Protecting privacy in trajectories with a user-centric approach

CRISTINA ROMERO-TRIS, Internet Interdisciplinary Institute (IN3), Universitat Oberta de Catalunya (UOC), CYBERCAT-Center for Cybersecurity Research of Catalonia

DAVID MEGÍAS, Internet Interdisciplinary Institute (IN3), Universitat Oberta de Catalunya (UOC), CYBERCAT-Center for Cybersecurity Research of Catalonia

The increased use of location-aware devices, such as smartphones, generates a large amount of trajectory data. These data can be useful in several domains, like marketing, path modeling, localization of an epidemic focus, etc. Nevertheless, since trajectory information contains personal mobility data, improper use or publication of trajectory data can threaten users' privacy. It may reveal sensitive details like habits of behavior, religious beliefs, and sexual preferences. Therefore, many users might be unwilling to share their trajectory data without a previous anonymization process. Currently, several proposals to address this problem can be found in the literature. These solutions focus on anonymizing data before its publication, i.e., when they are already stored in the server database. Nevertheless, we argue that this approach gives the user no control about the information she shares. For this reason, we propose anonymizing data in the users' mobile devices, before they are sent to a third party. This paper extends our previous work which was, to the best of our knowledge, the first one to anonymize data at the client side, allowing users to select the amount and accuracy of shared data. In this paper, we describe an improved version of the protocol, and we include the implementation together with an analysis of the results obtained after the simulation with real trajectory data.

Additional Key Words and Phrases: Trajectory anonymization, User-centric protocol, Privacy

ACM Reference format:
Cristina Romero-Tris and David Megías. 2018. Protecting privacy in trajectories with a user-centric approach. *ACM Trans. Knowl. Discov. Data.* 12, 6, Article 67 (August 2018), 28 pages.
DOI: 10.1145/3233185

1 INTRODUCTION

Location-aware technologies such as global positioning system (GPS), radio frequency identification (RFID) or location-based services (LBS) have significantly developed over the last few years. The expansion of mobile networking and the extended capabilities of mobile devices have caused the amount of data related to trajectories to increase to a great extent.

Mining and analyzing these spatio-temporal trajectory datasets can provide a valuable service. There are many examples of domains in which this information can be useful: traffic monitoring (e.g., inferring traffic jams), location-based advertising (e.g., sending discounts to nearby potential customers), homeland security (e.g., detecting migration anomalies), or tracking infections. Most existing works related to trajectory data mining –e.g. (Reumers et al. 2013; Tanuja and Govindarajulu 2016)– have an initial phase of trajectory data collection to create databases. These databases

are then used in several fields like urban planning, transportation, business, environment or public safety, among others (Mazimpaka and Timpf 2016).

However, trajectory data often contain detailed information about individuals. Knowledge of mobility data, in some cases combined with quasi-identifiers (gender, age, postal code, etc.), may reveal sensitive data that threaten users' privacy (e.g. information about home addresses, lifestyle, preferences, religious beliefs, political ideology, etc.) Some users may be unwilling to share their data if this causes a loss of privacy.

Consequently, there is a trade-off between the utility of trajectory data and the privacy threat caused by disclosing them. The main challenge that arises in these systems is to preserve users' privacy without losing data utility. For this reason, there is an emergent field of the literature that addresses this situation. Most existing solutions, e.g. (Abul et al. 2008; Terrovitis and Mamoulis 2008; Yarovoy et al. 2009), apply anonymization techniques to trajectory data already stored in databases, before their publication. This means that the methods are applied to already collected data, before analyzing or releasing them to a third party. Therefore, users are supposed to share their real information for storage.

Nevertheless, we argue that this solution is not secure enough. Non-anonymized information would rather not be sent to the database. Instead, in our proposal, true trajectory or location data are not stored in the server databases, but in the user's device. We claim that this user-centered approach has major advantages over the classical techniques, since users can see and control the trajectory information that they share with the server and, hence, they do not need to trust any server for preserving their privacy.

The problem that we are trying to solve is having a user who wants to help a server collect trajectory data. She agrees to share the trajectory data stored in her own device. However, she wants to do it without sacrificing her privacy. She does not trust the server to truthfully perform the anonymization, so she searches a solution to protect her trajectory data before revealing it.

The solution that we propose for this problem is to perform the anonymization inside the user's device. We intend to give the user full control over how much information she will release, and how accurate this information will be. In order to do so, the user must be able to distort or even completely remove those points of the trajectory which threaten her privacy. In order to distort them, we propose a model in which each trajectory point is initially defined by a longitude, latitude and timestamp. After anonymization, the point becomes a cylinder, which contains the real initial point, although an external viewer cannot identify exactly where. It is the user who decides, by setting some parameters, the volume of this cylinder. Therefore, she can choose a larger radius and height for those points that may pose a bigger threat for her privacy.

Besides the trajectory model, we also propose to increase privacy by applying k-anonymity. This means that the user will only submit a trajectory (or a part of it) if it already appears in the server database at least k times. This prevents a malicious adversary, who has some background knowledge of the user's location at a certain time, to unequivocally re-identify the victim in the database.

However, in order to know if a trajectory appears k times in the database, the user must query the database. Submitting the query directly to the server may compromise the user's identity. For this reason, we propose a distributed cryptographic algorithm, where a group of users collaborate in order to securely exchange their queries. After that, users will submit queries on behalf of other members of the group, so that the server cannot link a query containing a trajectory to its real user.

In summary, the solution that we propose includes a trajectory model combined with k-anonymization to protect user's privacy, and a distributed cryptographic protocol to perform secure queries. In (Romero-Tris and Megías 2015),

we presented an initial draft of our work. Since then, we have improved some parts of the protocol, but the main difference is that we have implemented it and simulated the system using real trajectory data, obtaining and presenting results regarding data utility and data privacy. Consequently, this paper describes the current version of the protocol and explains the implementation and the obtained results.

1.1 Related work

In this section, we review the state of the art of privacy-preserving trajectory data techniques. The ubiquitousness of trajectory data has caused some recent works to study these techniques from different perspectives.

(Abul et al. 2008) propose the (k, δ)-anonimity model, which modifies a location polyline to be represented by a single cylinder of radius δ. Then, k trajectories co-localized inside the same cylinder are indistinguishable from each other.

(Terrovitis and Mamoulis 2008) propose an algorithm that suppresses the existence of certain points in the trajectories. The challenge in this case is how to find the optimal set points to delete, with the minimum information loss. The authors propose a greedy heuristic that assumes that all the adversarial knowledge is known before data publication. Similarly, (Pensa et al. 2008) propose to remove frequent sequential patterns. They transform sequences by adding, deleting, or substituting some points of the trajectory.

(Yarovoy et al. 2009) employ the Hilbert curve described in (Hilbert 1891) in order to map a multi-dimensional space to one dimension. The purpose is to find the nearest neighbors at every point of the trajectory. Then, the neighbors are used to create anonymization groups to generalize the trajectory data of each member.

