PPTPF: Privacy-Preserving Trajectory Publication Framework for CDR Mobile Trajectories

Jianxi Yang 1, Manoranjan Dash 2 and Sin G. Teo 3,*

1 School of Information Science and Engineering, Chongqing Jiaotong University, Nanán District, Chongqing 400074, China; yjx@cqjtu.edu.cn
2 Data Science Consortium, National University of Singapore, Singapore 117602, Singapore; docsman@nus.edu.sg
3 Cybersecurity Department, Institute for Infocomm Research, Singapore 138632, Singapore
* Correspondence: teosg@i2r.a-star.edu.sg

Abstract: As mobile phone technology evolves quickly, people could use mobile phones to conduct business, watch entertainment shows, order food, and many more. These location-based services (LBS) require users’ mobility data (trajectories) in order to provide many useful services. Latent patterns and behavior that are hidden in trajectory data should be extracted and analyzed to improve location-based services including routing, recommendation, urban planning, traffic control, etc. While LBSs offer relevant information to mobile users based on their locations, revealing such areas can pose user privacy violation problems. An efficient privacy preservation algorithm for trajectory data must have two characteristics: utility and privacy, i.e., the anonymized trajectories must have sufficient utility for the LBSs to carry out their services, and privacy must be intact without any compromise. Literature on this topic shows many methods catering to trajectories based on GPS data. In this paper, we propose a privacy preserving method for trajectory data based on Call Detail Record (CDR) information. This is useful as a vast number of people, particularly in underdeveloped and developing places, either do not have GPS-enabled phones or do not use them. We propose a novel framework called Privacy-Preserving Trajectory Publication Framework for CDR (PPTPF) for moving object trajectories to address these concerns. Salient features of PPTPF include: (a) a novel stay-region based anonymization technique that caters to important locations of a user; (b) it is based on Spark, thus it can process and anonymize a significant volume of trajectory data successfully and efficiently without affecting LBSs operations; (c) it is a component-based architecture where each component can be easily extended and modified by different parties.

Keywords: spatio-temporal data; privacy-preserving; utility; k-anonymization

1. Introduction

The primary purpose of mobile phones is to keep people connected. The number of mobile phone users increased from 4.3 billion in 2016 to 4.8 billion in 2020 [1]. People use mobile phones to conduct business, watch entertainment shows, order food, and many more. Therefore, mobile phones have become an invaluable source of data to study various aspects of human society [2–4], such as human mobility patterns. Investigating human mobility patterns is essential for urban planning and monitoring, transportation infrastructure optimization, etc.

One way to collect human mobility data are to use GPS-enabled mobile phones. However, penetration of GPS-enabled mobile phones is still reasonably low in developing and underdeveloped countries [1,5–8]. An alternative is to use Call Detail Record (CDR) information in place of GPS. CDR data collection infrastructure has been in place, and telecommunication operators collect CDR data for billing purposes. Hence, it incurs no extra cost or overhead and is available both for GPS-enabled and non-GPS-enabled mobile phones. Unlike CDR data, we need to turn on location services in the GPS-enabled mobile...
phone for capturing GPS location information. In contrast, the CDR data are automatically recorded by the telecommunication operators when a user initializes a call or sends a text message. These advantages of CDR data motivate us to study it for a human mobility trajectory in this paper.

Latent patterns and behavior are hidden in trajectory data that can be extracted and analyzed to improve location-based services (LBS). For example, location-based service (LBS) providers can use the location data to provide a guide in planning more effective and efficient route by avoiding roads with serious traffic problems, find places of interest (recommendation), give background support for efficient urban planning, assist in smooth traffic flow, and so on. While LBSs offer relevant information to mobile users based on their locations, revealing such locations to 3rd party providers can pose user privacy violation problems as stipulated by HIPAA [9] and GDPR [10].

Unlike other types of data that require privacy preservation, the time and location information in trajectory data are inherently highly correlated for obvious reasons, and thus can easily give away the identity of the user. de Montjoye et al. estimated that just four pieces of spatio-temporal information of an anonymous user may reveal his identity with 95% probability [11].

Many privacy-preserving algorithms [12–14] using location obfuscation or masking techniques fail to provide useful utility. In Ref. [12], authors proposed to use count data by first converting spatio-temporal data to the count data. For a given time interval, the number of users present is counted at each lat-long. Subsequently, the authors applied differential privacy on the count data. Several other methods [14,15] have also been proposed based on the count data. However, a fallout of using the count data are that common patterns among trajectories are lost due to aggregation leading to a severe utility loss. For example, given two trajectories, $tr_1$ and $tr_2$, the locations of $tr_1$ are $o_1, o_2, \ldots, o_m$, and $p_1, p_2, \ldots, p_n$ for $tr_2$. Assume that $o_1, o_2, \ldots, o_i$ and $p_1, p_2, \ldots, p_j$ belong to the same cluster, where $i$ and $j$ are less than $m$ and $n$, respectively. To anonymize the locations in the cluster, the count data approach aggregates all the user locations, as stated before. This way of doing anonymization may lose some basic location patterns leading to a utility loss and not applicable in some trajectory applications. In this paper, we propose a privacy-preserving trajectory publication framework for CDR (PPTPF) that can solve this issue in which our published trajectories can still retain useful utility while preserving user trajectory privacy.

The above discussed algorithms [11,12,14,16] are developed to run in a single machine with limited disk and memory space that cannot handle a significant data volume. In recent years, various devices (e.g., RFID sensors, GPS, satellite, and wireless communication technologies) have been used to track and record trajectories of vehicles, people, and many more. With time, it grows without control, leading to difficulties in storing, processing, and analyzing such trajectory data. Most of the existing methods only run in a single machine, thus facing a severe bottleneck in processing and handling such large trajectory data. Again, our proposed PPTPF based Spark can solve the disk and memory limitations of a single machine, i.e., PPTPF stores big trajectory data in Hadoop distributed file system (HDFS) and then processes the data from HDFS.

Our proposed PPTPF consists of five components (i.e., a sorting component, stay region extraction component, trip consolidation component, trip anonymization component, and trip publication component). Each of the components with its salient features is discussed in Section 3. The proposed components can be extended and modified easily in PPTPF. It uses $k$-Trip anonymization to anonymize user trips before publishing them. This technique can be replaced by other anonymization techniques easily. This component-based feature contrasts with the many existing algorithms [11,12,14,16] that aim for a specific purpose, making them unfit to be reused for others without substantial modifications.

In this paper, our main contributions are listed as follows:

- Our framework (PPTPF) can preserve user trajectory privacy while maintaining the mobility patterns of the users. Note that, unlike other works on this topic, we handle
CDR data (not GPS data), which is a novelty by itself. In addition, the combination of techniques we propose to deal with CDR data are unique as well. For example, we use stay regions (Figure 1) for estimating a source and destination of any trajectory and the Markov model for estimating a representative trajectory among a group of trajectories between a given source and destination, etc.

