment within a trajectory.
- **Hybrid**. We implement multiple attacks on the trajectory. The order is LI, AGNC(χ), AGNT(χ), SIMP(χ), TS and RT.
- **DE**. After embedding process, we use the same scheme to embed another identity information by using different secret keys.

Scheme	Dataset	Spatial Attacks		Temporal Attacks		Spatio-Temporal Attacks				Transformation Attacks	
		AGNC	RT	AGNT	TS	SIMP	LI	CM	Hybrid	DE	MM
FDM	GeoLife	No	**Yes**	**Yes**	**Yes**	No	No	No	No	No	-
	T-Drive	**Yes**	**Yes**	**Yes**	**Yes**	No	No	No	No	No	**Yes**
DMFP	GeoLife	No	**Yes**	**Yes**	No	**Yes**	**Yes**	No	No	**Yes**	-
	T-Drive	No	**Yes**	**Yes**	No	**Yes**	**Yes**	No	No	**Yes**	No
TrajGuard	GeoLife	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	-
	T-Drive	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**

Table 1: Robustness comparison for various schemes.



(a) AGNC (b) AGNT (c) SIMP (d) Hybrid attack (e) RT, TS, LI, CM, DE, and MM

Figure 8: Robustness of TrajGuard against various attacks, where χ denotes the attacking strength.

• **MM**. In T-Drive dataset, we use IVMM [21] to map GPS points to the points on roads. GeoLife is not used because many types of human movement (e.g., hiking) cannot match to road networks.

7.2 Results

7.2.1 Robustness Comparison. Since data plagiarists can apply all types of attacks, the practicality of a scheme depends on the comprehensive robustness. To illustrate it, we firstly test the accuracy of recovered bits under attacks with the same amount of modifications as our embedding process (i.e., $\chi = 1$). Then we set the threshold of the accuracy as 85%, indicating whether a scheme is robust against a certain attack (yes when accuracy is larger than 85%, otherwise no). As shown in Table 1, TrajGuard is robust against all attacks, while FDM and DMFP have respective limitations (the detailed quantitative analysis is in Appendix Section B.1).

• **FDM** needs to transform the trajectory to frequency domain, however, changing relations among points (e.g., resampling trajectory) can make such transformation inconsistent. Consequently, it is vulnerable to SIMP, LI, CM, and hybrid attack. Moreover, FDM does not introduce secret keys to prevent others inverting the embedding process. As a result, DE can wipe out the embedded information.

• **DMFP** finds pairs of feature points of a trajectory, and then embeds information by modulating the pairwise distance. If a data plagiarist modifies the coordinates of feature points (e.g., adding noise), the embedded bit would be wiped out because of the tremendous change of pairwise distance. Thus AGNC, hybrid attack, and MM can make the embedded bits undetectable. Furthermore, applying TS or CM can change the pairing relationship of feature points. Thus the detection algorithm fails under TS attack.

In summary, TrajGuard can resist the above attacks by leveraging the stable properties of trajectory, outperforming the baselines. We further analyze the robustness of TrajGuard in detail as follows.

7.2.2 Robustness Analysis. The robustness of TrajGuard against AGNC, AGNT, SIMP, and hybrid attack is shown in Figure 8 (a)-(d), where each attack can apply various attacking strength χ. The accuracy of the recovered bits decreases slowly as the increase of χ. Even when the attacks introduce the same strength of modifications as the embedding process ($\chi = 1$), the accuracy in all cases is larger than 89%, showing strong robustness of TrajGuard. However, when $\chi = 0$ (no attack), the accuracy is less than 100%. The reason is that after scaling up/down centroid distance in the embedding process (Section 4.2), the distance between point p_i and point p_i' could larger than modification bound τ. To preserve data utility, p_i' is adjusted, resulting in errors of some recovered bits.