All these works are limited to privacy protection on already collected data. The proposed algorithms work on the server side, before data publication. Nevertheless, we argue that trajectory anonymization would rather be performed a step earlier, in the user side.

There are other proposals related to our work that provide privacy for users that employ location-based services within the location privacy category. For example, the work in (To et al. 2014) presents a solution for workers who have to travel to a specified nearby location. In order to know which workers are nearer, they need to disclose their current location. The authors propose a solution to do this while preserving their privacy, called Spatial crowdsourcing. This proposal is based on the spatial decomposition of an area into zones which contain a certain number of workers. It is similar to our proposal in the sense that workers are assumed to send their location information from their smartphones or similar devices to an untrusted server. However, it differs from our work because the anonymization is only applied to a single location and in real time. It does not consider trajectory data anonymization, that is, users that have a collection of historic location data.

Another interesting work is presented in (Beresford and Stajano 2004). This work is oriented to applications offering a real-time service to users. The applications define zones of interest (e.g., a supermarket), and they want to contact the users there. At the same time, users define a list of applications that interest them. Then, there is a middleware that limits the location information received by the application, protecting thus users' privacy. The proposed solution is based on pseudonyms: every time a user changes her zone, she is assigned with a new pseudonym. This solution does not work in our scenario, since it protects short-term locations. Changing pseudonyms prevents the creation of a trajectory of more than two points.

The work presented in (Dittler et al. 2016) is somehow different from the mentioned above. This work protects location privacy of users in a different scenario. It protects users' privacy in front of a service provider. In our scenario, we consider users who intend to share their trajectory data, but while maintaining a certain level of privacy

The proposal of (Gruteser and Liu 2004) allows users to define "sensitive areas" where their location should not be tracked. Although the objective of the system is to protect a specific current location, and not to anonymize a trajectory, the idea of letting users participate in the process is somehow similar to our proposal. However, this work has several shortcomings. Firstly, a trusted third party is necessary to allow or deny access to a user location. Secondly, sensitive areas are defined by users by a set of policies that affect rectangles in a pre-defined map. Additionally, the system is not flexible, i.e., a location is defined as private or public, it is not possible to distort a location so that some utility is preserved. Furthermore, if the user shares several locations close to a private one in a short period of time, it is likely to infer the location that she is trying to hide. Consequently, the advantage of our approach is that users are able to select the amount and accuracy of shared information before it is stored in the database.

Another well-known approach to provide privacy in databases is differential privacy. With this technique, regardless of the background knowledge, an adversary with access to a database comes to the same conclusion whether and individual's data is included in the database or not. Differential privacy has already been applied to privacy-preserving location data publishing (Andrés et al. 2013; ElSalamouny and Gambs 2016), and trajectory data publishing approaches (Chen et al. 2012; He et al. 2015; Jiang et al. 2013; Li et al. 2017) (as previously described, trajectory anonymization is more related to our work than location anonymization). The most commonly used approach to achieve differential privacy is the Laplace mechanism (Dwork et al. 2006), adding random noise sampled from the Laplace distribution to the coordinates of each location.

For example, currently, the most recent work in differential privacy for trajectory data publishing is (Li et al. 2017). In this work, the authors present a trajectory data differentially private publishing scheme that works as follows. First of all, they map all the existing locations in the database and group those which have occurred at the same moment. Inside each group, they use k-means clustering in order to create sub-groups of close locations (using the Euclidean distance) and find the centroid of each one of them. At the end of this phase, *generalized* trajectories are obtained. A *generalized* trajectory differs from the original one because each location is substituted by the centroid of her sub-group found with k-means. Then, the authors develop a mathematical function to calculate and apply Laplace noise to each trajectory. As a result, the scheme obtains a trajectory dataset that satisfies the differential privacy definition.

Other previous similar proposals applying differential privacy to trajectories (Chen et al. 2012; He et al. 2015; Jiang et al. 2013) might differ in the way trajectories are generalized or the way that random noise is generated. Nevertheless, they all have the same shortcomings with respect to our proposal:

- Existing differential privacy techniques are aimed at protecting finished databases. All the calculations and algorithms need the whole set of records as input. This means that anonymization is not performed during data collection, but once the database is complete. Differential privacy schemes are commonly referred to privacy-preserving data publishing techniques, since they are applied before publication (and after collection).

  Nevertheless, our scenario is meant for a database that grows progressively, where new users can add new records at any time. Dynamic (or stream) construction of databases using differential privacy is a work in progress. Some proposals (Dwork et al. 2010, 2014) remark that "Up to this point, research on differentially private data analysis has focused on the setting of a trusted curator holding a large, static, data set; thus every computation is a "one-shot" object", and study how to do it with non-static data. Nevertheless, this study is

very preliminary, since they focus on analysing how to do it with simple data. More specifically, their objective is to monitor a stream of D bits, and continually maintain an approximation of the number of 1's that have been observed so far. However, trajectory data is much more complex and it is not yet feasible to apply these techniques for a growing database of trajectory data.

- It does not allow interaction with users. Users do not participate in the process of anonymization of the points of the trajectory. One of the objectives of our work is to give users the control and resposibility of their data anonymization. To the best of our knowledge, with differential privacy the random noise addition can only be applied on the server side.

In conclusion, differential privacy can protect trajectory data, but might not be directly applied to our scenario, where we want to collect users' data progressively, and where we want to allow users to actively participate in the anonymization process.

1.2 Contribution and plan of this paper

This paper builds up on our previous work presented in (Romero-Tris and Megías 2015), which is, to the best of our knowledge, the first anonymizing trajectory scheme that works on the user side. The system collects trajectory data in the user's device and anonymizes them before sending it to the server for storage or analysis. With respect to (Romero-Tris and Megías 2015), the contributions of this paper are the following:

- We have analyzed why other alternatives to k-anonymization (l-diversity, t-closeness or differential privacy) are not suitable for our scenario.
- We have modified some parts of the protocol. More specifically, we have improved the sub-trajectory extraction algorithm in order to make it more efficient. We have also included a final optional step called "Sub-trajectory identifier retrieval", which allows an external entity to retrieve trajectories from a certain user, in scenarios where this might be necessary.
- We have provided a complete example of use of our protocol so that it can be easier understood.
- We have extended the privacy analysis with respect to dishonest users, formulating a theorem for the cases where an attacker can recover the secret key. We have also extended the privacy analysis with respect to a dishonest server, considering the consequences of a malicious server that provides fake responses.
- Regarding the protocol, the way that trajectory points are anonymized is improved. In (Romero-Tris and Megías 2015), we proposed to transform the exact latitude and longitude to a circle which contains them, although we do not know where. However, we worked with a Cartesian coordinate system, and the probability distribution inside the circle was not uniform. As a result, an attacker could guess which points inside the circle were more likely the exact latitude and longitude. Therefore, we have improved this step by using a polar coordinate system. Choosing a random angle and a random distance, the new center of the circle reveals no information about the exact real location.
- We have implemented the protocol as a Java application. More specifically, we have implemented the application that runs the server which holds the database, and the applications that users execute.
- We have simulated the implemented protocol using real data. We have used real data extracted from GeoLife GPS trajectories (Zheng et al. 2009), which contains trajectory data from 182 users in a period of over three years. The output of the simulations are statistics that show how much information is sent to the server for several parameter configurations.