- The framework based on Spark can process and anonymize a significant volume of trajectory data.
- The framework contains five components that each can be modified and extended easily without significant modifications. In addition, the security analysis and time complexity of each component are discussed in this paper.
- A unique method for $k-1$ Trip anonymizing the trajectories is proposed in the framework. First of all, it extracts short yet meaningful trajectories for each user. Next, the framework will apply $k-1$ trip anonymization over these trajectories to determine $k-1$ anonymized clusters.

The rest of the paper is organized as follows. We discuss the related work in the next section. The privacy-preserving trajectory publishing framework (PPTPF) for moving object trajectories is discussed in Section 3. Performance evaluation and discussions are in Section 4. Finally, conclusions and future work of this paper are presented in Section 5.

**2. Related Work**

In recent years, different privacy-preserving publishing techniques [17–23] have been proposed to anonymize micro-data stored in a statistical and tabular form by reducing their disclosure. In $k$-anonymity [17–20], quasi-identifiers of a record are indistinguishable from at least other $k-1$ records in the dataset (i.e., each equivalent class must contain least $k$ records of the dataset). The limitation of $k$-anonymity is that all of the class records have less than $k$ values for any sensitive attributes that cannot be guaranteed in the $k$-anonymity method. This issue is solved by the $\ell$-diversity [21,22], which ensures that each equivalence class of the $k$-anonymity has at least $\ell$ values of the sensitive attributes. Subsequently, $t$-closeness [23] has been proposed to improve data privacy protection by the $\ell$-diversity method. This $t$-closeness ensures that each equivalence class of the $\ell$-diversity with $t$-closeness, which is calculated by the distance between two distributions, e.g., the distance between the distribution of sensitive attributes each quasi-identifier group as well as their distribution in the entire dataset. All of the privacy-preserving publishing techniques above are given in detail in the survey paper [24]. The above-discussed methods are not applicable in preserving user privacy of spatio-temporal data.

Several existing works [25–28] have used different approaches to apply privacy-preserving methods on the location and trajectory data. Many of them use GPS-enabled devices to record mobility trajectories and then they are anonymized to preserve their...
location privacy before releasing the anonymized trajectory to the public. Ref. [25] proposed a
$k$-anonymity method to anonymize trips generated by vehicles. Their trip trajectories
are usually relatively short in length. The technique is targeted in anonymizing only
short trajectories. It differs from our proposed framework that can anonymize longer
trajectories generated from the CDR location data. Ref. [26] proposed a framework to
protect the worker location privacy in Spatial Crowdsourcing (SC) using a differential
privacy technique. This technique adds noise to worker locations and then the noisy
locations of the workers are submitted to the non-trusted SC server. Hence, it learns
nothing about the real locations of the workers. Authors [27] proposed a privacy scheme
(PCANNQ) based on spatial $k$-anonymity method to protect the location and trajectory
privacy of the groups of users in the continuous aggregate nearest neighbor query service
provided by the Internet of Things (IoT). In other words, anyone who uses the service
learns nothing about the user real locations and trajectory paths. The authors use entropy
under different $k$ values to measure location privacy. For trajectory security, they measure
a ratio of the actual number of query requests to the total number of query requests in
the service. The security is guaranteed if the ratio meets the threshold set by the authors.
The work [28] proposed a privacy-preserving trajectory publication method based on
generating start-points and end-points of the trajectories. This method uses two-way
dummy algorithms to generate $k - 1$ anonymous trajectories from the real trajectories,
in which the anonymous ones could maintain real trajectory similarity while preserving
user location privacy. Our work is similar to [28] that uses a $k - 1$ anonymity method
to anonymize trajectories and then release them to the public. However, we primarily
use CDR data for trajectories in the proposed framework instead of using GPS data as
in [28]. We use a large CDR dataset (containing 420,744,849 location points) to evaluate the
performance of the proposed framework, whereas just 1 million GPS location points were
used to evaluate the method of the paper [28]. In addition, authors [28] simply use various
$k$ values in the method to measure utility loss and trajectory leakage on the generated
anonymous trajectories with their real ones. In contrast, we propose two measurements,
discernibility and distortion, that could give much better measurements in terms of the
utility loss and trajectory leakage as compared to just simply $k$ values. Furthermore, all
the above privacy-preserving methods [25-28] were not discussed the way to handle and
anonymize voluminous location and trajectory data generated by GPS-enabled devices.
Our proposed framework can handle voluminous location and trajectory data generated
by call data record (CDR) information.

Refs. [29,30] proposed their methods to fake database results based on a user query.
However, the results are related to the query location. Refs. [31,32] used a space transforming
method to convert the data and query while preserving their inter-relationship. Instead of
faking database results, Ref. [33] proposed a method to fake the query location. As a result,
the database server keeps sending a resultant site to the user until the user is satisfied. Some
trajectory anonymization methods convert spatial-temporal data from CDRs into count
data, e.g., in [4], several users are partitioned based on their spatio-temporal whereabouts.
These spatio-temporal whereabouts are used in their cell tower lat-longs. This technique can
significantly reduce data utility after applying the conversion. In particular, information
about the sequence of the cell towers visited by a user cannot be captured. Hence, the flow
of mobility is lost. Instead, our proposed framework based on $k$-anonymization can
maintain utility much better while preserving user privacy.

In [7,34], authors proposed a method to estimate travel time between cities based
on CDRs that rely not on individual trajectories of people, but their collective statistical
properties. Compared to this paper, we deal a very different task, but there is a similarity
of grouping CDR trajectories in order to estimate statistics such as starting and ending
times, etc. Another similar implicit assumption between our work here and these papers is
that phone calls (or SMS) are correlated to actual travel times. In these papers, the main
motivation is to significantly increase low coverage and penetration rate vis-à-vis earlier
methods that are based on GPS data.
3. Privacy-Preserving Trajectory Publishing Framework (PPTPF)

Moving object trajectories of users contain spatio-temporal data. The trajectory data can reveal private and sensitive information of a user, e.g., places visited by the user as part of his or her daily routine. We propose a Privacy-Preserving Trajectory Publishing Framework PPTPF (Figure 2) that can (i) anonymize user trajectories without privacy violation as stated before, (ii) publish anonymized trajectories that still contain useful information for data analytics, and also (iii) handle a big volume of trajectory data. We first discuss some important terms used in this paper.

Figure 2. Privacy-preserving trajectory publishing framework.

**Trajectory data:** In this paper, the trajectory data contain features such as an identifier (id), timestamp, latitude (lat) and longitude (lon), and many more. We focus on four essential features (id, timestamp, lat, and lon) of the trajectory data that can quickly reveal private and sensitive information of a user. One way to retrieve the user location data are to use some cell towers that render required services when an event (e.g., phone/SMS call/receive) occurs. These location data of the users are also known as call detail records (CDRs).

**Stay Region:** A stay region is a region that has an area bound by a threshold radius ($SPT$) around a center point defined by lat and lon, and where a user spends more than a threshold amount of time ($TMT$). Spatial threshold $SPT$ helps to cluster cell towers in close proximity to each other. Temporal threshold $TMT$ discriminated events within the stay region from those on transit. Both thresholds are user-defined values in the proposed framework (PPTPF).