In Figure 8 (e), We show the robustness against RT, TS, LI, CM, DE, and MM. Specifically, RT, TS, LI, and CM cannot impact the partition algorithm or change the centroid distance, so these attacks do not take any effect. DE and MM can wipe out some embedded bits, but the accuracy is still larger than 90%. Overall, TrajGuard is robust against all these attacks. Once more, we also present the robustness of TrajGuard against attacks which adopt much larger modification with $\chi \gg 1$, as shown in Appendix Section B.2

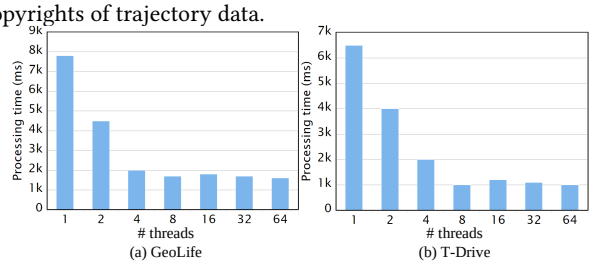
Besides robustness, we also observe the lower accuracy on T-Drive compared with GeoLife from Figure 8. The reason is that some sub-trajectories in T-Drive mainly consist of consecutive fixed points (e.g., taxi stops and waits for passengers), whose centroid distance is extremely small (e.g., $d = 0$ when the taxi did not move). It causes the arithmetic problem when scaling up/down the centroid distance, leading to the failure of the embedding process.

Although the accuracy of T-Drive is lower, it decreases much slower than that of GeoLife, showing less impact from attacks. As the introducing noise on a sparse dataset (T-Drive) is less likely to change the properties of trajectory, such as shape and velocity, the centroid distance is stable. While for a dense dataset (GeoLife), the noise can change the appearance of the trajectory (example in Section 7.5). Since the dense dataset is more sensitive to attacks, the accuracy decreases faster than the accuracy of the sparse dataset.

7.3 Evaluation on Efficiency

We show the efficiency of the embedding process, which are implemented in Java and run on a computer with 2 Intel Xeon E5-2665 (8 cores, 16 logical processors per CPU) and 128GB RAM. Figure

9 presents the efficiency of the algorithms. In the single thread manner, the processing time of both datasets is less than 8 seconds. In addition, since trajectories are independent of each other, concurrent computing can enhance the efficiency. When the number of threads is small, the increasing of threads reduces the processing time significantly. When the number of threads becomes larger, it gives adverse impact on the performance because of the limited computing resources. Overall, TrajGuard is efficient to protect copyrights of trajectory data.

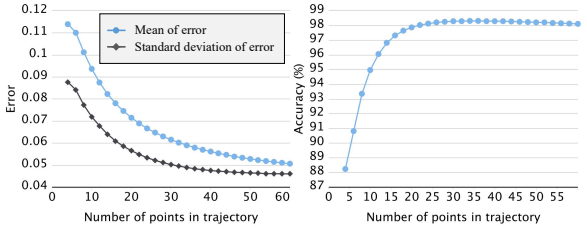


(a) GeoLife (b) T-Drive

Figure 9: Time cost of processing trajectory data.

7.4 Parameter Settings

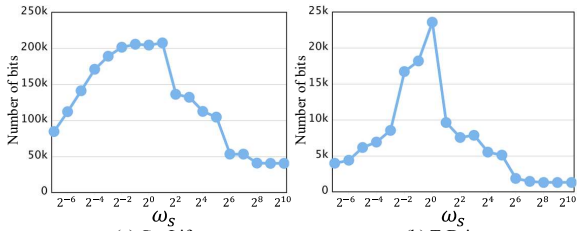
First we investigate how the accuracy is affected by the number of points n within sub-trajectory, which can be controlled by ω_t. For different n, we firstly collect every n continuous points in the original trajectories to form sub-trajectories and then randomly embed one bit into each of them. After that, we get a list of sub-trajectories $\{\mathcal{T}_i'\}$ with normalized centroid distance $\{d_i'\}$. Next, Gaussian noise (with $\chi = 1.0$) are added on all points. Finally attacked sub-trajectories $\{\mathcal{T}_i''\}$ are generated and the normalized centroid distance is $\{d_i''\}$. Figure 10 (a) depicts the statistics about introducing error of centroid distance under different n, where the error of i-th centroid distance is defined as: $error_i = |d_i' - d_i''|$. The mean and standard deviation of the errors decrease rapidly as the increase of n. As shown in Figure 10 (b), the scheme achieves good accuracy (larger than 98%) when $N \geq 30$. This guides us to choose the secret key ω_t. Suppose the sampling rate of the GPS device is r points/seconds. To reach the high accuracy by ensuring at least n points per sub-trajectory, we select ω_t by the formula: $\omega_t > \frac{n}{r}$. Thus, according to the sampling rate of the datasets, we choose $\omega_t = 300(s)$ for GeoLife and $\omega_t = 42,000(s)$ for T-Drive.



