GeoSecure-R: Secure Computation of Geographical Distance using Region-anonymized GPS Data

Vikram Patil and Pradeep K. Atrey
Albany Lab for Privacy and Security, College of Engineering and Applied Sciences
University at Albany, State University of New York
Albany, NY, USA
Email: {vpatil, patrey}@albany.edu

***Abstract*—Today Location-Based Services (LBS) are used by millions of users all over the world. These services widely use multimedia sensory data such as GPS location. LBS provides enormous convenience to the users, but at the expense of continuously tracking their location. This raises severe privacy concerns. The distance computation using users' GPS coordinates is a crucial component in many LBS, such as the fitness tracker app and understanding driving habits. In this paper, we propose a method, called GeoSecure-R, to calculate the distance without revealing the users' exact location. The proposed method ensures users' privacy with region anonymity while maintaining utility, i.e., it provides LBS that requires only the traveled distance rather than the actual location of the user. Experimental results on Microsoft's GeoLife dataset show that the proposed method calculates the distance metric very close to what we get using the actual location of the user.**

***Index Terms*—Trajectory compression, Trajectory privacy, Haversine, Trajectory security, Location based services (LBS), Delta compression, GPS sensor data.**

I. Introduction

Location-Based Services (LBS) primarily use multimedia sensory data, such as GPS location information. The LBS providers often launch new services based on GPS data without adequate privacy and security (confidentiality) considerations which results in loss of users' trust. GPS data can be misused in several ways. GPS data can easily reveal the home location, work location, places visited, routes taken by the user, etc. Further, sensitive information like political, religious affiliations, healthcare providers, banks, shopping patterns, etc. can also be easily derived from the GPS trajectory. In more severe cases such as continuously tracking a user, it can lead to dangerous situations like illegal surveillance, stalking, or other criminal activity. Often, users have to choose between allowing tracking of their location and facing inconvenience by not using the LBS. Therefore, while LBS enabled apps and devices provide convenience in our everyday life, they come at the expense of users' privacy. LBS providers often use cloud service providers (CSP) to host their applications. There is often a risk of a data breach at CSP. Further, CSP or LBS providers can itself act as a semi-malicious adversary, meaning that it provides the intended LBS but can use or misuse the data for other purposes as well. Hence, protecting GPS data is an immediate need.

When the users' GPS trajectory data is stored on the CSPs, the following four factors are considered: (1) *security* or confidentiality of the data (2) users' *privacy* (3) storage cost (*compression*) (4) *utility* of the data. Users are usually concerned about the security and privacy, but they still want to use the LBS, whereas the primary goal of the CSP or LBS providers is to reduce the storage cost and provide utility, though the security and privacy could also be their secondary goals. Users can achieve the goal of security and privacy by employing encryption before sending data to the CSP or LBS providers; however, in this case, the utility is severely affected. The encrypted data is typically useless for CSP or LBS providers, barring a set of limited operations accomplished by using homomorphic schemes. Further, if the users' actual (unencrypted) data is available to the CSP or LBS providers, they not only can compress it to reduce the storage space requirement but can also use it to any extent. CSP or LBS providers may also encrypt the data to avoid access to outside attackers; however, it doesn't restrict them to decrypt and access the data. Given that CSP or LBS providers could be semi-malicious adversary, this option does not guarantee security and privacy. In summary, it is hard to come up with a single approach that can ensure security, privacy, compression, and utility all together.

There are many LBS that require only distance between two GPS coordinates, for instance, travel model detection, Fitbit tracker, etc. We envision that such LBS can be performed without knowing the actual GPS coordinates of the users. Our previous work [1] leverages on this phenomenon, where we introduced a method called, GeoSecure, to compute the distance between two GPS coordinates without revealing users' actual location. GeoSecure method provides reasonable accuracy in many cases; however, its limitation lies in assigning a fixed value of 1 to the cosine product term (referred as β) in the haversine formula (Eq. 3). In some instances where the value of β is very different from 1, the distance computation error can exceed the acceptable threshold. In another work [2], we presented GeoSecure-O method which used an optimization technique, to find the optimal value of β in the haversine formula dynamically. However, we need to perform few operations on the client-side, and although the error in the distance calculation is significantly lower than GeoSecure, we need to reduce it further.

In this paper, we present a novel approach called GeoSecure-R. The GeoSecure-R method is an extension of our previously published works GeoSecure [1] and GeoSecure-O [2], and it significantly reduces the error in distance calculation and improves the accuracy as compared to our previous works. The core idea behind the GeoSecure-R approach is to exploit the fact that any pair of GPS points will have almost the same haversine distance if the difference between their corresponding latitude and longitude is the same, provided both pair of points are located in the same geographical region such as a city. In this method, only the geographical region of the user is known to the LBS provider; the exact GPS location is not known though. The LBS provider uses the GPS coordinate of a known point in the region along with the differences in latitude and longitude of the consecutive GPS points from the users' trajectory to calculate the distance traveled by the user.

The main contribution in this paper is a novel approach to calculate distance without revealing users' exact location, which has the following features:

1) The privacy of the users' location is ensured.
2) The error between the actual distance and the distance calculated using the proposed method is very small (high accuracy).
3) The accuracy of the derived quantities such as velocity and acceleration is also consistent with the distance accuracy; hence, it can be used for more applications.

Rest of the paper is organized as follows, Section II presents related work; the proposed work is explained in Section III; experimental results are presented in Section IV; and conclusion is provided in Section V.

II. Related Work

Several approaches are proposed in the literature to solve problems of GPS trajectory data compression, privacy, and security [3]. In this section, we will provide a brief overview of related work.

A. Existing Work on GPS Data Privacy

Generally, privacy of GPS data can be viewed in two ways: identity privacy and location privacy. In identity privacy, the focus is on delinking identity of the users from their GPS trajectory data; while in location privacy, the goal is to obscure the GPS location of users even though their identity is known.