Figure 1 shows stay regions on two different cases. In Case 1, A, B, C, and D are cell towers. Let $j$th be a stay region of a user $u_i$ denoted as $r_{ij}$. Center point of the stay region is $r_{ij}.lat$ and $r_{ij}.lon$. Any cell tower within an $SPT$ distance from $r_{ij}.lat$ and $r_{ij}.lon$ belongs to the stay region. Case 1 shows that only towers A and D can satisfy both the thresholds ($SPT$ and $TMT$), whereas B and C cannot meet those conditions. Therefore, the circles around towers A and D are the stay regions. In another case, Case 2, with similar settings as in Case 1, only towers A and E can satisfy the thresholds ($SPT$ and $TMT$), whereas B, C, and D fail to meet the conditions. Therefore, the circles around towers A and E are the stay regions. Note that a stay region can cover more than one cell tower.

**Trip:** A trip is traversed from one stay region to another, where a user spends a significant amount of time in the stay regions. In other words, stay regions are not transit locations traversed by the user. Note that, in a single day, a user can make one or more trips. Some examples of stay regions are home, workplace, friend-place, shopping mall, gymnasium, and many more. Obviously, some stay regions are regularly visited, whereas others are visited sporadically or just once.

Next, we discuss details of the proposed framework, PPTPF, to anonymize trajectories before publishing them. PPTPF consists of five components (i.e., the Sorting Component, Stay Region Extraction Component, Trip Consolidation Component, Trip Anonymization Component, and Trip Publication Component) that can run in Apache Spark. These five
components use a modular approach that allows each of them to be extended by third parties easily and quickly. This section is organized as follows: the features of PPTPF are discussed in Section 3.1. A summary of the PPTPF and evaluation metrics is discussed in Sections 3.2 and 3.3, respectively.

3.1. Components of the PPTPF

In this section, each component of PPTPF is discussed. Before that, a way to form trajectories using CDR is discussed.

**Convert Call Detail Records (CDRs) to Trajectories:** To preserve privacy of a trajectory over several months or weeks or even days as one trajectory is considered a challenging problem. Note that long trajectories cannot capture good mobility patterns of a user. We also observe that even trajectories using a day are too long to preserve privacy [11]. It then becomes very unique, leading to difficulties in satisfying $k$-anonymization conditions. Thus, a trajectory is a trip that starts and ends with a stay region. Typically, a user can repeat his trips or trajectories in a weekly manner. One good example is going from home to office and returning from office to home on weekdays, whereas going to a shopping mall and other places (except the office) on weekends. Each of these (a. going to office from home, b. going home from office, c. going to shopping mall from home, etc.) is a trajectory. Thus, the task is reduced to $k$-anonymization of these trips or trajectories.

A user trajectory usually contains many locations. Each location consists of $<ID>$, $<Timestamp>$, $<Lat>$ and $<Lon>$, where $ID$ is identifier of the user, $Timestamp$ is a visited date and time of the user at that location, and $Lat$ and $Lon$ are coordinates of that location. As mentioned previously, all trajectories are stored in a Hadoop Distributed Filesystem (HDFS).

**PPTPF: Sorting Component.** The sorting component can sort the locations of each user by his own timestamps. Each location can be presented using class Location (In PPTPF, a dataframe of the Apache Spark is a dataset to represent locations and then transform it into case class Location) as shown in List 1. The sorting algorithm is straightforward, as listed in Algorithm 1.

**List 1: Case classes are in PPTPF.**

```scala
1 case class Location (Id: String, Dt: Timestamp, Lat: Double, Pos: Long)
2 case class StayRegion (StartRegion: Location, EndRegion: Location, CentroidLat: Double, CentroidLon: Double)
3 case class Trip (StartTrip: StayRegion, EndTrip: StayRegion, IntermediateTrip: Array[Location])
```

**Algorithm 1: Sorting Component in PPTPF.**

```scala
Input: A—a list of Locations
Output: B—a list of (uid, sorted array of Locations)

1 foreach A do
2 | assign rid to each location, $\eta_i \leftarrow (rid, A_i)$
3 $G \leftarrow \eta$ group by uid
4 foreach G do
5 | $G' \leftarrow G(timestamp, lat, lon).zipped$\textsuperscript{1}
6 | $G'' \leftarrow (\eta_j(rid), (G'(timestamp, lat, lon)).zipped$\textsuperscript{1}
7 $S \leftarrow G''$ order by $timestamp$
8 foreach S do
9 | $S_k' \leftarrow Location(G_j(uid), S(timestamp, lat, lon))$
10 $B_j \leftarrow (G_j(uid), S')$
11 return B
```
This is just a concise presentation. The actual way to retrieve is $G_{rt}(Timestamp), G_{rt}(Lat), G_{rt}(Lon)$. An Apache Spark Zip function can combine two or more data lists into one list.

Algorithm 1 collects all the locations traversed by users as shown in lines 5 and 6. At the end of this algorithm, the output is the list of the locations traversed by the users that are sorted based on their individual traversed timestamps. This output is then input into the next component of the PPTPF.

**Time Complexity:** This component can sort the location data of different users in parallel by relying on available resources such as several processor cores and the available machine memory and disk space. With running on a single machine, the time complexity of this component is $O(n \log n)$.

**PPTPF: Stay Region Extraction Component.** The task of extracting the stay regions of this component is discussed as follows. For example, given $\eta$ locations of a user, this component needs to obtain stay regions for the user. The extracted stay regions need to meet the spatial threshold SPT and temporal threshold TMT conditions. The following Haversine Equation (1) [35] is to calculate a Euclidean distance (ED) (in km) between two locations traversed by a user is less than or equal to $SPT$. It first calculates a centroid (Equation (2)) as shown in lines 4 and 5 of Algorithm 2. Otherwise, the time spent between the two locations is solved by allowing time spent in a stay region based on a consecutive location within the cell towers that belongs to the same stay region. At the end of Algorithm 2, the output is a list of stay regions of the user. This output is then input to the next component of the PPTPF.

**Time Complexity:** This component can run in parallel to construct stay regions of the locations traversed by the users. The performance is based on available resources such as several processor cores and the available machine memory and disk space. With running on a single machine, the time complexity of this component is $O(n)$. 

\[ \Theta = \sin^2 \left( \frac{\text{lat}_2 - \text{lat}_1}{2} \right), \quad \Phi = \cos(\text{lat}_1) \times \cos(\text{lat}_2) \times \Theta, \]
\[ \alpha = 2 \times \tan^{-1} \left( \sqrt{\Theta + \Phi}, \sqrt{1 - (\Theta + \Phi)} \right), \]
\[ \text{ED}(\ell_1, \ell_2) = \alpha \times R, \quad (1) \]

where \((\text{lat}_1, \text{lon}_1)\) and \((\text{lat}_2, \text{lon}_2)\) are the coordinates of the locations \(\ell_1\) and \(\ell_2\), respectively, \(\alpha\) is the angle between two locations \(\ell_1\) and \(\ell_2\), and \(R\) is the earth radius. We need to convert a coordinate degree into radian in the above Equation (1). This component calculates a new centroid of a stay region using a weighted average technique as follows:

\[ \text{Centroid}(\text{lat}', \text{lon}') = \left( \frac{\sum_{p=1}^{q} \text{lat}_p}{q}, \frac{\sum_{p=1}^{q} \text{lon}_p}{q} \right), \quad (2) \]

where \(q\) is the number of locations in the stay region and \(p\) is the current location.