(a) Statistics of error (b) Accuracy

Figure 10: Introducing error on centroid distance and accuracy of extracted bits under different n.

ω_s is used to control the grid size. Figure 11 illustrates the number of sub-trajectories (embedded bits) produced by partition module with different ω_s. When ω_s is too small or too big, the embedding capacity is limited since many points are useless in the trajectory partition module. To embed more information, we set $\omega_s = 2$ and $\omega_s = 1$ for GeoLife and T-Drive, respectively.



(a) GeoLife (b) T-Drive

Figure 11: Number of embedding bits under different ω_s.

7.5 Case Study

A real-world example is visualized in Figure 12, showing how TrajGuard protects copyrights of trajectory data. The modification bound τ is $3e - 4$. We embed 6 bits of information into 6 sub-trajectories respectively, as shown in Figure 12 (c)-(h). The original trajectory and the embedded trajectory are almost overlapping, so the data utility (landmarks, routes, etc.) is preserved.

After embedding the bits, the hybrid attack is implemented to modify the embedded trajectory. Firstly, we randomly add 3 points between every pair of adjacent points by interpolation. Then, we apply AGNC with $\chi = 1.6$. Finally, we simplify the trajectory by $SIMP(\chi = 1.6)$. Note that the strength of the attack is much larger than the strength of the embedding process. As the red lines show in Figure 12, the modification introduces large errors which even changes the routes and landmarks of the original trajectory. However, due to the robustness of the centroid distance, only the third bit cannot be recovered correctly, which shows the advantages and practicality of the proposed scheme.

8 RELATED WORK

We study several categories of related works for protecting trajectory data, positioning our work in the research community.

Schemes based on spatial domain. [8] embeds identity information by modifying the distance between some pairs of points. [22] firstly select some feature points which can survive from the simplification and cropping. Then it embeds identity information into the distance between those feature points.

Compared with the above schemes, TrajGuard utilizes the stable properties of trajectory on both spatial and temporal domains, such that it can significantly improve the robustness against simplification, interpolation, and noise addition. Moreover, TrajGuard is robust against some complex attacks including hybrid attack, double embedding, and map matching.

Schemes based on frequency domain. [9, 10] transform data into the frequency domain and embed bits by modifying the magnitude of Fourier descriptors. This type of schemes is robust against geometry transformations and noise addition. [10] also considers the down-sampling and up-sampling attacks which sample the points with equidistant index. However, due to the limitation of Fourier descriptors, once the index is changed arbitrarily (e.g. adding points or removing points at some timestamps), the embedded information cannot be recovered.

In contrast to the schemes on frequency domain, TrajGuard models the curve of trajectories, such that it can deal with attacks which resamples the trajectories, e.g., simplification and interpolation.