There are a number of works that have focused on identity privacy in GPS data domain [9], [10], [11], [12], [6]. In [13], Sweeney introduced k-anonymity, which is a widely popular approach that ensures privacy in data sets by decoupling unique identifiers from a specific user. In anonymity approaches, often a Trusted Third Party (TTP) server is utilized for data anonymization between the user and the service provider. However, privacy attacks, such as linkage attacks can still de-anonymize the data. Additionally, l-diversity [14] and t-closeness [15] address limitations of k-anonymity and are also very widely used. In [16], Abul et al. proposed another improvement to k-anonymity called as (k, δ)-anonymity. In [4], Gao et al. demonstrated the application of k-anonymity to transform a GPS trajectory dataset unidentifiable. In [17], Xiong et al. elaborated more limitations of k-anonymity approach. The anonymity based approaches in which a TTP server is used, the TTP can also be a semi-malicious or curious adversary, and the risk of data theft from the TTP server is also present. In some approaches, the area is divided into cloaking regions, and the user is considered to be anonymous within that region.

Differential privacy is another popular approach to anonymize the data. Differential privacy based techniques add noise to the location using Laplace distribution; hence the exact location is not revealed. Although this is a popular method for anonymization, it is designed for aggregate data queries. In [18], Andres et al. explained how differential privacy can be used in the context of LBS.

To protect location privacy, the approaches include obfuscation and masking techniques such as displacement masks, affine transforms and donut masking [6], [17], [19]. These techniques are used extensively for hiding user location in the healthcare/disease datasets to reduce identification risk. Health or disease datasets often have patient's location as one of the parameters along with other data. Hence, the masking techniques add random noise or shift the location of each of these records to reduce identification risk. In [8], Armstrong et al. discussed various masking techniques for location data in the context of protecting the user' location. They also mention that the translation and rotation masks preserve the actual distance between the pair of GPS points. Also, in [20], Grushter proposed a technique to provide privacy using co-ordinate transformation, which uses translation and rotation operations. Subsequently, in [21], Di Pietro et al. used obfuscation function and Merkel trees to provide LBS without tracking the location of the user. Also, Although these obfuscation and masking techniques provide privacy, the major difference between them and the proposed method is that the proposed method compresses the data using delta compression, hence the resultant size of the data is reduced. On the other hand, the obfuscation and masking techniques do not compress the data. Also, in the proposed method, we do not have to define a specific function for masking or transformation.

B. Existing Work on GPS Data Compression

The LBS providers can save huge money on storage and data transfer cost if the data is compressed. Lossy [22], [23] and lossless [24], [25] are two approaches to compress GPS data.

In Delta encoding, we consider differences between successive points to represent the series of data, and this approach has been extensively used for the GSP data compression. In [26], Cudre-Mauroux et al. used delta encoding in the context of storing, trajectory comparison, and querying GPS trajectories. They also used fixed-point arithmetic, in which they multiply each latitude and longitude value with a fixed number so that the resultant difference is in the natural number. Although their approach results in a good compression ratio,

TABLE I: A comparison of proposed work with the existing techniques

Work	Approach	Privacy?	Security*?	Compression?	Utility for aggregate data?	Utility for individual user?
Gao et al. [4]	*k*-anonymity	Yes	No	No	Yes	No
Marias et al. [5]	Secret Sharing Based	Yes	No	No	Yes	Yes
Seidl et al. [6]	Obfuscation and masking	Yes	No	No	Yes (limited)	Yes (limited)
Šedenka and Gasti [7]	Homomorphic encryption	Yes	Yes	No	Yes	Yes
Armstrong et al. [8]	Masking/ Obfuscation	Yes	Yes	No	Yes	Yes
Patil et al. [1]	GeoSecure	Yes	Yes	Yes	Yes	Yes
Patil et al. [2]	GeoSecure-O	Yes	Yes	Yes	Yes	Yes
Proposed	**GeoSecure-R**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**

* Security refers to protection of data from semi-malicious CSP/TTP or LBS providers.

they do not focus on the security, privacy, or utility of the GPS data. The proposed method also uses delta compression, fixed point arithmetic, as mentioned in [26], but since we do not store the first point, it not only provides security and privacy but also maintains the utility of the resulting data. In [27], Nibali and He proposed an enhancement to the [26] by adding predictor function to the delta compression, resulting in better compression ratio.

C. Existing Work on GPS Data Security

To maintain the confidentiality of the data, various approaches such as encryption are proposed. However, to perform the operations on encrypted data, we must first decrypt it. The only exception to this is homomorphic encryption, in which operations can be performed directly on the encrypted data. For instance, in [28], Liu et al. used the Paillier cryptosystem to find similarities in encrypted trajectory data. Although homomorphic encryption provides a way of working directly on encrypted data, it only supports limited operations. In [5], Marias et al. introduced a secret sharing based approach to provide privacy in LBS. Their approach is based on dividing the location and user identifier into multiple shares using secret sharing algorithms and only the intended recipients will be able to combine the shares to know the information. Although this approach can provide security against the external adversary, the semi-malicious/curious LBS provider can still access it. In [29], Hu and Zhu described a novel way to for proximity testing using homomorphic encryption and spatial cloaking. In [30], Li and Dai explored a secure two-party protocol for calculating Euclidean distance between two points and other geometric computations.

In [31] and [32], several techniques related to the LBS security and privacy in Wireless Sensor Networks (WSN) domain have been reviewed at length. In [33], Gruteser and Grunwald proposed an anonymizer framework, which relies on the trusted central server for anonymization. In [34], Ghinita et al. proposed encryption based method to securely provide LBS without revealing the user' location for the nearest neighbor (NN) queries.

In [35], Narayanan et al. proposed three protocols for proximity detection using cryptographic techniques. In [7], Šedenka and Gasti introduced a novel method for securely computing haversine and Euclidean distance between two GPS coordinates using homomorphic encryption. The main difference between their work and the proposed method is that they do not focus on compression, and because of encryption, the computational overhead is high. In [36], Hallgren et al. introduced a method for privacy ensured ride sharing based on secured multi-party computation and Shamir's secret sharing. The secured distance calculation methods are also used in determining trajectory similarity and proximity testing. In [37], Pesara et al. have reviewed these techniques. Data cleaning is also an essential part of LBS. In our previous work, GeoSClean [38], we explored secured trajectory data cleaning based on the GeoSecure approach.