In Algorithm 2, the Euclidean distance (Equation (1)) between two locations traversed by a user is less than or equal to $SPT$. It first calculates a centroid (Equation (2)) as shown in lines 4 and 5 of Algorithm 2. Otherwise, the time spent between the two locations is greater than the TMT. The locations are then used to construct a stay region, as shown in lines 6 and 9 of Algorithm 2. The following will be addressed in this component. The first and last user locations can substitute consecutive locations of the user within the same cell tower. Another case is when the user has two consecutive locations with different cell towers, the time the user spent in each cell tower cannot be determined. This problem is solved by allowing time spent in a stay region based on a consecutive location within the cell towers that belongs to the same stay region. At the end of Algorithm 2, the output is a list of stay regions of the user. This output is then input to the next component of the PPTPF.
Algorithm 2: Stay Region Extraction Component in PPTPF.

Input: $\beta$—the sorted list of Locations of a user

Output: $M$—a list of StayRegion of the user

1 /* Assume that $\beta$ is not empty */
2 $\Gamma \leftarrow \beta_1$
3 $h \leftarrow 2$, $s \leftarrow 1$, $j \leftarrow 1$
4 while not EOF($\beta$) do
5 if $ED(\beta_h$.$lat$, $\beta_h$.$lon$, $\Gamma$.$lat$, $\Gamma$.$lon) \leq SPT$ then
6 $(\Gamma$.$lat$, $\Gamma$.$lon) \leftarrow \text{Centroid}(\beta_s$,$\cdot$,$\cdot$,$\cdot$, $\beta_h$.$lat$, $\beta_s$,$\cdot$,$\cdot$,$\cdot$,$\cdot$,$\cdot$,$\cdot$,$\cdot$,$\cdot$)
7 else if $\beta_h$.$dt - \Gamma_{h-1}$.$dt > TMT$ then
8 $\Gamma$.pos $\leftarrow s$
9 $M_j \leftarrow \text{StayRegion}(\Gamma, \beta_h, \Gamma$.$lat$, $\Gamma$.$lon)$
10 $s \leftarrow h + 1$, $j \leftarrow j + 1$
11 $h \leftarrow h + 1$
12 return $M$

PPPF: Trip Consolidation Component. This component helps to form user trips, which are trajectories between the stay regions. The details of the trip consolidation component are depicted in Algorithm 3, with the additional case classes (List 2). The input to the algorithm is the list of the StayRegions and the locations of a user.

List 2: Other case classes are in PPTPF.

1 case class UserTrips (TripIdentifier: String, TripList: Array[Trip])
2 case class AnonymizeTrips (TripIdentifier: String, TripList: Array[Trip],
TotalUser: Long)

Algorithm 3: Trip Consolidation Component in PPTPF.

Input: $M$—a list of StayRegions of a user, $\beta$—a list of Locations of a user

Output: $C$—a list of the UserTrips

1 foreach $M$ do
2 beginTrip $\leftarrow M_i$.$endRegion$
3 endTrip $\leftarrow M_{i+1}$.$startRegion$
4 for $j \leftarrow \text{beginTrip}.pos$ to $\text{endTrip}.pos$ do
5 intermediateTrip$_k = \beta_j$
6 $k \leftarrow k + 1$
7 $t \leftarrow \text{Trip(beginTrip,endTrip,intermediateTrip)}$
8 $c$.tripIdentifier$.tripList.add(t)
9 $i \leftarrow i + 1$
10 return $ht$

A trip identifier consists of Lat and Lon of the trip beginning and ending, as shown in Algorithm 3. All the user trips are stored in a list, as shown in line 8 of Algorithm 3. At the end of this algorithm, the output is the consolidated trips of the user. All the consolidated trips (i.e., UserTrips) are then input into the next component of the PPTPF.

Time Complexity: Again, this component can run parallel to construct user trips from the stay regions and locations traversed by the users. The performance is based on available resources such as several processor cores and the available machine memory and disk space. With running on a single machine, the time complexity of this component is $O(n^2)$.

PPPF: Trip Anonymization Component. This component is to anonymize the user trips from the last part. In this paper, this component uses the existing $k$-anonymity [36] to anonymize the trips. Two anonymization methods are discussed in the following.
Method 1: This method finds at least \( k - 1 \) Trips using some pre-defined hierarchies of distance and time. This method is first to generalize each trip to be indistinguishable from \( k - 1 \) other trips at least. A trip has information about stay regions and intermediate locations with beginning and end times of the trip. For example, hierarchical temporal and spatial features can be used to form \( k - 1 \) trips. Given that the beginning time of a trip is 1 p.m., setting an interval of 1 h, i.e., 1:00 p.m.–2:00 p.m., all the user trips within this interval are put together. Similarly, the trips are also put together based on their ending times. Another spatial hierarchy uses radius or specific geographical regions. For example, given that a location (i.e., lat-lon) with some pre-defined radius (e.g., 0.5 km or 1 km) is a region, all the user trips within the region are put together.

Method 2: This method finds exactly \( k - 1 \) trips using some pre-defined hierarchies of distance and time. In the previous method, each anonymized trip contains at least \( k \)-trips. It may result in some trips for which we fail to find \( k - 1 \) trips for anonymization. To solve/alleviate this problem, Method 2 tries to find exactly \( k \)-trips for anonymization. Obviously, this method can create more anonymized trips as compared to Method 1. However, the privacy loss in this method is much higher than in Method 1.

Steps to anonymize \( k - 1 \) trips in Methods 1 and 2 are as follows.

**Step 1** Source (lat-lon) of a trip is initialized.

**Step 2** The trip with the closest distance between the source lat-lon and origin lat-lon is selected.

**Step 3** The trip with the closest beginning time is selected based on Step 2. The trip with the closest destination distance is chosen first, and then the nearest ending time.

**Step 4** Finally, the selected trip is added as an anonymized trip.

The above steps are repeated until exactly \( k - 1 \) trips (Method 2) or at least \( k - 1 \) trips (Method 1) are found.

The details of this proposed component are depicted in Algorithms 4 and 5. The input to this component is the list of the consolidated user trips from the last part. In Algorithm 4, \( v \) and \( \ell \) are user given values, where \( v \) indicates the number of iterations and \( \ell \) indicates the number of split data partitions, e.g., each split data partition is sent to find \( k - 1 \) trips in each processor core (using Algorithm 5). Obviously, after each iteration, some of the trips in \( \ell \) data partitions can not form \( k - 1 \) trips in the processor cores, i.e., the trips can not meet the \( k - 1 \) trip conditions. The unused ones in one processor core could probably form \( k - 1 \) trips with other new trips in other processor cores in the next iteration. Hence, this problem can be easily solved by only increasing the number of iterations (\( v \)). In other words, all of the unused trips in the previous iterations can be gathered again and then processed in the current iteration. At the end of this component, the output is a list of the anonymized trips, where each trip consists of at least \( k \) trips (Method 1) or exactly \( k \) trips (Method 2). This output is then input to the next component of the PPTPF for trip publishing.