This paper is organized as follows: Section 2 defines relevant concepts for the proposed system, including the employed cryptographic blocks. Section 3 describes our proposal in detail. Privacy is analyzed in Section 4 from a theoretical point of view. This section also describes a way of analysing the utility of the anonymized trajectory data. The description of the simulations and the results obtained using real data is provided in Section 5. Finally, Section 6 concludes the paper.

2 CONTEXT DESCRIPTION

This section outlines the background concepts which are needed to understand how the system works. First of all, the threats to users' privacy are described. Then, the proposed solutions to these threats are briefly depicted. Finally, we describe the anonymization model and the cryptographic building blocks that are used in the protocol later described in Section 3.

2.1 Privacy threats

We assume an scenario in which a server is trying to *collect* trajectory data from a population. Explicit identifiers (e.g., name, SSN, etc.) have been removed. For every user, the only stored information is a randomly assigned identifier associated with her trajectory. We also assume that the server responds truthfully to requests regarding the data stored in its database.

Users who wish to collaborate with this server are willing to do it if this does not mean losing their privacy. For this reason, we have identified three main sources of privacy threats:

(1) *The correct anonymization of location points.* Users must not be forced to delegate the anonymization of the visited points to an external entity. Instead, they should decide which information they release and how accurate this information is. Disclosing non-anonymized data outside a personal device is a threat to users' privacy.
(2) *The uniqueness of a trajectory.* An attacker with background knowledge may re-identify a user in the database if a unique record is found. If no other user in the database has the same trajectory, an attacker may use background knowledge to identify her victim. Successfully associating a record in the database to a user is also a threat to users' privacy.
(3) *The trust on the server.* In order to protect her privacy, a user may need a certain degree of interaction with the server. Releasing too much information to the server may be a threat to users' privacy.

2.2 Privacy protection

In response to these threats, it is our mission to respond and give users alternatives that allow them to collaborate with a server without losing their privacy. More specifically, for the three listed threats, we propose these three privacy-protection solutions:

(1) An anonymization model inside the personal device. This anonymization model gives uncertainty to the exact time and location of each point of the trajectory. This model allows the user to decide how much uncertainty is given to each point. The anonymization is performed before any data comes out of the personal device.
(2) An anonymization model inside the database. We assume an attacker who has some background knowledge about some locations that the victim has visited at a certain moment. In order to prevent this attacker from re-identifying a user because of a unique trajectory, we use the k-anonymization model. The idea is that if only

one user in the database has visited some known places at the same moment, the attacker can easily identify the victim's record. On the other hand, if k users have visited the set of locations at similar moments, the probability of correctly linking the victim to its record is $1/k$, achieving then the definition of k-anonymity (Samarati and Sweeney 1998). Following the k-anonymity model, a user will not submit a trajectory or a subset of points in her trajectory, unless appearing at least k times in the database.

Finding all the subsets of points in a trajectory is a "combination without repetition" problem. Mathematically, we have to find all the κ-combinations for all different κ. A κ-combination of a set is a subset of κ distinct elements in the set. If the size of the set is η, then the κ-combination is often denoted in elementary combinatorics texts by $C(\eta, \kappa)$.

In our scenario, we have an anonymized trajectory T' of size $|T'|$, and we want to find all the combinations of points of size at least 2, but less or equal to $|T'|$. Consequently, for a trajectory T' we have to find all the subsets s:

$$\forall \kappa, \text{with } 2 \leq \kappa \leq |T'|, s \in \{C(|T'|, \kappa)\},$$

and the number of subsets is calculated with the formula:

$$\sum_{2 \leq \kappa \leq \eta} \binom{\eta}{\kappa} = \sum_{0 \leq \kappa \leq \eta} \binom{\eta}{\kappa} - \binom{n}{1} - \binom{n}{0} = 2^n - n - 1.$$

For example, for a trajectory with four points a→b→c→d, we would obtain $2^4 - 4 - 1 = 11$ different sub-trajectories, which are: a→b, a→c, a→d, b→c, b→d, c→d, a→b→c, a→b→d, a→c→d, b→c→d, a→b→c→d.

It can be observed that the number of possible sub-trajectories exponentially increases for larger trajectories. For this reason, in a more practical scenario, we suggest to use an algorithm that generates random sequences of sub-trajectories to be checked. If these sub-trajectories are common to at least k users in the database, then the whole trajectory can be disclosed. The number of sub-trajectories to be checked is a parameter defined by each user. In more hostile scenarios, the whole set of sub-trajectories should be generated and checked.

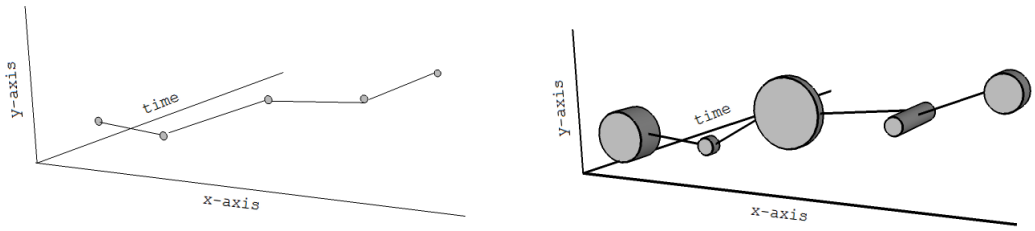
(3) A cryptographic distributed protocol: A user cannot directly ask the server if a trajectory or sub-trajectory appears k times in the database. The user cannot be sure if the server will record it (associated to her identity) even if it appears less than k times. For this reason, we propose a solution in which a group of users collaborate and submit queries to the server in behalf of other members of the group. The motivation for a user to do this is that another member of the group will submit her queries in her place. The submitted queries are in fact the sub-trajectories of each user, so they must be encrypted to prevent any member of the group from learning any information about another user. At the end of the protocol, neither the server nor any member of the group can associate a sub-trajectory to its original user, but each user obtains the server responses for her queries.

2.3 Anonymization model

This section describes the anonymization model that users execute inside their personal devices. First of all, we define our model of trajectory prior to and after the anonymization, then we give some definitions regarding sub-trajectories (i.e., subsets of points in the trajectory).