Table I summarizes the comparison of the existing work with the proposed method. Note that although the GeoSecure method [1] and GeoSecure-O [2] are designed to fulfill all four criteria (i.e., privacy, confidentiality, compression, and utility), the proposed method, GeoSecure-R, provides the improved accuracy and utility.

III. Proposed Work

A. Overview

The proposed method is illustrated in Fig.1. In the proposed method, we define a specific geographical region R where the user is present. This region can be a metropolitan area such as New York City or a smaller state. Similar to GeoSecure [1], we store the first point of the trajectory at the users' device and calculate the difference between consecutive points, which is sent to the CSP or LBS provider.

The LBS provider prefers to perform as many operations as possible on the cloud since the processing power of the users' device, storage is limited, and the battery will be drained faster. Further, it is less expensive to perform these operations, such as distance calculation in the cloud. Since distance calculation can be used for several applications that are also hosted on the cloud, performing this operation at the cloud makes it more efficient.

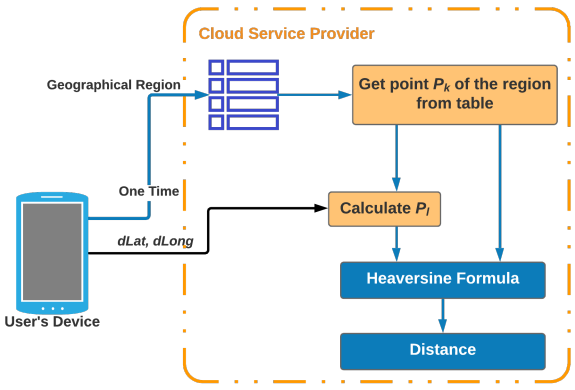


Fig. 1: GeoSecure-R workflow

B. Definitions

We present the definitions of Trajectory Point, Trajectory Privacy and Trajectory Utility as follows,

Definition 1. *(Trajectory Point) A trajectory point P_i is characterized by the following three attributes: lat_i (latitude), $long_i$ (longitude), and $time_i$ (timestamp).*

Definition 2. *(Trajectory Privacy) A trajectory is said to possess k-anonymity privacy if an attacker/adversary is unable to find the exact location of the user within k different points. Assuming that the privacy loss, $\phi_{T,S}$, with the original trajectory T for a LBS S, is 1 (i.e. full privacy loss), with the transformed trajectory T', it will be $\phi_{T',S} = \frac{1}{k}$ using k-anonymity phenomenon. Therefore, the Trajectory Privacy $\mathcal{P}_{T',S} \in [0,1]$ with the transformed trajectory T' for LBS S is given as:*

$$\mathcal{P}_{T',S} = 1 - \frac{\phi_{T',S}}{\phi_{T,S}} \tag{1}$$

Definition 3. *(Trajectory Utility) Let $O_{T,S}$ and $O'_{T',S}$ be the outcome of a given LBS S using the original trajectory T and the transformed trajectory T'. The Trajectory Utility $\mathcal{U}_{T',S}$ of the transformed trajectory T' for a given LBS S is calculated as:*

$$\mathcal{U}_{T',S} = 1 - \frac{|O_{T,S} - O'_{T',S}|}{O_{T',S}} \tag{2}$$

C. System and Threat Model

We consider the user as honest, and the users' device is assumed to be secure. The CSP and the LBS provider are considered as semi-malicious/curious, i.e., they will provide the service but might be interested in knowing the users' data. Any other party is considered an adversary.

Generally, users trust certain LBS providers such as navigation or ride-sharing services, etc. However, many times, users prefer to hide the location from other LBS providers such as fitness trackers, which it may not trust. The proposed method is focused on protecting users' location from the LBS providers which it does not trust.

In this threat model, we make the following assumptions: (A1) The connection between the users' device and the CSP or LBS provider is considered to be secure. (A2) The users do not reveal their location voluntarily via other sources such as social media. (A3) The LBS providers do not collude with the other LBS providers which has access to the users' location.

D. Haversine Distance

There are several approaches to calculate the distance between two points on the surface of the earth, such as Euclidean distance, haversine distance, etc. Euclidean distance is calculated by considering the earth's surface as flat and 2D Euclidean space. This is a simple way of calculating distances but introduces large errors. Haversine distance considers the earth as a sphere, and it calculates distance on any two points on the surface of the earth. The haversine distance is intended to be used with latitude and longitude as compared to the Euclidean distance designed for generic Cartesian coordinates. Distances calculated using the haversine formula are pretty accurate and widely used in many applications.

The haversine distance has been used in navigation, astronomy, and many other applications [39]. Haversine formula [40], [1] to calculate distance can be described as follows.

For any two given points P_i and P_j, the distance between them can be calculated as follows

$$dlong_{i,j} = long_j - long_i$$

$$dlat_{i,j} = lat_j - lat_i$$

$$a = (\sin(\frac{dlat_{i,j}}{2}))^2 + \cos(lat_i) \times \cos(lat_j) \times (\sin(\frac{dlong_{i,j}}{2}))^2$$

$$c = 2 \times atan2(\sqrt{a}, \sqrt{1-a})$$

$$d_{i,j} = R \times c \tag{3}$$

where R is the radius of the earth (mean radius = 6,371 km), $d_{i,j}$ is the distance between P_i and P_j, and $long_i$, $long_j$, lat_i, lat_i are longitude and latitude of point P_i and P_j respectively.

E. Approximated Haversine Formula

In the GeoSecure method, the LBS provider only has differences between consecutive points and do not have actual latitudes of the points. Hence, to calculate the distance using the haversine formula, the LBS provider will have to approximate the term $cos(lat_i) \times cos(lat_j)$. One option is to approximate this term to a constant value and calculate the distance. In our previous work (GeoSecure [1]), we approximated it to 1. The value of this constant can be chosen based on the geographical area or other considerations for the best results. Although it introduces error in the approximated distance and actual distance, the resulting error is limited.