**Time Complexity:** Again, this component can run parallel to construct \( k - 1 \) trips provided by the user trips list. The performance is based on available resources such as several processor cores and the available machine memory and disk space. With running on a single machine, the time complexity of this component is \( O(n^2) \).

**Security Complexity:** Each anonymized trip contains at least \( k \) trips. In other words, this anonymized trip is indistinguishable from at least \( k - 1 \) trips. Furthermore, all the user identifiers have been removed in the anonymized trips. Therefore, this component can preserve user privacy, such as locations and dates and time traversed. All of the trips that have not met the \( k - 1 \) trips conditions are truncated. Some applications need to know the number of users in the anonymized trip. For example, in the COVID-19 Trace App, some government agencies need to learn the number of people infected with COVID-19 in some anonymized trips. Our proposed component is allowed to capture user count data with some privacy loss in the anonymized trip. The security analysis of Method 2 is similar to Method 1, as discussed above. Hence, we skip it in this paper.
Algorithm 4: Trip Anonymization Component in PPTPF.

Input: \( C' \) — a list of the UserTrips of all of the users
Output: \( T \) — a list of AnonymizeTrips

```plaintext
1 /* \( v \) and \( \ell \) are user-defined values */
2 for \( 1 \) to \( v \) do
3     \((C', T') \leftarrow \text{call Algorithm 5 (partition} \ell \text{ parts of } C')\)
4     \( T \leftarrow T + T' \)
5 return \( T \)
```

Algorithm 5: Sub-function of Trip Anonymization Component in PPTPF.

Input: \( D \) — a list of UserTrips
Output: \((D', E)\) — a list of (UserTrips, AnonymizeTrips)

```plaintext
1 \( D' \leftarrow D \)
2 for \( i \leftarrow 1 \) to \( \text{size}(D') \) do
3     /* Finding \( k-1 \) Trips using either Method 1 or Method 2 */
4     \((D', E') \leftarrow k-1 \text{Trips of } D' \text{ against all } D'_{\neq i}\)
5     if \( E' \) meets \( k-1 \) Trip conditions then
6         \( E \leftarrow E + E' \)
7         \( i \leftarrow 1 \)
8 return \((D', E)\)
```

PPTPF: Trip Publication Component. This final component of the proposed PPTPF is to create representative trips from the list of \( k \)-anonymized trips. Many existing algorithms [15,37,38] can be used for this purpose. Some of these algorithms allow PPTPF to create a representative trip based on Markov Model [15,37] from the anonymized trips. These trips may reveal some user location information. Hence, we propose a metric to calculate distortion between the representative trip and the list of the anonymized trips. We will discuss the distortion metric in detail in Section 3.3. The higher \( k \) is in the anonymized trips, the higher the distortion is in the representative trip. As a result, privacy loss decreases in the representative trip. In the following, we discuss trip construction based on the Markov model.

Representative Trip Construction: Let \( o_d \) and \( d_r \) be the origin and destination of the anonymized trips. First, all the trips with the same \( o \) and \( d \) are used to calculate a representative trip by matching all trip timestamps. In addition, all of their intermediate trips are taken into consideration as well. The frequency at each location is calculated based on a radius (e.g., 250 m or 500 m). For example, using a radius of 500 m, all intermediate locations within this radius are included.

The above representative trip construction problem can be formulated as a graph problem. Let all anonymized trips with the same \( o \) and \( d \) be vertices of a directed graph \( G \). We use graph edge to indicate the direction from one vertex to another. The weight of the edge will be increased by one each time as traversing from one vertex to another via that edge. For simplification, two anonymized trips with the same \( o \) and \( d \) are used to create a representative trip, as shown in Figure 3a,b.

Figure 3c is the directed graph with the weighted edges formulated using the input of Figure 3a Anonymized Trip 1 \((V_o, V_1, V_2, V_3, V_d)\) and Figure 3b Anonymized Trip 2 \((V_o, V_1, V_3, V_d)\). The weight calculation on each edge of Figure 3c is pretty straightforward. For example, the weight of the edge \( e_{o_1, o_2} \) is two, as it is clearly seen that Anonymized Trip 1 and Anonymized Trip 2 each traverse from the vertex \( V_o \) to \( V_1 \), respectively, as shown in Figure 3a,b. To create a representative trip, first, we start at the vertex \( V_o \), and then move to \( V_1 \), as shown in Figure 3c. At this vertex \( V_1 \), we move to next vertex \( V_3 \) instead of \( V_2 \) as the weight of \( e_{o_1,v_3} \) is higher than the weight of \( e_{o_1,v_2} \). If more than one edge has the same
weight as the next available vertices, we will randomly move to one of the vertices. Finally, we move from \( V_3 \) to \( V_d \) of the destination. Hence, all the visited vertices and edges are used to create the representative trip as shown in Figure 3d.

![Graph Image](image.png)

**Figure 3.** (a) Anonymized Trip 1, (b) Anonymized Trip 2, (c) Anonymized Trips 1 and 2, and (d) Representative Trip.

The details of the trip publication component are depicted in Algorithm 6. The input into this algorithm is the list of anonymized trips from the last part. As discussed before, the graph-based approach is to find representative trips, as shown in line 2 of Algorithm 6. At the end of this algorithm, the output is the list of representative trips. For example, Figure 4 shows a representative trip in red color, which is created from the list of anonymized trips in blue.

**Algorithm 6:** Trip Publication Component in PPTPF.

```
Input: \( T' \)—a list of AnonymizeTrips
Output: \( R \)—a list of Representative Trips

1 for \( i \leftarrow 1 \) to \( \text{size}(T') \) do
2     /* vertices are locations and a connection between two vertices is edge */
3     build graph \( G_i \) based on \( T'_i \)
4     /* traverse \( G_i \) by selecting a next vertex with higher weight */
5     \( R_i = R_i + (V_j, \text{No_User}), V_j \in \forall V \text{ of } G_i \)
6 return \( R \)
```

**Time Complexity:** Again, this component can run in parallel to create representative trips from the anonymized trips. The performance is based on available resources such as several processor cores and the available machine memory and disk space. With running on a single machine, the time complexity of this component is \( O(n^3) \).

**Security Complexity:** As discussed above, the anonymized trips can prevent user privacy leaks, i.e., without disclosing individual users with their trip patterns. As a result, the representative trips created from the list of the anonymized trips can preserve user privacy. Hence, representative trips without privacy violations can be released to the public for data analytics.
3.2. PPTPF Summary

Our proposed privacy-preserving trajectory publication framework (PPTPF) consists of five modular components. The details of the PPTPF are given in Algorithm 7. The published representative trips stored in the Hadoop distributed file system (HDFS) (HDFS is a distributed file system with highly fault-tolerant that can provide high throughput access to large datasets) are the final output of the PPTPF. The intermediate results of the PPTPF components are also in HDFS for security analysis.

Algorithm 7: Algorithm for a privacy-preserving trajectory publication framework.