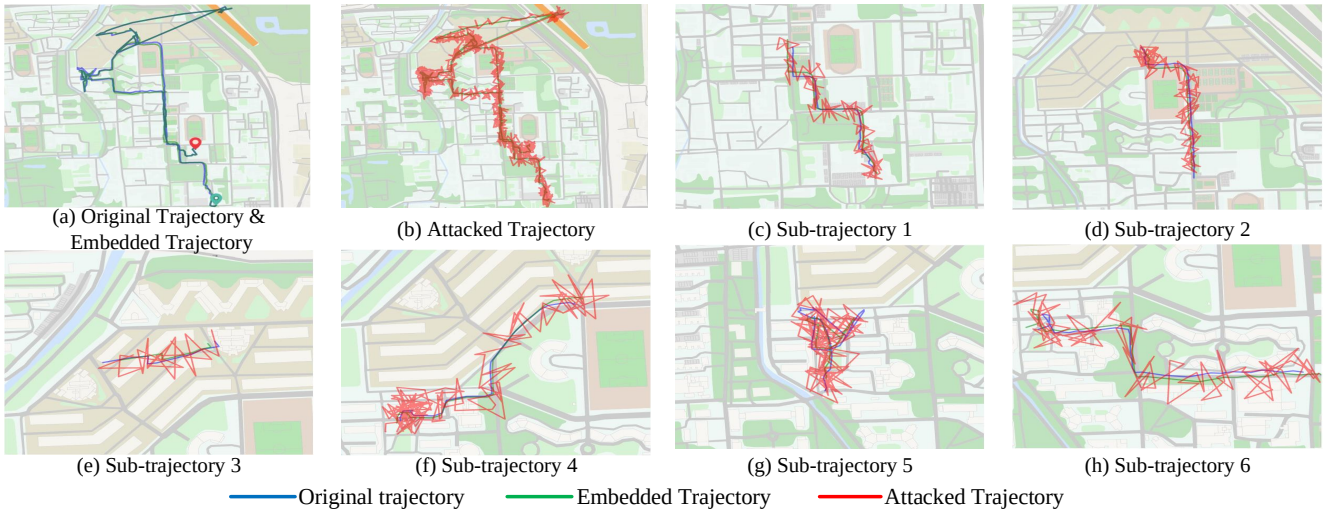


Figure 12: Example of the proposed scheme against hybrid attack ($\chi = 1.6$). For the sub-trajectories shown in the sub-figure (c)-(h), the change of the centroid distance after embedding and attacking are shown as follows: (c) $4.20 \rightarrow 4.74 \rightarrow 4.83$**, (d)** $4.87 \rightarrow 4.26 \rightarrow 4.29$**, (e)** $1.83 \rightarrow 1.31 \rightarrow 1.51$**, (f)** $3.52 \rightarrow 3.74 \rightarrow 3.79$**, (g)** $1.69 \rightarrow 1.25 \rightarrow 1.43$**, (h)** $4.12 \rightarrow 4.70 \rightarrow 4.65$**. The embedded bits are 1, 0, 0, 1, 0, 1 respectively. While the extracted bits are 1, 0, 1, 1, 0, 1. Only the third bit cannot be recovered correctly.**

9 CONCLUSION

In this paper, we propose a novel and comprehensive scheme to protect the copyrights of trajectory data. The scheme embeds the identity information, while preserving the data utility. By using the sub-trajectories and their centroid distance for embedding, our proposed trajectory copyright protection scheme is robust against a various and comprehensive set of attacks. Finally, by using the blockchain to record the data distribution transactions, the proposed scheme works without the third party to monitor and store the original trajectory data. The effectiveness of our proposed scheme is evaluated extensively based on the two real-world datasets: GeoLife and T-Drive. In the experiments, more than 89% of the embedded information are recovered correctly against any attacks, which outperforms all the baseline solutions.

ACKNOWLEDGMENTS

The work is sponsored by APEX-MSRA Joint Research Program, the National Natural Science Foundation of China Grant (61672399, U1609217, 61702327, 61772333, and 61632017), and Shanghai Sailing Program (17YF1428200).