Considering the value of the term $\beta = cos(lat_i) \times cos(lat_j)$ as 1, we get the following approximation in Eq.3,

$$a' = (\sin(\frac{dlat_{i,j}}{2}))^2 + \beta \times (\sin(\frac{dlong_{i,j}}{2}))^2$$

$$c' = 2 \times atan2(\sqrt{a'}, \sqrt{1-a'})$$
$$d'_{i,j} = R \times c' \quad (4)$$

F. Improvement over GeoSecure [1] and GeoSecure-O [2]

In Geosecure [1], we described the method to send GPS trajectory data in the form of differences $(dlat_{i,i+1}, dlong_{i,i+1})$ in successive points to the LBS provider and maintaining the first point in the trajectory at the users' device. The users' device saves the first point of trajectory T as key K and computes differences between successive points. These differences are then sent to the LBS provider which uses approximated haversine formula (Eq. 4), to calculate the distance between two successive points. In this way, we can maintain security, privacy, and compress the data. In many applications, this method can be used efficiently. However, this results in error $\epsilon = |d_{i,j} - d'_{i,j}|$. Some applications or LBS need even higher accuracy. Hence, we proposed an improved version called GeoSecure-O [2].

In our previous work, GeoSecure-O [2], we used optimization technique to dynamically find the value of the cosine product term β by minimizing the mean square error (MSE) between the actual and calculated haversine distance. The GeoSecure-O method involves calculating the actual distance between the first two points on the users' device and sending the actual distance, $(dlat_{1,2}, dlong_{1,2})$ to the LBS provider. The LBS provider then iteratively calculates MSE for all values of β from 0.001 to 1.0. The value of β, which provides the least MSE is called as β^o and used for distance calculation for the rest of the trajectory. The difference between GeoSecure-O and the proposed GeoSecure-R method is that we do not have to perform any calculation on the client-side, and there is no costly optimization involved at the server-side. Also, the proposed method provides higher distance calculation accuracy as compared to GeoSecure-O. The only limitation of the proposed method over the GeoSecure-O is that we have to provide the geographical region to the LBS provider which is not required in the GeoSecure-O method. However, the accuracy is higher for the proposed method.

G. GeoSecure-R Method

In the proposed method, we declare the region (e.g., city) in which the user is present and take its well-known location, e.g., a center point and refer it as the origin (P_k). Then we only take differences in latitude and longitudes between consecutive points and send it to the server. The server calculates the corresponding point (P_l) by adding these differences to the origin point (P_k) calculated earlier. Now the distance between the consecutive points sent by the user and the origin and corresponding point at the server will be very close since all of them belong to the same region.

In the following, first we describe the proposition on which the method is based, and subsequently we describe the steps in the proposed method.

Proposition 1. *The haversine distance between any two points P_i, P_j separated by $dlat_{i,j}$ and $dlong_{i,j}$ is nearly equal to the distance between any two points P_m, P_n which are also separated by the same $dlat_{i,j}$ and $dlong_{i,j}$ (such that $dlat_{i,j} = dlat_{m,n}$ and $dlong_{i,j} = dlong_{m,n}$), provided they belong to same geographical region R.*

Proof. Consider a geographical region R (lat_E, lat_W, $long_N$, $long_S$), where lat_E, lat_W, $long_N$, $long_S$ represent the eastern and western latitudes, northern and southern longitudes respectively. The error in the distance between P_i, P_j and P_m, P_n can be represented as follows:

$$\epsilon = d_{i,j} - d_{m,n}$$

$$\epsilon = [(\sin(\frac{dlat_{i,j}}{2}))^2 + \cos(lat_i) \times \cos(lat_j) \times (\sin(\frac{dlong_{i,j}}{2}))^2] - [(\sin(\frac{dlat_{m,n}}{2}))^2 + \cos(lat_m) \times \cos(lat_n) \times (\sin(\frac{dlong_{m,n}}{2}))^2]$$

$$\epsilon = (\cos(lat_i) \times \cos(lat_j)) - (\cos(lat_m) \times \cos(lat_n))$$

If $(lat_i, lat_j) \sim (lat_m, lat_n)$ i.e. are very close to each other (same geographical region) then their latitudes will be closer to each other i.e. few hundred meter change in actual location is also a small change in terms of latitude values.

The cosine value of the latitudes will be much closer. Since the product of cosines of these points will further reduce their difference, the error between them will be close to 0.

Hence, as long as the latitudes of the points are located in the same geographical region and not substantially distant like a few hundred miles, the resultant error will be minimal. ■

We consider a trajectory T consisting of point $P_1, \ldots, P_n$, where n is the length of the trajectory. The application on the users' device is expected to declare a geographical region R where the user will be using the service. This R will be passed to the server in the first step, and after that, only the differential data without the first point is sent to the LBS provider e.g., if P_i and P_{i+1} are two points of a trajectory in the region R then the users' device sends

$$dlat_{i,i+1} = lat_{i+1} - lat_i$$
$$dlong_{i,i+1} = long_{i+1} - long_i$$

The LBS provider then retrieves a point P_k which lies in the region R by querying a lookup table. Using the point P_k and the difference between the given two points $dlat_{i,i+1}$ and $dlong_{i,i+1}$, the server calculates the point P_l such that

$$lat_l = lat_k + dlat_{i,i+1}$$
$$long_l = long_k + dlong_{i,i+1}$$

Then the LBS provider calculates the distance between P_k and P_l which is close to the distance between P_i and P_{i+1} using the actual haversine formula.

The following example explains the steps described above: The steps for the proposed algorithm are divided into two parts region declaration and data processing. Steps 2 to 7 are common for GeoSecure[1] and GeoSecure-R.

Step 1 Declare a region R e.g. Beijing. (which is used in the GeoLife dataset) and send it to the LBS provider.