Definition 2.1 (Trajectory). A trajectory T of length $|T|$ is an ordered list of spatio-temporal points

$$\{(x_1, y_1, t_1), (x_2, y_2, t_2), \ldots, (x_{|T|}, y_{|T|}, t_{|T|})\}$$



(a) Trajectory (b) Anonymized trajectory

Fig. 1. Schematic representation of a trajectory before and after anonymization

where (x_i, y_i, t_i) means that the user was at a physical location with Cartesian coordinates (x_i, y_i) at instant t_i. During the time segment $[t_i, t_{i+1}]$ the user is assumed to move along a straight line from (x_i, y_i) to (x_{i+1}, y_{i+1}).

Figure 1(a) represents the definition of a five point trajectory. The three-dimensional space represents the time and the Cartesian coordinates of the position (abscissas and ordinates).

Definition 2.2 (Uncertain point). Anonymization of a specific spatio-temporal point (x_i, y_i, t_i), results into another vector $(cx_i, cy_i, r_i, a_i, b_i)$ where (cx_i, cy_i) are the Cartesian coordinates of the center of a circle of radius r_i that contains (x_i, y_i), and $[a_i, b_i]$ is a time interval that contains t_i.

Definition 2.3 (Anonymized trajectory). An anonymized trajectory T' of length $|T'|$ is an ordered list of *uncertain point* vectors:

$$\{(cx_1, cy_1, r_1, a_1, b_1), (cx_2, cy_2, r_2, a_2, b_2), \ldots, (cx_{|T'|}, cy_{|T'|}, r_{|T'|}, a_{|T'|}, b_{|T'|})\}.$$

During the time segment between $[a_i, b_i]$ and $[a_{i+1}, b_{i+1}]$, the user is assumed to move along a straight line from some point inside the circle defined by (cx_i, cy_i, r_i) to some point inside $(cx_{i+1}, cy_{i+1}, r_{i+1})$.

Figure 1(b) represents the same trajectory of Figure 1(a) after being anonymized. The anonymization transforms the exact location into a circle of a variable radius, and the exact instant into a time interval. Thus, in this case, for each spatio-temporal point, a cylinder is obtained.

In the example of Figures 1(a) and 1(b), T and T' have the same size. Nevertheless, this is not a system requirement: $|T|$ and $|T'|$ might differ. In fact, the user is not compelled to anonymize all the spatio-temporal points; she can remove any point from it. This is useful and recommended for sensitive locations, like places of worship or a demonstration, but also for frequently visited points, like home or work.

Definition 2.4 (Anonymized sub-trajectory). Let us consider an already anonymized trajectory T'. Then, an anonymized sub-trajectory s of size $|s| \leq |T'|$ is an ordered subset of the vectors composing T'. The conditions to be fulfilled are the following:

- To prevent sub-trajectories with a single point, the size of the sub-trajectory must be $|s| > 1$.
- The order of the vectors in s must be the same as in T'.

Definition 2.5 (Similar anonymized sub-trajectories). Given two anonymized trajectories s_1 and s_2 of sizes $v = |s_1|$ and $\eta = |s_2|$, respectively:

$$s_1 = (cx_{11}, cy_{11}, r_{11}, a_{11}, b_{11}), (cx_{21}, cy_{21}, r_{21}, a_{21}, b_{21}), \ldots, (cx_{v1}, cy_{v1}, r_{v1}, a_{v1}, b_{v1}),$$
$$s_2 = (cx_{12}, cy_{12}, r_{12}, a_{12}, b_{12}), (cx_{22}, cy_{22}, r_{22}, a_{22}, b_{22}), \ldots, (cx_{\eta 2}, cy_{\eta 2}, r_{\eta 2}, a_{\eta 2}, b_{\eta 2}),$$

we define two system parameters θ_L and θ_T as the maximum distance to consider two points similar in terms of location and time, respectively. Then, s_1 and s_2 are similar if the following two conditions are fulfilled:

(1) $v = \eta$.
(2) For $i = 1, 2, \ldots, v$:
   (a) $\sqrt{(cx_{i1} - cx_{i2})^2 + (cy_{i1} - cy_{i2})^2} \leq (r_{i1} + r_{i2} + \theta_L)$. This means that the Euclidean distance between the circles that include the exact location is lower that θ_L.
   (b) $((b_{i2} + \theta_T) \geq a_{i1})$ and $(b_{i2} + \theta_T) \leq b_{i1}))$ or $((b_{i1} + \theta_T) \geq a_{i2})$ and $(b_{i1} + \theta_T) \leq b_{i2}))$. This means that the time intervals are separated at most by θ_T, starting the i-th point of s_2 before the i-th point of s_1, or the other way round.

2.4 Cryptographic building blocks

As stated before, our system includes a distributed protocol in order to exchange trajectories inside the group. In this section, we describe cryptographic building blocks underlying this distributed protocol.

2.4.1 n-out-of-n threshold ElGamal encryption. In an n-out-of-n threshold ElGamal encryption (Desmedt and Frankel 1990), n users share a public key y and the corresponding unknown private key α is divided into n shares α_i. Using this protocol, a certain message m can be encrypted with the public key y and it can only be decrypted if all n users collaborate in the process. The key generation, encryption and decryption processes are described below.

- *Key generation*

  First, a large random prime number p is generated, where $p = 2q + 1$ and q is a prime number too. Also, a generator g of the multiplicative group $\mathbb{Z}_q^*$ is chosen. Then, each user generates a random private key $\alpha_i \in \mathbb{Z}_q^*$ and publishes $y_i = g^{\alpha_i}$. The common public key is computed as $y = \prod_{i=1}^{n} y_i = g^{\alpha}$, where $\alpha = \alpha_1 + \cdots + \alpha_n$.
- *Message encryption*

  Message encryption can be performed using the standard ElGamal encryption function (ElGamal 1985). Given a message m and a public key y, a random value Φ is generated and the ciphertext is computed as follows:

$$E_y(m, \Phi) = c = (c1, c2) = (g^{\Phi}, m \cdot y^{\Phi}).$$

- *Message partial decryption*

  A cyphertext encrypted under ElGamal (before any operation is performed) is denoted as $E_y(m, \Phi) = (c1, c2)$. This ciphertext is encrypted with a public key $y = g^{\alpha_1 + \cdots + \alpha_n}$, where α_i is the private key generated by the user U_i. In order to partially decrypt the ciphertext, the user U_i employs her private key as follows:

$$c2' = \frac{c2}{c1^{\alpha_i}}.$$

  The result of this operation is used to build another ciphertext, denoted as $E_{y'}(m, \Phi) = (c1, c2')$. In this case, the ciphertext is encrypted with a public key $y' = g^{\alpha_{(i+1)} + \cdots + \alpha_n}$.

With this operation, users can individually contribute to the decryption of the queries during the execution of the protocol. When a user partially decrypts a ciphertext, this ciphertext is no longer encrypted with her share. Thus, with the contribution of all the users, the ciphertexts are finally decrypted.