REFERENCES

[1] Jie Bao, Tianfu He, Sijie Ruan, Yanhua Li, and Yu Zheng. 2017. Planning bike lanes based on sharing-bikes' trajectories. In *SIGKDD*. ACM, 1377–1386.
[2] Chi-Yin Chow and Mohamed F Mokbel. 2011. Trajectory privacy in location-based services and data publication. *SIGKDD* 13, 1 (2011), 19–29.
[3] 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 (2013), 1376.
[4] Sébastien Gambs, Marc-Olivier Killijian, and Miguel Núñez del Prado Cortez. 2010. Show me how you move and I will tell you who you are. In *SIGSPATIAL*. ACM, 34–41.
[5] Frank Hartung and Bernd Girod. 1998. Watermarking of uncompressed and compressed video. *Signal processing* 66, 3 (1998), 283–301.
[6] Tianfu He, Jie Bao, Ruiyuan Li, Sijie Ruan, Yanhua Li, Chao Tian, and Yu Zheng. 2018. Detecting Vehicle Illegal Parking Events using Sharing Bikes' Trajectories. *SIGKDD. ACM* (2018).
[7] Kai Jiang, Kenny Q Zhu, Yan Huang, and Xiaobin Ma. 2013. Watermarking road maps against crop and merge attacks. In *Proc. of the first ACM workshop on Information hiding and multimedia security*. ACM, 221–230.
[8] Xiaoming Jin, Zhihao Zhang, Jianmin Wang, and Deyi Li. 2005. Watermarking spatial trajectory database. In *DASFAA*. Springer, 56–67.
[9] Claudio Lucchese, Michail Vlachos, Deepak Rajan, and S Yu Philip. 2008. Rights Protection of Trajectory Datasets.. In *ICDE*. 1349–1351.
[10] Claudio Lucchese, Michail Vlachos, Deepak Rajan, and Philip S Yu. 2010. Rights protection of trajectory datasets with nearest-neighbor preservation. *The VLDB Journal* 19, 4 (2010), 531–556.
[11] Chao Ma, Jiannong Cao, Lei Yang, Jun Ma, and Yanxiang He. 2014. Effective social relationship measurement based on user trajectory analysis. *Journal of Ambient Intelligence and Humanized Computing* 5, 1 (2014), 39–50.
[12] Satoshi Nakamoto. 2008. Bitcoin: A peer-to-peer electronic cash system. (2008).
[13] Maha Sharkas, Dahlia ElShafie, Nadder Hamdy, et al. 2005. A Dual Digital-Image Watermarking Technique.. In *WEC (5)*. 136–139.
[14] Vassilios Solachidis and Ioannis Pitas. 2004. Watermarking polygonal lines using Fourier descriptors. *IEEE computer graphics and applications* 24, 3 (2004), 44–51.
[15] Melanie Swan. 2015. *Blockchain: Blueprint for a new economy*. O'Reilly Media, Inc.
[16] Xiang-Yang Wang and Hong Zhao. 2006. A novel synchronization invariant audio watermarking scheme based on DWT and DCT. *IEEE Transactions on signal processing* 54, 12 (2006), 4835–4840.
[17] Marius Wernke, Pavel Skvortsov, Frank Dürr, and Kurt Rothermel. 2014. A classification of location privacy attacks and approaches. *Personal and ubiquitous computing* 18, 1 (2014), 163–175.
[18] Fei Wu, Tobias Kin Hou Lei, Zhenhui Li, and Jiawei Han. 2014. MoveMine 2.0: mining object relationships from movement data. *VLDB* 7, 13 (2014), 1613–1616.
[19] Andy Yuan Xue, Rui Zhang, Yu Zheng, Xing Xie, Jin Huang, and Zhenghua Xu. 2013. Destination prediction by sub-trajectory synthesis and privacy protection against such prediction. In *ICDE*. IEEE, 254–265.
[20] Jing Yuan, Yu Zheng, and Xing Xie. 2012. Discovering regions of different functions in a city using human mobility and POIs. In *SIGKDD*. ACM, 186–194.
[21] Jing Yuan, Yu Zheng, Chengyang Zhang, Xing Xie, and Guang-Zhong Sun. 2010. An interactive-voting based map matching algorithm. In *MDM*. 43–52.
[22] Mingliang Yue, Zhiyong Peng, Kai Zheng, and Yuwei Peng. 2014. Rights Protection for Trajectory Streams. In *DASFAA*. Springer, 407–421.
[23] Vincent W Zheng, Yu Zheng, Xing Xie, and Qiang Yang. 2012. Towards mobile intelligence: Learning from GPS history data for collaborative recommendation. *Artificial Intelligence* 184 (2012), 17–37.
[24] Yu Zheng. 2015. Trajectory data mining: an overview. *TIST* 6, 3 (2015), 29.
[25] Yu Zheng, Quannan Li, Yukun Chen, Xing Xie, and Wei-Ying Ma. 2008. Understanding mobility based on GPS data. In *UbiComp*. ACM, 312–321.
[26] Yu Zheng, Xing Xie, and Wei-Ying Ma. 2010. GeoLife: A Collaborative Social Networking Service among User, Location and Trajectory. *IEEE Data Eng. Bull.* 33, 2 (2010), 32–39.
[27] Yu Zheng, Lizhu Zhang, Xing Xie, and Wei-Ying Ma. 2009. Mining interesting locations and travel sequences from GPS trajectories. In *WWW*. ACM, 791–800.