Step 2 The LBS provider then calculates a point $P_k \in R$ from the lookup table. In this case, we just took the point from Wikipedia as the center of the Beijing city. Hence,

$$(lat_k, long_k) = (39.913818, 116.363625)$$

Step 3 Let's assume trajectory T at the users' device as

$$\begin{pmatrix} 26.898747 & 112.591398 & 2009-10-02 & 06:02:28 \\ 26.898631 & 112.591448 & 2009-10-02 & 06:02:30 \\ 26.89874 & 112.591413 & 2009-10-02 & 06:02:31 \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix}$$

Step 4 Multiply latitude and longitude of all the points by 10^6 as per fixed point arithmetic as mentioned in [26], so that we get integer numbers and their difference will be in integer numbers.

$$\begin{pmatrix} 26898747 & 112591398 & 2009-10-02 & 06:02:28 \\ 26898631 & 112591448 & 2009-10-02 & 06:02:30 \\ 26898740 & 112591413 & 2009-10-02 & 06:02:31 \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix}$$

Step 5 Find the difference between consecutive points of the trajectory by subtracting every point from its predecessor point, starting from the second point.

$$\begin{pmatrix} 26898747 & 112591398 & 2009-10-02 & 06:02:28 \\ -116 & 50 & 0 & 2 \\ 109 & -35 & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix}$$

Step 6 Name the first point of the trajectory as key K and the differences as D.

$$K = \begin{pmatrix} 26898747 & 112591398 & 2009-10-02 & 06:02:28 \end{pmatrix}$$

$$D = \begin{pmatrix} -116 & 50 & 0 & 2 \\ 109 & -35 & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix}$$

Step 7 Keep the first point key K on the users' device and send the difference to the LBS provider.

$$D = \begin{pmatrix} -116 & 50 & 0 & 2 \end{pmatrix}$$

The key K can be used to transform the differences back into original trajectory.

Step 8 LBS provider calculates the corresponding point P_l by adding these differences in latitude and longitude to the point P_k calculated in the second step. Hence,

$$\begin{aligned} P_l &= P_k + (dlat_{i,i+1}, dlong_{i,i+1}) \\ P_l &= (lat_k, long_k) + (dlat_{i,i+1}, dlong_{i,i+1}) \\ (lat_l, long_l) &= (39.913818, 116.363625) \\ &\quad + (dlat_{i,i+1}, dlong_{i,i+1}) \end{aligned}$$

The $(dlat_{i,i+1}, dlong_{i,i+1})$ is (-115, 50), so after dividing by 10^6 the equation becomes

$$\begin{aligned} (lat_l, long_l) &= (lat_k, long_k) + (-0.000116, 0.000050) \\ &= (39.913818, 116.363625) \\ &\quad + (-0.000116, 0.000050) \\ &= (39.913702, 116.363675) \end{aligned}$$

Hence, we get the corresponding point P_l with

$$(lat_l, long_l) = (39.913702, 116.363675)$$

Step 9 The LBS provider then calculates the distance between P_k and P_l using the actual haversine formula, which will be very close to the given points in trajectory T.

These steps will be repeated for the rest of the points of the trajectory T. The LBS provider can also keep track of erroneous points such as same timestamp for multiple successive points, and also stop points (where the latitude and longitude is the same for a few consecutive points) and ignore them accordingly.

IV. Experimental Results

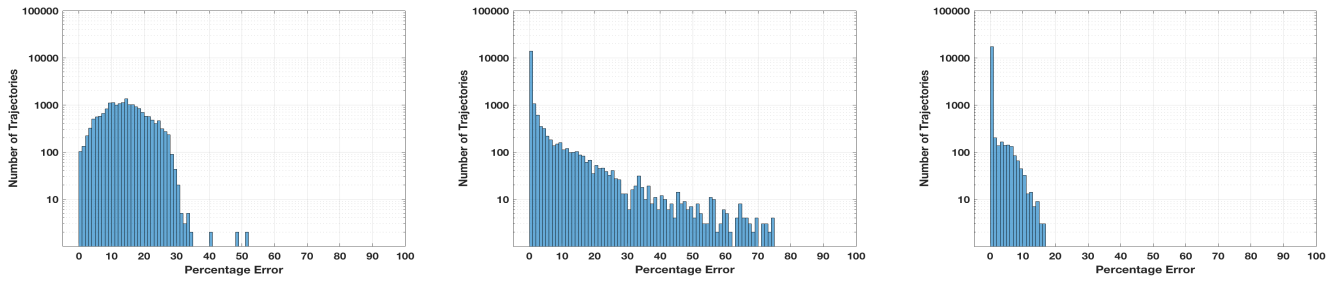
A. Data Set

To evaluate the proposed method, we tested it using Microsoft's GeoLife dataset [41], [42]. Microsoft's GeoLife dataset [41], [42], is collected in Beijing, China, by 182 users over three years. The trajectories are in the 'plt' format, which contains descriptions in the first six lines and descriptive fields for each point of the trajectory. We only selected the latitude, longitude, and timestamp (converted date and time fields to a single field). After this, we stored the first point as Key and multiplied latitude and longitude of all the remaining points of the trajectory by 10^6 in order to get the differences in the integer number. Then we calculated the differences in consecutive points. To calculate the actual distances in the trajectories, we used the haversine formula (Eq. 3).

There are few erroneous points where the same timestamp repeated for multiple consecutive points. Since they affect the calculation of velocity, and acceleration, we have kept the first point and removed the following successive points which had the duplicate timestamp. We have also removed stop points, i.e., successive points where the latitude and longitude are the same but timestamps are different. This happens when the user is stopped at the same place for a few minutes, e.g., waiting at a traffic signal. Similarly, we have kept the first point and removed the following points with the same latitude and longitude for different timestamps. We also ignored trajectories with less than 5 points. In this way, we have cleaned the data.

B. Distance Computation

First, we calculated the actual distance using the real data and the actual haversine formula and stored the distance as the actual distance. For this experiment, we took the center of the Beijing city as (39.913818, 116.363625) and stored it at the LBS provider's side. Then we preprocessed the trajectories by only considering the latitude, longitude, and timestamp fields.



(a) (b) (c)

Fig. 2: Comparison of the average percentage error (ϵ_{T_i}) in distance using (a) GeoSecure (b) GeoSecure-O and (c) GeoSecure-R

We then used the steps described in the algorithm section to calculate the distance using the proposed method.

We also calculate the percentage error ϵ_{P_i} in the distance between two points P_i and P_{i+1} by

$$\epsilon_{P_i} = \frac{|d_{i,i+1} - d'_{i,i+1}|}{d_{i,i+1}} \times 100 \tag{5}$$

and similarly for average velocity, and average acceleration.