2.4.2 ElGamal re-masking. The re-masking operation performs some computations over an encrypted value. In this way, its cleartext does not change but the re-masked message is not linkable to the same message before re-masking.

Given an ElGamal ciphertext $E_y(m, \Phi)$, it can be re-masked (Abe 1999) by computing:

$$E_y(m, \Phi) \cdot E_y(1, \Phi').$$

For $\Phi' \in \mathbb{Z}_q^*$ randomly chosen and where $\cdot$ stands for the component-wise scalar product. The resulting ciphertext corresponds to the same cleartext m.

3 PROPOSED SYSTEM

3.1 Entities

Several different entities participate in the protocol:

- The server $\boldsymbol{\mathcal{S}}$. It is a semi-trusted central entity intended to collect data trajectory information from the users' devices. It is semi-trusted in the sense that it is assumed to truthfully respond to requests, but has no motivation to preserve users' privacy. In our scenario, the server has two functions:
  (1) Store anonymized trajectory data in the database.
  (2) Respond users' requests about the uniqueness or rareness of a trajectory. The objective of the privacy-preserving process is to obtain k-anonymity. Therefore, in order for a user to disclose a part of a trajectory, she must be sure that it cannot be used to re-identify her. The server helps the user by letting her know how many times (ϕ) the trajectory already appears in the database.
- The users. They are the owners of the devices that store the trajectory data. They have absolute control over the quantity and the degree of distortion of the information shared with the server.
- A legal trusted authority $\mathcal{A}$. Let us consider an scenario where a health department is investigating the focus of an epidemic. Trajectory data of some affected users may be helpful for the research. Instead of querying each user's device that contains the information, it would be better to retrieve information from a centralized entity. For these cases, we propose to use a legal trusted authority $\mathcal{A}$. The authority database does not contain any trajectory, just trajectory identifiers associated to each user. This entity only participates in the last step of the protocol, and only in scenarios where this may be applicable.

3.2 Protocol overview

We assume that points are added to the trajectory according to each user's device. Some devices may record a location and time when a request to a Location-Based Service (LBS) is issued. Some other devices may record it at specific time intervals or every time that user stays in a new location longer than a certain amount of time. The specific way of adding a new location to the trajectory is out of the scope of this paper.

The privacy-preserving process consists of several phases:

(1) *Creation of an anonymization group.* The user must associate with other users to create the group that will be employed in later phases of the protocol.

(2) *Trajectory anonymization.* The user adjusts the generalization parameters for every spatio-temporal point stored in her device, obtaining an anonymized trajectory T'.
(3) *Sub-trajectories extraction.* In this phase, from the anonymized trajectory T', τ sub-trajectories are extracted.
(4) *Fake sub-trajectories generation.* In this step of the process, τ' other "fake" sub-trajectories that do not appear in T' are generated.
(5) *Distribution of sub-trajectories.* The real sub-trajectories, extracted in Step 3, and the fake sub-trajectories, generated in Step 4, are distributed among the group of users created in Step 1.
(6) *Sub-trajectories submission and response forwarding.* For every received sub-trajectory, the user sends it inside a request to the server $\mathcal{S}$. The server responds according to its database, sending the number of times ϕ that the sub-trajectory already appears. This response is then broadcast. Each user only minds the responses corresponding to the trajectories that she generated and discards the rest.
(7) *Anonymized trajectory trimming.* Responses related to fake sub-trajectories are automatically ignored.Then, according to the k-anonymity definition, each user only shares a sub-trajectory if it means that it will appear in the database at least k times. Consequently, the user will remove from T' all the sub-trajectories where $\phi < k$. The user sends the resulting anonymized T' to $\mathcal{S}$, so that it is stored in its database and she obtains k-anonymity.
(8) *Sub-trajectory identifier retrieval.* For each stored sub-trajectory, the server sends a unique identifier of the sub-trajectory to the user. This identifier can be used, later on, by the user to recover her particular anonymized sub-trajectories from the server, as required for some applications (such as the location of the focus of an epidemic). For this process, a legal trusted authority $\mathcal{A}$ is used. This authority collects sub-trajectory identifiers to be later used in scenarios where it is necessary to anonymously retrieve sub-trajectories belonging to a specific user.

3.3 Detailed protocol

The proposed system is now described in higher detail. We assume that the user U_i's device already contains her trajectory $T = \{(x_1, y_1, t_1), (x_2, y_2, t_2), \ldots, (x_{|T|}, y_{|T|}, t_{|T|})\}$ and that a server $\mathcal{S}$ requests the trajectory information.

3.3.1 Creation of the anonymization group. The process starts when the server $\mathcal{S}$ sends a request to collect trajectory data from users. Then, users who are willing to share their information send a confirmation to the server. Let us denote the total number of users who send a confirmation by N.

The users who want to participate in the process must be arranged into groups of size n, where n is a predetermined system parameter. In order to prevent $\mathcal{S}$ from grouping users as it wishes, a joint coin-tossing protocol adapted from (Lindell and Waisbard 2010) is executed. Note that, apart from preventing a server from dishonestly creating the groups, it also prevents user collusion. The coin-tossing protocol compels the group creation to be pseudo-random. Consequently, a group of malicious users trying to be grouped with a target user have low probability of succeeding. The probability of achieving a bad grouping is analyzed in Section 4.

The group creation protocol assumes that every user U_i already has a personal public key (pk_i) provided by a PKI. The protocol employs two random oracles $H_1 = 0,1^* \rightarrow 0,1^{\Upsilon}$ (where Υ is the bit-length of the public key),
and $H_2 = 0,1^* \rightarrow 0,1^{N \log N}$. The following steps are executed:

(1) Every user U_i generates a random $rand_i$ and uses her IP address (IP_i) to calculate and send $H_1(IP_i, pk_i, rand_i)$ to $\mathcal{S}$.

(2) U_i waits for a short predefined time.
(3) $\mathcal{S}$ sends $H_1(IP_i, pk_i, rand_i)$ for $i = 1, \ldots, N$ to all the users.
(4) Then, each user U_i computes $h = H_2(H_1(IP_i, pk_i, rand_i), \ldots, H_1(IP_N, pk_N, rand_N))$ and divides the result h into chunks of size $\log N$, denoted as $h_1, \ldots, h_N$.
(5) The user U_i takes h_i as her identifier.
(6) The grouping is carried out by taking groups of n parties according to the sorting. That is, for $i = 1, \ldots, \lfloor N/n \rfloor$, the i-th group is formed by users with identifiers $(h_{n(i-1)+1}, \ldots, h_{ni})$.
(7) $\mathcal{S}$ sends the IP addresses of each user to the members of her group.
(8) The members of each group send each other their IP addresses, public keys and the $rand_i$ they used at the beginning of the protocol.
(9) Each group member computes $H_1(IP_j, pk_j, rand_j)$ for every user U_j in her group, and verifies that it matches what she received from $\mathcal{S}$. Additionally, she computes H_2 as in Step 4 to verify that all the IP addresses assigned to her group are inside it. If any verification fails, she sends *abort* to the group members and exits the system.