A SYSTEM IMPLEMENTATION

In this section, we outline the system implementations, and analyze complexity of algorithms.

The implementation of data distribution procedure is shown in Algorithm 1. It mainly consists of three stages: transaction generation (line 1-3), identity information embedding (line 4-11), and data transmission (line 12-13). The bottleneck of the efficiency is the identity information embedding process. In this process, the time complexity of trajectory partition function and the bit embedding function are linear corresponding to the number of points. Since each point in the original trajectory is processed only once by trajectory partition function and bit embedding function, the overall time complexity is $O(N)$ where N is the total number of points in trajectory dataset. In addition, as trajectories are independent from each other, we can apply parallel computing to further accelerate the algorithm.

Algorithm 1: Implementation of a data distribution procedure, that a data provider **A** distributes a trajectory dataset $\{\mathcal{T}_1, ..., \mathcal{T}_m\}$ to a user **B**.

1 **A** issues a transaction with $\mathrm{ID_A}$ and $\mathrm{ID_B}$, and broadcast the transaction information to the blockchain
2 **B** confirms the transaction, and broadcast it to the blockchain
3 The transaction is appended to the blockchain, and **A** receives a transaction ID from the blockchain, denoted as $\mathrm{ID_{Trans}}$
4 // **A** embeds $\{\mathrm{ID_A}, \mathrm{ID_B}, \mathrm{ID_{Trans}}\}$ into trajectory data
5 **for** $i \in \{1, 2, \ldots, m\}$ **do**
6 $\mathcal{T}_i^{(1)}, \ldots, \mathcal{T}_i^{(m_i)}$ = trajectory_partition($\mathcal{T}_i$)
7 **for** $j \in \{1, 2, \ldots, m_i\}$ **do**
8 get the index x, y, k of $\mathcal{T}_i^{(j)}$
9 get the embedded bit by hash($\mathrm{ID_A}$, $\mathrm{ID_B}$, $\mathrm{ID_{Trans}}$, x, y, k)
10 embed the bit into $\mathcal{T}_i^{(j)}$, and get the embedded $\mathcal{T'}_i^{(j)}$
11 get embedded $\mathcal{T'}_i$ by concatenating $\{\mathcal{T'}_i^{(1)}, \ldots, \mathcal{T'}_i^{(m_i)}\}$
12 **A** sends embedded dataset $\{\mathcal{T'}_1, \ldots, \mathcal{T'}_m\}$ to **B**
13 **B** confirms the reception and broadcast it to the blockchain

Algorithm 2: Implementation of ownership detection, that the data provider **A** validates whether the dataset $\{\mathcal{T}_1, ..., \mathcal{T}_m\}$ contains identity information $\{\mathrm{ID_A}, \mathrm{ID_B}, \mathrm{ID_{Trans}}\}$ by parameter L and α introduced in Section 5.3