We calculated the percentage error between actual distance and the calculated distance using the proposed method for all the points of the trajectory by using Eq. 5. Then for a trajectory T_i we calculated the average percentage error as ϵ_{T_i} as follows:

$$\epsilon_{T_i} = \frac{1}{n} \sum_{i=1}^{n-1} \epsilon_{P_i} \tag{6}$$

where n is the total number of points in the trajectory T.

The mean percentage error for all the trajectories ϵ_{mean} in the dataset is defined as follows

$$\epsilon_{\text{mean}} = \frac{1}{n} \sum_{i=1}^{n} \epsilon_{T_i} \tag{7}$$

where n is the total number of trajectories in the dataset.

To compare the current algorithm with the previous works, GeoSecure [1] and GeoSecure-O [2], used the differences in the consecutive points and calculated the secured distance, as explained in the algorithms. Using this secured distance, we also calculated the average velocity and average acceleration. To compare the accuracy, we use the percentage error between the actual distance and the secured distance calculated using GeoSecure [1], GeoSecure-O [2], and GeoSecure-R.

In Fig. 2, we have plotted histograms of average percentage error for the distance, as explained in Eq. 6, using GeoSecure, GeoSecure-O, and GeoSecure-R methods respectively. We can observe that for the GeoSecure method, for the majority of the trajectories, the average percentage error lies between 0% and 35%, with a mean of 14%. For the GeoSecure-O method, the for the majority of the trajectories, the average percentage error is between 0% and 10%, the number of trajectories after 10% is significantly lower as compared to 0% and 10%. In the case of GeoSecure-R, the average percentage error is close to 0. Hence, we can say that the GeoSecure-R method provides the least average percentage error for securely calculating the distance among GeoSecure, GeoSecure-O, and GeoSecure-R. The mean of the average percentage error (ϵ_{mean}) of all the trajectories in the dataset, for distance calculation for GeoSecure(14.23%), GeoSecure-O (2.6%), and GeoSecure-R (0.39%). We can observe that the proposed method Geo-Secure-R reduces the percentage error for distance significantly as compared to other methods.

The percentage errors in secured average velocity as well as secured average acceleration have similar distribution as the secured distance, since these quantities are derived from the distance. Hence, we can say that the average percentage error reduces significantly in GeoSecure-R as compared to GeoSecure-O, and GeoSecure has the highest percentage error.

C. Trajectory Utility Analysis

For the service S for distance calculation, the outcome is calculated distance. Hence, the equation for trajectory utility (Def. 3) becomes

$$\mathcal{U}_{T',S} = 1 - \frac{|d_{i,j} - d'_{i,j}|}{d_{i,j}}$$

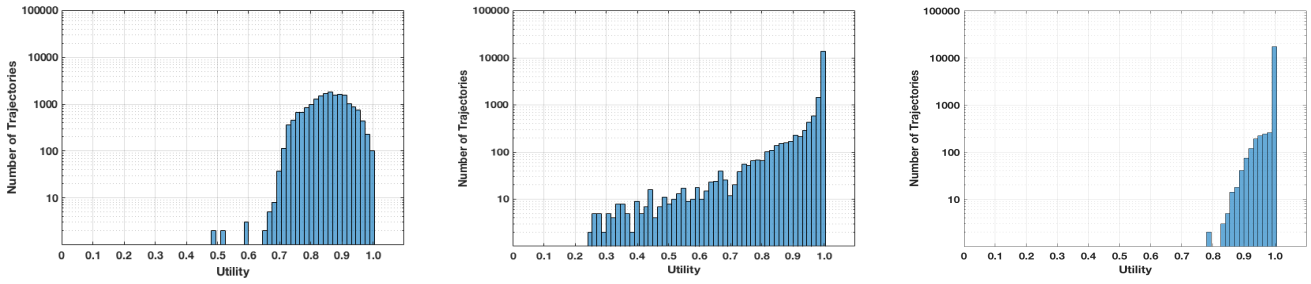
Fig. 3 shows the utility of the service for distance calculation in the form of a histogram for GeoSecure, GeoSecure-O, and GeoSecure-R methods. We can observe that for GeoSecure-R, the majority of the trajectories have the utility between 0.9 to 1.0 with most of the trajectories close to 1.0. On the other hand, the majority of trajectories have utility between 0.7 and 1.0 for GeoSecure, and between 0.8 and 1.0 for GeoSecure-O. Hence, we can say that the proposed method provides better trajectory utility for the distance calculation service.

D. Security and Privacy Analysis

The proposed method preserves privacy and confidentiality of the data. The following preposition explains this in detail.

Proposition 2. *The privacy loss in GeoSecure-R method is negligible (equal to* $\frac{1}{10^{12}}$*).*

Proof. The GeoSecure-R method only sends the differences in the consecutive points, and the exact location is never shared with the service provider. For the privacy analysis, let us consider a geographical region R separated by at least 1 degree



(a) (b) (c)

Fig. 3: Comparison of utility of the distance calculation service using (a) GeoSecure (b) GeoSecure-O and (c) GeoSecure-R

TABLE II: A comparison of privacy loss

Method	$\mathcal{P}_{T',S}$	Location can be
GeoSecure	$1 - \frac{1}{181\times360\times10^{12}}$	Anywhere on earth
GeoSecure-O	$1 - \frac{1}{360\times10^{6}}$	Anywhere on longitude
GeoSecure-R	$1 - \frac{1}{10^{12}}$	Anywhere in the region

latitude and 1 degree longitude. A GPS data point/location is generally defined till the 6^{th} decimal point of latitude and longitude, e.g. (39.123456, 40.123456). Hence, for R, there will be at least $10^6 \times 10^6$ total number of locations. Thus, the probability that a random GPS point P_x is precisely the same as P_y is $\frac{1}{k}$, where k is the total number of possible GPS points in R. As per the Eq. 2, the privacy loss for the actual data $\phi_{T,S}$ is 1 and $\phi_{T',S}$= $\frac{1}{k}$. Hence, the trajectory privacy becomes

$$\mathcal{P}_{T',S} = 1 - \phi_{T',S}$$
$$\phi_{T',S} = \frac{1}{k}$$
$$\text{Hence } \mathcal{P}_{T',S} = 1 - \frac{1}{10^{12}}$$

Since the probability of identifying the starting point of the trajectory in the given region R is very low, it is challenging to identify the exact location. Therefore, the privacy loss is negligible (equal to $\frac{1}{10^{12}}$). Therefore, we can say that the trajectory privacy is preserved by the proposed method as described in Definition 2. ■

In the GeoSecure method, the first point of the trajectory can be anywhere on earth. Hence, adversary will have to choose a point from k points, where k= $181 \times 360 \times 10^{12}$. Similarly, in the GeoSecure-O method, the first point of the trajectory can be anywhere on the latitude, hence the value of k = 360×10^6.