Input: $NI$ is a number of iterations

Input: $NP$ is a number of partitions

1. $DL \leftarrow$ retrieve location data from HDFS
2. Convert $DL$ into a dataframe of type Location, $DL'$
3. $C_1 \leftarrow$ call Algorithm 1 ($DL$) with $NP$
4. $C_2^{p_1 \ldots q} \leftarrow$ call Algorithm 2 ($C_1^{p_1 \ldots q}$) with $NP$
5. $C_3^{p_1 \ldots q} \leftarrow$ call Algorithm 3 ($C_2^{p_1 \ldots q}$) with $NP$
6. $C_4^{j_1 \ldots k} \leftarrow$ call Algorithm 4 ($C_3^{j_1 \ldots k}$) with $NP$ and $NI$
7. $C_5^{j_1 \ldots k} \leftarrow$ call Algorithm 6 ($C_4^{j_1 \ldots k}$) with $NP$
8. Store $C_5$ in HDFS

Time Complexity: The performance of PPTPF on Spark is based on resources such as several processor cores, available machine memory, and disk space of the machines. Several bottlenecks [39] can be caused by the network, disk, and straggler tasks. These bottlenecks can affect the performance significantly. Many have proposed efficient and effective methods [39] to optimize algorithm-based Spark performance, especially in reducing job completion time as follows: (i) applying some network optimization techniques, (ii) reducing or eliminating disk accesses, and, lastly, (iii) detecting straggler tasks and then optimizing them. With running on a single machine, time complexity of the proposed framework is $O(n(1 + \log n) + 2n^2 + n^3)$.

Security Complexity: Output of PPTPF is a list of representative trips generated from the anonymized trips. As discussed before, both the representative and anonymized trips are $k$-anonymized privacy preserved. Furthermore, user count data (number of users) can also be released to the public. However, the count data can cause some privacy leaks. Another approach is to replace a real identity (userid) with pseudo-identity (anonymized userid) for the count data. This approach can yield more privacy leaks with a higher probability. For example, let a single anonymized userid appear in different stay regions of the trips. In each stay region of the anonymized trips, we can probably identify a user by...
combining several stay regions containing the user. In the worst case, a user only exists in the combination of the different stay regions—therefore, the identity replacement is not a suitable solution, and thus it is not used in PPTPF.

3.3. Evaluation Metrics for PPTPF

Performance of the proposed PPTPF is measured in terms of risk and utility of representative trips. We use the concepts of discernibility and distortion [25] to measure performance. Let \( P = \{ p_1, \ldots, p_n \} \) be a clustering of \( D \), where \( p_1, \ldots, p_{n-1} \) are clusters and \( p_n \) is a trash bin. Discernibility is defined as:

\[
DM(D) = \sum_{i=1}^{n-1} \| p_i \|^2 + \| p_n \| \| D ||, \tag{3}
\]

where \( \| D \| \) is the data size or the number of trips in PPTPF.

Another measure, information distortion (\( ID \)), is defined as:

\[
ID(D, D') = \sum_{t \in D} ID(t, t') \tag{4}
\]

\[
ID(t, t') = \begin{cases} 
    DTW \text{ dist}(t, t') & \text{if } t \text{ is in a cluster} \\
    \Omega & \text{otherwise}
\end{cases} \tag{5}
\]

where \( DTW \text{ dist}(t, t') \) is a distance between two trajectories (trips) of \( t \) and \( t' \). The distance is measured using DTW (Dynamic Time Warping) technique [40]. \( t' \) is the representative trip of a cluster. \( DTW \text{ dist} \) is applied only when the anonymized trip \( t \) is clustered with at least \( k-1 \) trips, and otherwise the distance is given a constant weight, which is typically high.

Discernibility measures risk, whereas distortion measures utility. Intuitively, discernibility decreases when more trips are clustered or \( k \)-anonymized, and, vice versa, i.e., the higher the number of non-clustered trips, the higher the discernibility. Distortion measures distance between each trip and its corresponding representative trip, i.e., low distortion means representative trip is not very different from the trips it represents, thus leading to high utility.

4. Performance Evaluation and Discussion

In this section, we evaluate the performance of our proposed PPTPF using two datasets. First, we discuss characteristics of the datasets, followed by implementation details and experimental set-up of PPTPF. Finally, we discuss experimental results and analyze the performance. Note that, among existing methods, there is no method that addresses the privacy issues of publishing large CDR-based trajectories. Thus, we do not compare our method with any existing methods.

Two datasets are used to evaluate the performance of our proposed PPTPF as follows.

- **Taxi dataset**: Singapore taxi dataset contains 59,183,257 records of 25,860 taxis moving for one day (23 April 2015) in Singapore. Each taxi driver has a smartphone to record his movement. Each record consists of a timestamp, taxi identifier, latitude and longitude information. To further test a larger dataset with approximately 85 GB, we use 420,744,849 records of 25,860 taxis moving for seven days (1–7 April 2015). The results of this dataset are discussed in Appendix A.

- **DataSpark Singapore dataset**: This dataset is from ‘IDA Personal Data Protection Challenge’ held in 2016. It contains 192,689 records of more than 900 people moving for 15 days (i.e., 1–15 September 2015). Each record consists of a timestamp, user identifier, latitude, and longitude information.

**Implementation Details**: The proposed PPTPF consists of five components that use Spark library. In other words, PPTPF can run on a machine installed with Spark. In this experiment, three machines installed with Spark 1.6 run the PPTPF, each containing 24 CPU cores. Each machine uses 64 GB for the experiments. All machines are used as
worker nodes that run the components of the PPTPF. One of these machines serves as a master node that helps to schedule and coordinate the machine resources and components.

**Experiment Settings:** In the experiments, the following settings are used. Values of $SPT$ and $TMT$ were set to $1$ h and $1$ km respectively as shown in Algorithm 2. The number of partitions ($NP$) and the number of iterations ($NI$) of the Algorithm 2 were set to $200$ and $3$. Gupta et al. [41] have suggested an optimal value of five cores per executor. This setting was also used in our experiment. After allocating one core for a Hadoop/Yarn daemon in each machine, we have a total of $69$ cores in the cluster ($3 \times (24 - 1) = 69$). It contains $13$ executors per core ($69/5 \approx 14$ − $1$, again, we need to allocate one executor for Spark Application Manager). Therefore, for each machine with $64$ GB, executor memory is set to $11$ GB ($64/5 - (0.07 \times 64/5) \approx 11$). The approximated value $0.90$ ($0.07 \times 64/5$) is a reserved space for heap overhead. Hence, based on the previous calculations, the number of executors, executor memory, and executor cores based on the available machine resources are $12$, $11$ GB, and $5$, respectively. Various $k$ values (5, 10, 15, and 20) applied on $k-1$ anonymized trips measure discernibility and distortion in PPTPF. All of the above settings are not optimized to give the best performance in this paper.

**Stay Region:** In the experiment, all stay regions of DataSpark dataset and Taxi dataset were extracted using Algorithm 2. For example, in the DataSpark dataset, the top five most frequented stay regions of a user are as shown in Figure 5. The darker color indicates higher frequency. The two most frequent stay regions are the home and office of the user.