3.3.2 Trajectory anonymization. In this phase, U_i decides the granularity of spatio-temporal disclosure for every point of T. This means that, for every point (x_i, y_i, t_i), the user will obtain a 5-tuple $(cx_i, cy_i, r_i, a_i, b_i)$ based on the values that she chooses for these two parameters:

- *The radius r_i.* This parameter, expressed in kilometers, is the radius of the circle that contains the Cartesian coordinates (x_i, y_i). A larger radius means higher generalization and hence, higher distortion. Based on the value chosen by the user, we randomly select a point (cx_i, cy_i) that fulfills the equation:

$$(x_i - cx_i)^2 + (y_i - cy_i)^2 \leq r_i^2.$$

  In order to obtain a center of the circle which reveals no information about (x_i, y_i), we work with a polar coordinate system. We choose random values for the distance and the angle, and then we transform it again to cartesian coordinates.
- *The time gap γ_i.* This parameter, expressed in hours (but working with real numbers), indicates the time difference between a_i and b_i. Therefore, to obtain these values, we randomly choose a value v between 0 and γ_i. Then, we compute:

$$a_i = t_i - v,$$
$$b_i = t_i + \gamma_i - v.$$

Repeating this process for all the points $(x_1, y_1, t_1), (x_2, y_2, t_2), \ldots, (x_{|T|}, y_{|T|}, t_{|T|})$ in T, we obtain the anonymized trajectory:

$$T' = \{(cx_1, cy_1, r_1, a_1, b_1), (cx_2, cy_2, r_2, a_2, b_2), \ldots, (cx_{|T'|}, cy_{|T'|}, r_{|T'|}, a_{|T'|}, b_{|T'|}).\}$$

Note that the user can also completely remove a spatio-temporal point from the list. Therefore, $|T|$ and $|T'|$ might differ.

3.3.3 Sub-trajectory extraction. The sub-trajectory extraction depends on two parameters:

- The maximum number of sub-trajectories to extract (τ), and
- The maximum number of points that each sub-trajectory should contain, μ.

Having τ, μ as system parameters, Algorithm 1 shows how to extract the sub-trajectories. The idea is to generate all the possible sub-trajectories using a binary coded system. For example, for a trajectory with four points a→b→c→d, we would generate all the different binary numbers of four digits (0000, 0001, 0010, etc.). Then, the first position of the binary number corresponds to a, the second to b, and so on. If the digit is a 0, that point does not appear in the sub-trajectory. For example, for the number 1101, we would generate the sub-trajectory a→c→d.

Note that we must consider neither void sub-trajectories (0000) nor sub-trajectories with a single point (e.g., 0100). In order to do this, we assume that our algorithm uses two build-in functions:

- toBinary(integer, digits): this function converts an integer number into a binary number of the specified number of digits. Then, it allocates them into the positions of a table. For example: toBinary(2, 4) would output a table of 4 positions containing |0|0|1|0|.
- count(table): this function counts how many "1"s appear in a table.

3.3.4 Fake sub-trajectory generation. Similarly to the real sub-trajectory extraction, the fake sub-trajectory generation needs the following two parameters:

- The number of fake sub-trajectories to generate (τ'), and
- The maximum number of points that each fake sub-trajectory should contain, μ'.

In the literature, there are many works that describe how to generate a fake trajectory. The generation of a particular algorithm for this step is out the scope of this paper. For our purposes, we employ the method proposed in (Gkoulalas-Divanis and Verykios 2008).

3.3.5 Distribution of sub-trajectories. The objective of this phase is to distribute the real and fake sub-trajectories among the group of users $\{U_1, \ldots, U_n\}$ created in Section 3.3.1. At the beginning of this step, each user has a list of real and fake sub-trajectories that she has generated. The goal of this step is to allow users to securely shuffle and exchange them. At the end of this step, each user has a list of real and fake sub-trajectories (probably most of them not generated by her), that she must submit to the server.

In order to prevent one malicious member of the group from learning all the sub-trajectories that belong to another user, the group executes a multi-party privacy-preserving protocol composed by three phases: group key generation, anonymous sub-trajectory retrieval, and sub-trajectory submission.

In the group key generation, the members of the group create one common encrypting public key . The main feature of this process is that all users must collaborate in order to decrypt a message. No user alone has enough information to generate the secret key and decrypt.

During the anonymous sub-trajectory retrieval, each user performs three operations:

(1) Partial decryption. The user partially decrypts the list of sub-trajectories. This means that the list is no longer encrypted with her share of the group key
(2) Re-masking. The re-masking operation changes the appearance of the ciphertexts in order to prevent an attacker from linking a ciphertext before and after the partial decryption.
(3) Permutation. The ciphertexts are re-ordered at random inside the list before they are sent to the next user, again to prevent an attacker from linking them to their previous version.

When a user completes these three operations, forwards the list of ciphertexts to the next user. When partially decrypting, the last user obtains the cleartexts, and she broadcasts them to the rest of users.

Algorithm 1 Sub-trajectory extraction algorithm

function SUB-TRAJECTORY EXTRACTION
Input: τ, μ, anonymized trajectory T' as a list (or vector) of uncertain spatio-temporal points
Output: List *subtraj* of anonymized sub-trajectories
Uses:
function toBinary(n, m) // This function returns a vector of bits corresponding to the binary form of number n, with the required number of heading '0's to complete m bits.

function count(*binVector*) // This function returns the number of '1' bits in the binary vector *binVector*.

function append(*list*,*element*) // This function returns a new list that contains the elements of *list* with *element* attached as the trailing member of the new list.

subtraj←new (empty) list of lists of uncertain spatio-temporal points
curtraj←new (empty) list of uncertain spatio-temporal points
$[subtrsize] \leftarrow 0$
binaryVector ←new table[$|T'|$] of bits

Loop: $i \leftarrow 1$ to $(2^{|T'|} - 1)$ by 1
  binaryVector←toBinary($i, |T'|$)
  If: count(*binaryTable*)> 1 and count(*binaryVector*)≤ μ then
    Loop: $j \leftarrow 0$ to $|T'| - 1$ by 1
      If: $binaryVector[j] = 1$ then
        curtraj←append(*curtraj*,$T'[j]$)
      If-end
    Loop-end: j
    subtraj←append(*subtraj*,*curtraj*)
    subtrsize←*subtrsize*+1
  If-end
Loop-end: i
While (*subtrsize* > τ)
  Remove one random element from *subtraj*
  subtrsize←*subtrsize*−1
While-end
end function

Finally, in the sub-trajectory submission, the users are assigned a portion of the list of sub-trajectories in cleartext. They must submit them to the server, and broadcast the response.