Table II, provides a comparison of trajectory privacy offered by GeoSecure, GeoSecure-O, and GeoSecure-R methods. We can observe that GeoSecure provides the maximum privacy, but the percentage error in distance calculation is maximum as compared to GeoSecure-R, which also provides reasonable trajectory privacy (negligible privacy loss) as well as offers better accuracy as compared to other two approaches.

TABLE III: A comparison with PP-HS protocol [7] for each distance calculation operation between two points

	PP-HS [7]	GeoSecure-R
Operations	$6 \times g^m h^r \mod N$	6 (4 multiplications of 10^6 and 2 subtractions)
Encryption	Encrypt 6 terms	NA
Decryption	Decrypt 1 term	NA
Number of terms	10 encrypted terms	2 integers

The potential applications of the proposed method include fitness trackers, travel mode detection [2], for understanding users' driving habits such as reckless driving, number of miles driven every day, and over-speeding.

E. Comparison with PP-HS protocol [7]

The PP-HS protocol for haversine distance calculation in [7] uses DGK homomorphic encryption scheme. This scheme produces smaller ciphertexts as compared to other homomorphic schemes such as Pailier and ElGamal. As described in [7], the computation overhead for encryption of message m is determined by $[[m]] = g^m h^r \mod N$, where $N = p.q$, p and q are k bit primes, the size of the modulus N is 1024 bits. On the other hand, the proposed method requires only 6 computations on client side, i.e., 4 multiplications by 10^6 and 2 subtractions from the previous point for calculating the distance between two points. We have summarized the comparison in Table III. Hence, we can say that, the computational overhead is significantly lower in the proposed method as compared to PP-HS protocol [7].

V. Conclusion

The GeoSecure-R method calculates users' traveled distance without revealing their actual location with a small loss in accuracy compared to using actual location. The proposed method also reduces the percentage error significantly than the other two methods, GeoSecure and GeoSecure-O. Future work will explore adaptations of the proposed method to other LBS, such as detecting real-time traffic congestion.

REFERENCES

[1] V. Patil, S. Parikh, P. Singh, and P. K. Atrey, "GeoSecure: Towards secure outsourcing of GPS data over cloud," in *The 5th IEEE Conference on Communications and Network Security (CNS)*, Las Vegas, NV, USA, 2017, pp. 495–501.

[2] V. Patil, S. B. Parikh, and P. K. Atrey, "GeoSecure-O: A method for secure distance calculation for travel mode detection using outsourced gps trajectory data," in *The 5th IEEE International Conference on Multimedia Big Data (BigMM)*, Singapore, Singapore, 2019, pp. 348–356.

[3] Y. Zheng, "Trajectory data mining: An overview," *ACM Transactions on Intelligent Systems and Technology*, vol. 6, no. 3, pp. 1–29, May 2015.

[4] S. Gao, J. Ma, C. Sun, and X. Li, "Balancing trajectory privacy and data utility using a personalized anonymization model," *Journal of Network and Computer Applications*, vol. 38, pp. 125–134, 2014.

[5] G. Marias, C. Delakouridis, L. Kazatzopoulos, and P. Georgiadis, "Location privacy through secret sharing techniques," in *The 6th IEEE International Symposium on World of Wireless Mobile and Multimedia Networks, (WoWMoM)*, Taormina-Giardini Naxos, Italy, 2005, pp. 614–620.

[6] D. Seidl, P. Jankowski, and M.-H. Tsou, "Privacy and spatial pattern preservation in masked GPS trajectory data," *International Journal of Geographical Information Science*, vol. 30, no. 4, pp. 785–800, 2016.

[7] J. Šeděnka and P. Gasti, "Privacy-preserving distance computation and proximity testing on earth, done right," in *The 9th ACM Symposium on Information, Computer and Communications Security (ASIACCS)*, Kyoto, Japan, 2014, pp. 99–110.

[8] M. Armstrong, G. Rushton, and D. Zimmerman, "Geographically masking health data to preserve confidentiality," *Statistics in Medicine*, vol. 18, no. 5, pp. 497–525, 1999.

[9] J. Krumm, "A survey of computational location privacy," *Personal and Ubiquitous Computing*, vol. 13, no. 6, pp. 391–399, 2009.

[10] K. Shin, X. Ju, Z. Chen, and X. Hu, "Privacy protection for users of location-based services," *IEEE Wireless Communications*, vol. 19, no. 1, pp. 30–39, 2012.

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

[12] A. Monreale, S. Rinzivillo, F. Pratesi, F. Giannotti, and D. Pedreschi, "Privacy by design in big data analytics and social mining," *EPJ Data Science*, vol. 3, no. 1, pp. 1–26, 2014.

[13] L. Sweeney, "k-anonymity: A model for protecting privacy," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 05, pp. 557–570, 2002.

[14] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam, "l-diversity: Privacy beyond k-anonymity," in *The 22nd IEEE International Conference on Data Engineering (ICDE)*, Atlanta, GA, USA, 2006, pp. 24–24.

[15] N. Li, T. Li, and S. Venkatasubramanian, "t-closeness: privacy beyond k-anonymity and l-diversity," in *The 23rd IEEE International Conference on Data Engineering (ICDE)*, Istanbul, Turkey, 2007, pp. 106–115.