![Figure 5. Top five stay regions of a person.](image)

**Behavior of Discernibility and Distortion:** Discernibility and distortion were calculated based on different $k$ values. Tables 1 and 2 show experimental results for the two datasets. Clearly, the number of valid clusters reduces as $k$ increases. Each valid cluster contains at least $k - 1$ anonymized trips. As $k$ increases, discernibility increases as well. It indicates that the number of non-clustered trips influences discernibility. The higher the discernibility, the higher the number of non-clustered trips. One of the main reasons is that forming a cluster to satisfy $k - 1$ trip conditions becomes more difficult as $k$ increases. As discussed before, a higher number of non-clustered trips can cause a higher risk. To reduce this risk, we can suppress the non-clustered trips. Therefore, in the experiment, the suppressed set (trips that can not be clustered) increases as $k$ increases. This result has been again proven based on Equation (3). Let trash bin $p_n$ be the suppressed set. On the right-hand side of Equation (3), $|p_n| \cdot |D|$ contributes significantly more than the other term, $\sum_{i=1}^{n-1} ||p_i^2||$. Thus, as the suppressed set size increases, so also does discernibility.
Table 1. Study of discernibility and distortion with a varying $k$ on the DataSpark Singapore dataset.

| DataSpark Singapore Dataset | $k = 5$ | $k = 10$ | $k = 15$ | $k = 20$ |
|----------------------------|---------|----------|----------|----------|
| No valid clusters          | 35,319  | 29,547   | 26,234   | 24,332   |
| Discernibility             | $7.008 \times 10^8$ | $9.860 \times 10^8$ | $1.1499 \times 10^9$ | $1.2300 \times 10^9$ |
| Largest cluster size       | 257     | 275      | 231      | 256      |
| Suppressed set size        | 14,214  | 19,896   | 23,209   | 25,111   |
| Distortion MM              | $9.7113 \times 10^4$ | $1.0370 \times 10^5$ | $1.0525 \times 10^5$ | $1.0623 \times 10^5$ |
| Distortion MIP             | $1.0362 \times 10^5$ | $1.1074 \times 10^5$ | $1.1305 \times 10^5$ | $1.1445 \times 10^5$ |

MM—Markov Model, MIP—Maximum Intermediate Points.

Next, we discuss distortion results in Tables 1 and 2. Clearly, as $k$ increases, distortion increases. The experimental results are consistent with the performance analysis of the proposed PPTPF, as discussed in the previous section. However, in the taxi dataset, distortion based on the Markov Model gets highest when $k = 10$, as shown in Table 2. This performance may result from the number of iterations ($NI$) and the number of partitions ($NP$) set in the experiment. For example, some specific trips that meet $k−1$ trip conditions stay in various partitioned sections. To overcome this issue, we can increase the number of iterations and reduce the number of partitions in the experiment. The distortion results indicate that, when representative trips are computed using the Markov model, the resultant method outperforms another method where representative trips are trips with a maximum number of intermediate locations.

Results of Privacy Preservation: Our proposed PPTPF created representative trips for DataSpark and taxi datasets. These trips are generated based on the $k−1$ trip anonymization approach. As a result, these trips do not disclose privacy information such as user identities and traversed locations. For example, Figure 6 pictorially shows one of the experiment results that contain various representative trips of 10 clusters based on the DataSpark dataset. Each representative trip has an origin and destination. The timestamp of the representative trip origin is the average of all origin timestamps of the related trips, including weekdays or weekends. Similarly, the timestamp of the destination of the representative trip is calculated. This way of representing the anonymized trips is much more useful than using count data [12]. As discussed before, our PPTPF can also provide user count data. Our proposed framework can withstand the adversarial attack as discussed in Appendix A. Above all, the proposed framework can be applied in a wide range of trajectory applications that need to preserve user privacy.
5. Conclusions and Future Work

We propose a Privacy-Preserving Trajectory Publication Framework (PPTPF) for moving object trajectories that can preserve user trajectory privacy while still maintaining the user mobility patterns. PPTPF uses the stay region and the trip concepts to ensure the privacy of trajectories while still retaining as much pattern information as possible. In addition, as PPTPF is based on Spark framework, it readily processes and anonymizes big trajectory data. PPTPF consists of five modular components that can be easily reused and extended without significant modifications. Furthermore, two measurements, discernibility and distortion, have been proposed and used to estimate risk and utility on the published trajectories, respectively. Experimental results have shown that PPTPF can provide good user privacy preservation on user trajectories while still maintaining good mobility patterns for data analytics. We will investigate to add different privacy-preserving techniques into the proposed framework.

Author Contributions: Conceptualization, Jianxi Yang and Sin G. Teo; methodology, Sin G. Teo and Manoranjan Dash; software, Sin G. Teo; validation, Jianxi Yang, Manoranjan Dash and Sin G. Teo; data curation, Sin G. Teo; writing—original draft preparation, Jianxi Yang, Manoranjan Dash and Sin G. Teo; writing—review and editing, Jianxi Yang, Manoranjan Dash and Sin G. Teo; funding acquisition, Jianxi Yang. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: The study was conducted according to the guidelines of the Declaration of Helsinki, and approved by the Institutional Review Board (or Ethics Committee) of Institute for Infocomm Research.

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: Data sharing not applicable.

Acknowledgments: We thank Vincent C.S. Lee (Monash University Australia) and Senior Scientist Wee Siong Ng (A*STAR, Singapore) for their useful comments and suggestions.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Additional Experiment and Adversarial Model

Appendix A.1. Large Taxi Dataset on the Proposed Framework (PPTPF)

We further evaluate PPTPF on a larger dataset. Details of the dataset and performance results are discussed below.
• **Taxi dataset:** Singapore taxi dataset contains 420,744,849 records of 25,860 taxis moving for 7 days (1–7 April 2015) in Singapore. The dataset size is approximately 85 GB. Each taxi driver has a mobile phone to record his movement in this experiment. Each record consists of a timestamp, taxi identifier, latitude, and longitude information. Based on $k - 1$ trip anonymization, we varied $k$ as 5, 10, 15, and 20. Discernibility and distortion were calculated based on different $k$ values. Table A1 shows that number of valid clusters reduces as $k$ increases.

| Taxi Dataset | $k = 5$ | $k = 10$ | $k = 15$ | $k = 20$ |
|--------------|---------|----------|----------|----------|
| No valid clusters | 369,761 | 349,844 | 341,391 | 339,192 |
| Discernibility | $1.7880 \times 10^{12}$ | $1.8186 \times 10^{12}$ | $1.8315 \times 10^{12}$ | $1.8349 \times 10^{12}$ |
| Largest cluster size | 4905 | 4158 | 3935 | 3641 |
| Suppressed set size | 1,164,843 | 1,184,760 | 1,193,213 | 1,195,412 |
| Distortion MM | $1.1137 \times 10^7$ | $1.1047 \times 10^7$ | $1.1680 \times 10^7$ | $1.2050 \times 10^7$ |
| Distortion MIP | $7.0884 \times 10^7$ | $7.0904 \times 10^7$ | $6.8385 \times 10^7$ | $6.8566 \times 10^7$ |

MM –Markov Model, MIP –Maximum Intermediate Points.