1 **for** $i \in \{1, 2, \ldots, m\}$ **do**
2 $\mathcal{T}_i^{(1)}, \ldots, \mathcal{T}_i^{(m_i)}$ = trajectory_partition($\mathcal{T}_i$)
3 $S_0 = \beta_0 = 0$
4 **for** $j \in \{1, 2, \ldots, m_i\}$ **do**
5 get the index x, y, k of $\mathcal{T}_i^{(j)}$
6 get the embedded bit b by hash($\mathrm{ID_A}$, $\mathrm{ID_B}$, $\mathrm{ID_{Trans}}$, x, y, k)
7 extract one bit b' from $\mathcal{T}_i^{(j)}$
8 $S_j = S_{j-1} + \mathbf{1}_{b==b'}$
9 $\beta_j = S_j - \alpha j$
10 **if** $\min(\beta_1, \ldots, \beta_{j-L-1}) < \beta_j$ **then**
11 Claim the ownership if $\{\mathrm{ID_A}, \mathrm{ID_B}, \mathrm{ID_{Trans}}\}$ is on the blockchain

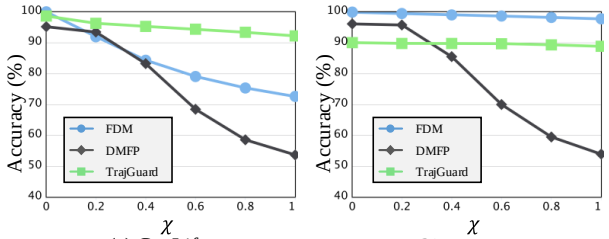
The implementation of ownership detection is shown in Algorithm 2. Being similar to the data distribution procedure, each point is processed only once by trajectory partition function and bit extraction function, and both functions are linear. Thus the time complexity of ownership detection process is $O(N)$.

B DETAILED EXPERIMENT RESULTS

B.1 Quantitative Comparison with Baselines

In this part, we make quantitative comparison for different schemes.

Comparison with FDM: As shown in Figure 13, the robustness of FDM against AGNC goes down rapidly as the increase of χ in GeoLife dataset. While in sparse dataset T-Drive, it gets the best accuracy. The reason is that FDM directly embeds the identity information into the whole trajectory on frequency domain, so the introducing error is not easy to control. Especially in the dense dataset, it will introduce larger error, resulting in the limitation of embedding strength. In contrast, TrajGurad partitions the trajectory into small pieces each of which contains one bit. Thus the introducing error is controllable and the accuracy is guaranteed. Similarly, with respect to map matching shown in Table 2, the accuracy of FDM is lower than TrajGuard with the same reason. Being different, RT does not change the magnitude of Fourier descriptor and distance between points, so both of FDM and TrajGuard are robust against RT (in Table 2).



(a) GeoLife (b) T-Drive

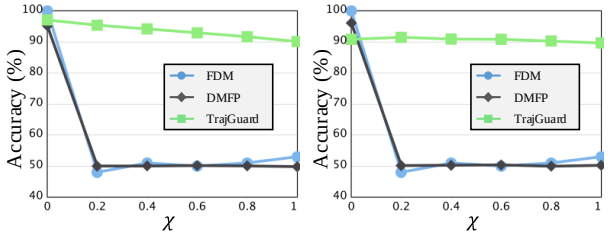
Figure 13: AGNC with different attacking strength χ.

Table 2: Robustness against RT, TS, LI, CM, DE, and MM, where N/A means the scheme is vulnerable to the attack, i.e., the accuracy of the recovered bits is less than 55%.

Scheme	Dataset	RT	TS	LI	CM	DE	MM
FDM	GeoLife	100%	100%	N/A	N/A	N/A	-
	T-Drive	100%	100%	N/A	N/A	N/A	85.7%
DMFP	GeoLife	100%	N/A	100%	N/A	93.3%	-
	T-Drive	100%	N/A	100%	N/A	94.9%	62.3%
Traj-Guard	GeoLife	98.6%	98.6%	98.6%	98.6%	91.4%	-
	T-Drive	90%	90%	90%	90%	89.8	90.1%

As for the attacks on temporal domain, i.e., AGNT and TS, because FDM does not consider the temporal information, these attacks cannot take effect (as shown in Figure 14 and Table 2).