[16] O. Abul, F. Bonchi, and M. Nanni, "Never walk alone: uncertainty for anonymity in moving objects databases," in *The 24th IEEE International Conference on Data Engineering (ICDE)*, Cancun, Mexico, 2008, pp. 376–385.

[17] P. Xiong, T. Zhu, W. Niu, and G. Li, "A differentially private algorithm for location data release," *Knowledge and Information Systems*, vol. 47, no. 3, pp. 647–669, 2016.

[18] M. Andrés, N. Bordenabe, K. Chatzikokolakis, and C. Palamidessi, "Geo-indistinguishability: differential privacy for location-based systems," in *The ACM Conference on Computer Communications Security (CCS)*, Berlin, Germany, 2013, pp. 901–914.

[19] B. Hoh, M. Gruteser, H. Xiong, and A. Alrabady, "Achieving guaranteed anonymity in GPS traces via uncertainty-aware path cloaking," *IEEE Transactions on Mobile Computing*, vol. 9, no. 8, pp. 1089–1107, 2010.

[20] A. Gutscher, "Coordinate transformation-a solution for the privacy problem of location based services?" in *The 20th IEEE International Parallel & Distributed Processing Symposium (IPDPS)*, Rhodes Island, Greece, 2006, pp. 7–pp.

[21] R. Di Pietro, R. Mandati, and N. Verde, "Track me if you can: transparent obfuscation for location based services," in *The IEEE World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, Madrid, Spain, 2013, pp. 1–9.

[22] D. Douglas and T. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *Cartographica: The International Journal for Geographic Information and Geovisualization*, vol. 10, no. 2, pp. 112–122, 1973.

[23] J. Muckell, J.-H. Hwang, V. Patil, C. T. Lawson, F. Ping, and S. S. Ravi, "SQUISH: an online approach for GPS trajectory compression," in *The 2nd ACM International Conference on Computing for Geospatial Research and Applications (COM.Geo)*, vol. 13, Washington DC, USA, 2011, pp. 1–8.

[24] R. Horspool, "Improving LZW (data compression algorithm)," in *The 1st IEEE Data Compression Conference (DCC)*, Snowbird, Utah, 1991, pp. 332–341.

[25] D. Xu, Y. Wang, L. Jia, G. Zhang, and H. Guo, "Compression algorithm of road traffic spatial data based on lzw encoding," *Journal of Advanced Transportation*, vol. 2017, 2017.

[26] P. Cudre-Mauroux, E. Wu, and S. Madden, "Trajstore: An adaptive storage system for very large trajectory data sets," in *The 26th IEEE International Conference on Data Engineering (ICDE)*, Long Beach, CA, USA, 2010, pp. 109–120.

[27] A. Nibali and Z. He, "Trajic: An effective compression system for trajectory data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 11, pp. 3138–3151, 2015.

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

[29] P. Hu and S. Zhu, "Poster: Location privacy using homomorphic encryption," in *The 12th International Conference on Security and Privacy in Communication Systems (SecureComm)*, Guangzhou, China, 2016, pp. 758–761.

[30] S.-D. Li and Y.-Q. Dai, "Secure two-party computational geometry," *Journal of Computer Science and Technology*, vol. 20, no. 2, pp. 258–263, 2005.

[31] P. Asuquo, H. Cruickshank, J. Morley, C. P. Anyigor Ogah, A. Lei, W. Hathal, and S. Bao, "Security and privacy in location-based services for vehicular and mobile communications: An overview, challenges and countermeasures," *IEEE Internet of Things*, 2018.

[32] N. Li, N. Zhang, S. Das, and B. Thuraisingham, "Privacy preservation in wireless sensor networks: A state-of-the-art survey," *Ad Hoc Networks*, vol. 7, no. 8, pp. 1501–1514, 2009.

[33] M. Gruteser and D. Grunwald, "Anonymous usage of location-based services through spatial and temporal cloaking," in *The 1st ACM International Conference on Mobile Systems, Applications and Services (MobiSys)*, San Franscisco, CA, USA, 2003, pp. 31–42.

[34] G. Ghinita, P. Kalnis, A. Khoshgozaran, C. Shahabi, and K.-L. Tan, "Private queries in location based services: anonymizers are not necessary," in *The ACM International Conference on Management of Data (SIGMOD)*, Vancouver, BC, Canada, 2008, pp. 121–132.

[35] A. Narayanan, N. Thiagarajan, M. Lakhani, M. Hamburg, D. Boneh *et al.*, "Location privacy via private proximity testing." in *The Network and Distributed System Security Symposium (NDSS)*, vol. 11, 2011.

[36] P. Hallgren, C. Orlandi, and A. Sabelfeld, "Privatepool: Privacy-preserving ridesharing," in *The 30th IEEE Computer Security Foundations Symposium (CSF)*, Santa Barbara, CA, 2017, pp. 276–291.

[37] A. C. Pesara, V. Patil, and P. K. Atrey, "Secure computing of GPS trajectory similarity: a review," in *The 2nd LocalRec Workshop at The 26th ACM International Conference on Advances in Geographic Information Systems (SIGSPATIAL)*, Seattle, WA, USA, 2018, pp. 3:1–3:7.

[38] V. Patil, P. Singh, S. Parikh, and P. K. Atrey, "GeoSClean: Secure cleaning of GPS trajectory data using anomaly detection," in *The 1st IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, Miami, FL, USA, 2018, pp. 166–169.

[39] J. Inman, *Navigation and Nautical Astronomy, for the Use of British Seamen*. F. & J. Rivington, 1849.

[40] R. Sinnott, "Virtues of the haversine," *Sky and Telescope*, vol. 68, p. 159, 1984.

[41] Y. Zheng, X. Xie, and W. Y. Ma, "Geolife: A collaborative social networking service among user, location and trajectory." *IEEE Data Engineering Bulltein*, vol. 33, no. 2, pp. 32–39, 2010.

[42] Y. Zheng, L. Zhang, X. Xie, and W. Y. Ma, "Mining interesting locations and travel sequences from GPS trajectories," in *The 18th ACM International Conference on World Wide Web (WWW)*, Madrid, Spain, 2009, pp. 791–800.