- Group key generation
  (1) In order to generate the group key, each user U_i performs the following steps:
    (a) Generate a random number $a_i \in \mathbb{Z}_q^*$.
    (b) Calculate her own share $y_i = g^{a_i} \bmod p$.
    (c) Broadcast a commitment to her share $h_i = \mathcal{H}(y_i)$, where $\mathcal{H}$ is a one-way function. As analyzed below, this will ensure that no user alters the key generation process.
    (d) Broadcast y_i to the other members of the group.
    (e) Check that $h_j = \mathcal{H}(y_j)$ for $j = 1, \ldots, n$.

(f) Calculate the group key using the received shares:

$$y = \prod_{1 \leq j \leq n} y_j = g^{a_1} \cdot g^{a_2} \cdot \ldots \cdot g^{a_n}.$$

- Anonymous sub-trajectory retrieval Assuming that each user U_i has $(\tau+\tau')$ sub-trajectories: $s_{i1}, s_{i2}, \ldots, s_{i(\tau+\tau')}$:
  (1) The user U_i encrypts every real and fake sub-trajectory. For each s_{ij}:
    (a) U_i generates a random number Φ_{ij}.
    (b) U_i encrypts s_{ij} with the group key y:

$$c_{ij}^0 = E_y(s_{ij}, \Phi_{ij}) = (g^{\Phi_{ij}}, s_{ij} \cdot y^{\Phi_{ij}}) = (c1_{ij}, c2_{ij}).$$

  (2) For $i = 2, \ldots, n$, $j = 1, \ldots, \tau + \tau'$, each user U_i sends c_{ij}^0 to the first member of the group (U_1).
  (3) For $i = 1, \ldots, n-1$, each user U_i performs the following operations:
    (a) Receive the list of ciphertexts $\left\{c_{11}^{i-1}, c_{12}^{i-1}, \ldots, c_{n(\tau+\tau')}^{i-1}\right\}$.
    (b) Using her share of the group key, partially decrypt the list of ciphertexts using the algorithm described in Section 2.4. The resulting list of ciphertexts is denoted as $\left\{\hat{c}_{11}^{i-1}, \ldots, \hat{c}_{n(\tau+\tau')}^{i-1}\right\}$.
    (c) The list of ciphertexts $\left\{\hat{c}_{11}^{i-1}, \ldots, \hat{c}_{n(\tau+\tau')}^{i-1}\right\}$ is re-masked using the re-masking algorithm described in Section 2.4.2 with a key $y' = \prod_{w=i+1}^{n} g^{\alpha_w}$. As a result, U_i obtains a re-encrypted version $\left\{e_{11}^{i-1}, \ldots, e_{n(\tau+\tau')}^{i-1}\right\}$.
    (d) Permute the order of the ciphertexts at random, obtaining a reordered version $\left\{e_{\sigma(11)}^{i-1}, \ldots, e_{\sigma(n(\tau+\tau'))}^{i-1}\right\}$.
    (e) Send the list of ciphertexts $\left\{c_{11}^{i}, \ldots, c_{n(\tau+\tau')}^{i}\right\} = \left\{e_{\sigma(11)}^{i-1}, \ldots, e_{\sigma(n(\tau+\tau'))}^{i-1}\right\}$ to U_{i+1}.
  (4) The last user U_n performs the following operations:
    (a) Receive the list of ciphertexts $\left\{c_{11}^{i-1}, \ldots, c_{n(\tau+\tau')}^{i-1}\right\}$.
    (b) Using her share of the group key, partially decrypt the list of ciphertexts using the algorithm described in Section 2.4. At this point, U_n owns the cleartexts of the sub-trajectories.
    (c) Broadcast the list of sub-trajectories to the rest of users $\{U_1, \ldots, U_{n-1}\}$.
- Sub-trajectory submission and retrieval
  (1) Each group member U_i must send $\tau + \tau'$ sub-trajectories to the server $\mathcal{S}$. More specifically, from the received list, the user U_i submits the sub-trajectories found between positions i and $i + \tau + \tau' - 1$.
  (2) Upon receiving the $\tau + \tau'$ answers from the server, each user broadcasts them to the rest of the group members.
  (3) Each user takes the answers that correspond to her original sub-trajectories from all the received answers.
  (4) The answer of the server for each sub-trajectory is ϕ, the number of sub-trajectories in the database similar to the one submitted according to Definition 2.5. Sub-trajectories where $\phi < k$ must be removed from the general anonymized trajectory, and hence, they are put in a list L to be used in the next step.

3.3.6 Anonymized trajectory trimming. Using Algorithm 2, the list L of real sub-trajectories is removed from the anonymized trajectory T' of each user.

The resulting anonymized trajectory T' is sent to the server $\mathcal{S}$. The server can store it in its database for future analysis or publication.

Algorithm 2 Anonymized trajectory trimming algorithm

function ANONYMIZED TRAJECTORY TRIMMING
Input: table of sub-trajectories to be removed L, anonymized trajectory T'
Output: Resulting anonymized trajectory T'
$ls \leftarrow$size of L
Loop: $i \leftarrow 0$ to $(ls - 1)$ by 1
  For every spatio-temporal vector q in L_i
    Remove q from T'
  Loop-end: i
end function

3.3.7 Sub-trajectory identifier retrieval. As described in Section 3.1, in scenarios in which an external entity needs to retrieve trajectories from a certain user, we use a legal trusted authority $\mathcal{A}$. The steps for this system to work are the following:

(1) For each resulting anonymized trajectory T' sent to the server, the server replies with an identifier $id_{T'}$.
(2) The identifier is encrypted using the public key of $\mathcal{A}$, and it is sent together with the user identifier $(E_{y_A}(id_{T'}, r), U_i)$ to $\mathcal{A}$.
(3) Upon receiving an official request from an entity, $\mathcal{A}$ can reveal the trajectory identifiers which belong to a particular user U_i. Then, the entity can contact $\mathcal{S}$ to retrieve the anonymized trajectories associated to those identifiers.

3.4 Example of use

Let us illustrate how the protocol works with an example of use from a user's perspective. Let us consider a user Alice, who owns a smartphone capable of recording locations at predefined intervals of time and defines a k-anonymization parameter of $k = 3$.