**Appendix A.2. Adversarial Knowledge**

The proposed framework, PPTPF, uses a $k$-anonymity location obfuscation technique to generate trajectories while preserving user privacy before releasing them to the public. Therefore, the aim of the $k$-anonymity technique in PPTPF is to preserve location privacy. In other words, the adversary cannot learn the location of a user at a given time. Let an adversary have access to statistical information about the user mobility patterns. For example, the adversary could learn some user workplace and home via publicly available information. Hence, the adversary may know the user who will be in the office during office hours and home during the night, with high confidence. This knowledge may expose user trajectory patterns to the adversary. To prevent this attack, the framework, PPTPF, constructs $k - 1$ trips based on the Markov Model with a sufficiently large trajectory dataset. Furthermore, PPTPF uses the concept of the stay region for trip anonymization. This makes the inference attack on the proposed framework harder here.

**References**

1. How Many Mobile Phones Are in the World? 2021. Available online: https://www.bankmymobile.com/blog/how-many-phones-are-in-the-world (accessed on 25 January 2021).
2. Blondel, V.D.; Decuyper, A.; Krings, G. A survey of results on mobile phone datasets analysis. *EPJ Data Sci.* 2015, 4, 10. [CrossRef]
3. Naboulsi, D.; Fiore, M.; Ribot, S.; Stanica, R. Large-scale mobile traffic analysis: A survey. *IEEE Commun. Surv. Tutor.* 2015, 18, 124–161. [CrossRef]
4. Saramäki, J.; Moro, E. From seconds to months: An overview of multi-scale dynamics of mobile telephone calls. *Eur. Phys. J. B* 2015, 88, 1–10. [CrossRef]
5. James, J. The smart feature phone revolution in developing countries: Bringing the internet to the bottom of the pyramid. In *The Impact of Smart Feature Phones on Development*; Springer: Cham, Switzerland, 2020; pp. 226–235.
6. Number of Smartphone in 2021. Available online: https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide (accessed on 25 January 2021).
7. Kujala, R.; Aledavood, T.; Saramäki, J. Estimation and monitoring of city-to-city travel times using call detail records. *EPJ Data Sci.* 2016, 5, 1–16. [CrossRef]
8. Poushter, J.; Oates, R. *Cell Phones in Africa: Communication Lifeline*; Pew Research Center: Washington, DC, USA, 2015.
9. Summary of the HIPAA Privacy Rule. Available online: https://www.hhs.gov/hipaa/for-professionals/privacy/laws-regulations/index.html (accessed on 25 January 2021).
10. GDPR Privacy Policy Template. Available online: https://www.privacypolicies.com/blog/gdpr-privacy-policy/ (accessed on 25 January 2021).
11. de Montjoye, Y.A.; Hidalgo, C.A.; Verleysen, M.; Blondel, V.D. Unique in the crowd: The privacy bounds of human mobility. *Sci. Rep. Nat.* 2013, 3, 1–5. [CrossRef] [PubMed]

12. Acs, G.; Castelluccia, C. A case study: Privacy preserving release of spatio-temporal density is Paris. In Proceedings of the 20th SIGKDD Conference, New York, NY, USA, 24–27 August 2014.

13. Dash, M.; Koo, K.K.; Gomes, J.B.; Krishnaswamy, S.P.; Rugeles, D.; Shi-Nash, A. Next, Place Prediction by Understanding Mobility Patterns. In Proceedings of the 14th PerCom (PerMoby Workshop), St. Louis, MO, USA, 23–27 March 2015.

14. Kellaris, G.; Papadopoulos, S. Practical differential privacy via grouping and smoothing. In Proceedings of the VLDB Endowment, Riva del Garda, Italy, 30 August 2013; pp. 301–312.

15. Gagniuc, P.A. *Markov Chains: From Theory to Implementation and Experimentation*; John Wiley & Sons: Hoboken, NJ, USA, 2017.

16. Mathew, W.; Raposo, R.; Martins, B. Predicting future locations with hidden Markov models. In *Proceedings of the ACM Conference on Ubiquitous Computing*, Pittsburgh, PA, USA, 5–8 September 2012; pp. 3:1–3:6.

17. Kido, H.; Yanagisawa, Y.; Satoh, T. An anonymous communication technique using dummies for location based services. In Proceedings of the 13th International Conference on Extending Database Technology, Lausanne, Switzerland, 22–26 March 2010; pp. 135–146.

18. Li, N.; Li, T.; Venkatasubramian, R. Mondrian multidimensional k-anonymity. In Proceedings of the ICDE 2006, Atlanta, GA, USA, 3–7 April 2006; p. 25.

19. Kellaris, G.; Papadopoulos, S. A Privacy-Preserving Trajectory Publication Method Based on Secure Start-Points and End-Points. In Proceedings of the ACM Conference on Management of Data, Vancouver, BC, Canada, 9–12 June 2008.

20. Hong, J.I.; Landay, J.A. An architecture for privacy-sensitive ubiquitous computing. In Proceedings of the International Conference on Mobile Systems, Applications, and Services, Boston, MA, USA, 6–9 June 2004.

21. Kellaris, G.; Shahabi, C. Blind evaluation of nearest neighbor queries using space transformation to preserve location privacy. In Proceedings of the ACM Conference on Management of Data, Vancouver, BC, Canada, 9–12 June 2008.

22. Alam, C.N.; Manaf, K.; Attmadja, A.R.; Aurum, D.K. Implementation of haversine formula for counting event visitor in the radius based on Android application. In Proceedings of the International Conference on Cyber and IT Service Management, Bandung, Indonesia, 26–27 June 2016; pp. 1–6.

23. Mathew, W.; Raposo, R.; Martins, B. Predicting future locations with hidden Markov models. In Proceedings of the ACM Conference on Ubiquitous Computing, Pittsburgh, PA, USA, 5–8 September 2012; pp. 911–918.

24. Gagniuc, P.A. *Markov Chains: From Theory to Implementation and Experimentation*; John Wiley & Sons: Hoboken, NJ, USA, 2017.

25. Ousterhout, K.; Rasti, R.; Ratnasamy, S.; Shenker, S.; Chun, B.G. Making sense of performance in data analytics frameworks. In Proceedings of the 12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15), Oakland, CA, USA, 4–6 May 2015; pp. 293–307.
Short Biography of Authors

Jianxi Yang received the PhD degree in Bridge Engineering from Chongqing Jiaotong University, China, in 2011, and the BS and MS degrees from Chongqing Jiaotong University, China, in 2000 and 2007, respectively. Currently, he is a professor of Chongqing Jiaotong University. His current research interests include big data analysis of bridge structure monitoring.

Manoranjan Dash is a senior data scientist in the National University of Singapore. Prior to this, he worked for A*STAR and Nanyang Technological University of Singapore. He has published more than 60 technical papers in reputable conferences and journals. His main research area includes different aspects of machine learning such as clustering, feature selection, sampling, etc.

Sin G. Teo is a research scientist at the Institute for Infocomm Research (I²R) in Singapore. He obtained the PhD degree from Monash University, Australia in 2016. His research interests include applied cryptography, data privacy and security, malware and network anomaly classification, data mining, and deep learning.