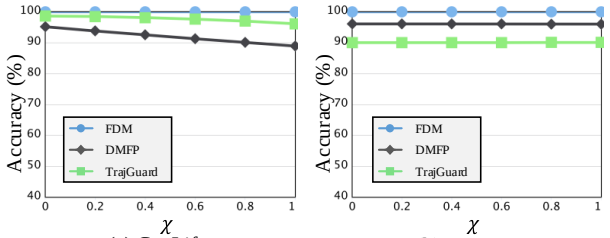
For the attacks which change the set of points, i.e., SIMP, LI, and CM, FDM fails because the Fourier descriptors become totally different. While TrajGuard utilizes the centroid distance which considers all estimated points on trajectory. So these attacks can be handled by TrajGuard.



(a) GeoLife (b) T-Drive

Figure 16: Hybrid attack with different attacking strength χ.

Besides, DE is also not supported by FDM because data plagiarist can directly embed his identity information into the trajectory to modify Fourier descriptors. In contrast, as data plagiarists do not know how data provider partition the sub-trajectories in TrajGuard, this attack can be handled.



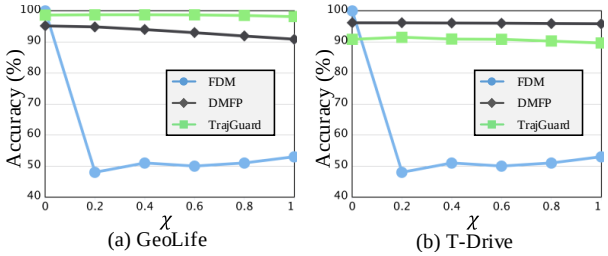
(a) GeoLife (b) T-Drive

Figure 14: AGNT with different attacking strength χ.

Comparison with DMFP: First, we discuss the robustness of DMFP against spatial attacks. Since DMFP simply embeds bits by modifying some pairs of feature points, it is vulnerable to the additive noise on coordinate, resulting in the low robustness against AGNC and MM, as shown in Figure 13 and Table 2. Whereas RT does not change the distance between points, so DMFP is robust against RT (in Table 2).

For AGNT on temporal domain, as both the pairing relations of feature points and the pairwise distance are not changed, AGNT cannot take effect (as shown in Figure 14). While for TS, the timestamps are shifted, which changes the selected feature points, so DMFP is vulnerable under this attack (as shown in Table 2).

For the attacks which change the set of points, DMFP is robust against SIMP and LI (in Table 2 and Figure 15), because the feature point selected by DMFP is those points which furthest away from the reference lines, most of the feature points cannot be removed (otherwise the utility is not preserved). However, CM can crop or add fragments of trajectories, which can break the feature point selection of DMFP. Thus DMFP is not robust against CM.



(a) GeoLife (b) T-Drive

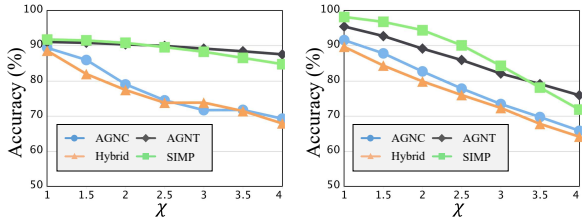
Figure 15: SIMP with different attacking strength χ.

At last, as DMFP also applies secret keys to find feature points such that data plagiarists do not know those points, DE can be handled by DMFP.

Overall Comparison: As what we discussed above, both FDM and DMFP have their own limitations, while TrajGuard is robust against all these attacks, that we achieve over 89% accuracy in all scenarios. In addition, we plot the scheme performance under hybrid attacks shown in Figure 16, showing comprehensive robustness of TrajGuard beyond baselines. Thus TrajGuard is a more practical scheme for trajectory copyright protection in the real world.

B.2 Advanced Robustness of TrajGuard

We also show the robustness of TrajGuard in extreme cases. Figure 17 illustrates the robustness against attacks which introduce much larger modification than the embedding process, i.e., $\chi \gg 1$. When data plagiarists apply attacks with bound 4τ, which totally change the appearance of trajectories, more than 64% bits can still be correctly recovered. Since the accuracy of randomly guessing the embedded bits is 50%, TrajGuard still shows a certain degree of robustness in such cases.



(a) GeoLife (b) T-Drive

Figure 17: Advanced robustness of TrajGuard.