(1) *Creation of the anonymization group.* The server contacts Alice requesting her trajectory information, and she sends a confirmation back. Then, the server sends the IP addresses of two other users (let us denote them as Bob and Charlie). If, after they execute Steps 1-9 of Section 3.3.1, Alice confirms that the grouping was random, she continues the protocol. Otherwise, she disconnects.
(2) *Trajectory anonymization.* Let us assume, without loss of generality, that Alice has a small trajectory of four points:

$$T = (39.975275, 116.3300116, 2018/01/30\ 14{:}36)$$
$$\downarrow$$
$$(39.8556766, 116.4171266, 2018/01/31\ 7{:}30)$$
$$\downarrow$$
$$(39.9590916, 116.3060849, 2018/02/01\ 12{:}27)$$
$$\downarrow$$
$$(39.881218, 116.292929, 2018/02/00\ 11{:}00)$$

For the first point, Alice is not very concerned about the uniqueness, so she chooses an anonymization radius of 100 meters and a time interval of 30 minutes. Automatically, the system computes a cylinder that contains the exact location. The center of the circle is in 39.976310, 116.330316 and the new time interval is

between 2018/01/30 14:18 and 2018/01/30 14:38. For the second and third points she wants to increase her privacy, so she chooses a radius of 700 meters and a time interval of 180 minutes. Finally, the last point is a location that she visited but she does not want to share, so she completely removes it from the trajectory. The resulting anonymized trajectory T' is:

$$T' = (39.976310, 116.330316, 0.1, 2018/01/30\ 14{:}18, 2018/01/30\ 14{:}38)$$
$$\downarrow$$
$$(39.857739, 116.415657, 0.7, 2018/01/31\ 5{:}47, 2018/01/31\ 8{:}47)$$
$$\downarrow$$
$$(39.962677, 116.299905, 0.7, 2018/02/01\ 12{:}22, 2018/02/01\ 15{:}22)$$

(3) *Sub-trajectory extraction.* Let us call A_1 the first point of the anonymized trajectory, A_2 the second one and A_3 the third one. Let us also assume that the system parameters are $\tau = 3$ (the maximum number of sub-trajectories to extract) and $\mu = 2$ (the maximum number of points that each sub-trajectory should contain). Then, Alice would obtain sub-trajectories $A_1 \rightarrow A_2$, $A_2 \rightarrow A_3$, $A_1 \rightarrow A_3$.

(4) *Fake sub-trajectory generation.* The way to generate fake sub-trajectories is out of the scope of this work, so in this step we assume that Alice generates one fake-subtrajectory that we call $A_4 \rightarrow A_5$. So, all the sub-trajectories of Alice are: $A_1 \rightarrow A_2$, $A_2 \rightarrow A_3$, $A_1 \rightarrow A_3$, $A_4 \rightarrow A_5$.

(5) *Distribution of sub-trajectories.* So far, Alice has executed the protocol on her own. At this point, Alice is about to exchange her sub-trajectories with Bob and Charlie, following the steps detailed below:

- Group key generation: First of all, Alice generates a random number $a_{\text{alice}} \in \mathbb{Z}_q^*$ and calculates her share of the public key $y_{\text{alice}} = g^{\text{alice}} \bmod p$. In order to prove Bob and Charlie that she is honest and she will not later modify her share, Alice sends a commitment (hash) of y_{alice} and waits until she receives the commitments of her two partners. Then, each of them broadcasts the share of the public key and checks that what they receive matches the previous commitments. If everything has been honestly executed, Alice calculates the group key $y = y_{\text{alice}} \cdot y_{\text{bob}} \cdot y_{\text{charlie}}$
- Encryption: Alice encrypts her four sub-trajectories $A_1 \rightarrow A_2$, $A_2 \rightarrow A_3$, $A_1 \rightarrow A_3$, $A_4 \rightarrow A_5$ using the public group key y. Let us denote the resulting four ciphertexts as $c_{\text{alice}_1}, c_{\text{alice}_2}, c_{\text{alice}_3}, c_{\text{alice}_4}$.
- Ciphertext collection: Let us assume that the server randomly designated Alice as the first user of the group, Bob is the second and Charlie the third. Then, at this point, Alice receives Bob's and Charlie's ciphertexts. Thus, for example, now she has a list of 12 ciphertexts, namely c_{alice_1}, c_{alice_2}, c_{alice_3}, c_{alice_4}, c_{bob_1}, c_{bob_2}, c_{bob_3}, c_{bob_4}, c_{charlie_1}, c_{charlie_2}, c_{charlie_3}, c_{charlie_4}.
- Partial decryption: Alice uses her secret key in order to partially decrypt the 12 ciphertexts. As a result, $\hat{c}_{\text{alice}_1}, \hat{c}_{\text{alice}_2}, \hat{c}_{\text{alice}_3}, \hat{c}_{\text{alice}_4}, \hat{c}_{\text{bob}_1}, \hat{c}_{\text{bob}_2}, \hat{c}_{\text{bob}_3}, \hat{c}_{\text{bob}_4}, \hat{c}_{\text{charlie}_1}, \hat{c}_{\text{charlie}_2}, \hat{c}_{\text{charlie}_3}, \hat{c}_{\text{charlie}_4}$ are no longer encrypted with Alice's share of the group key.
- Re-masking: In order to change the appearance of the ciphertexts, Alice re-masks the 12 ciphertexts from the previous step, obtaining $e_{\text{alice}_1}, e_{\text{alice}_2}, e_{\text{alice}_3}, e_{\text{alice}_4}, e_{\text{bob}_1}, e_{\text{bob}_2}, e_{\text{bob}_3}, e_{\text{bob}_4}, e_{\text{charlie}_1}, e_{\text{charlie}_2}, e_{\text{charlie}_3}, e_{\text{charlie}_4}$.
- Permutation: Now, Alice shuffles the ciphertexts, changing their order at random, and obtaining for example $e_{\text{alice}_4}, e_{\text{bob}_2}, e_{\text{charlie}_1}, e_{\text{alice}_1}, e_{\text{bob}_3}, e_{\text{alice}_3}, e_{\text{charlie}_2}, e_{\text{charlie}_3}, e_{\text{bob}_4}, e_{\text{bob}_1}, e_{\text{alice}_2}, e_{\text{charlie}_4}$.
- List forwarding: Alice sends the resulting list to Bob, who will also perform the partial decryption, the re-masking and the permutation on it. Then, Bob will forward the results to Charlie, who will partially

decrypt them and, consequently, will obtain the cleartexts in a random order. The results, are broadcast and now Alice, Bob and Charlie have them.

- *Sub-trajectory submission and retrieval.* Alice submits the first four ($\tau + \tau'$) elements of the random list to the server, and obtains four responses containing ϕ for each sub-trajectory. She broadcasts them and obtains the four responses from Bob and the four responses from Charlie.

(6) *Anonymized trajectory trimming.* Alice discards the ϕ results that correspond to Bobs' and Charlies' sub-trajectories, and she also discards the results for her fake sub-trajectory. Then, she knows how many times (ϕ) each of her real sub-trajectories appears in the server database. Since Alice defined $k = 3$, if, for example, $A_2 \rightarrow A_3$ has $\phi = 1$, she will remove it from T'. Then, Alice will only send $T' = A_1 \rightarrow A_2$ to the server.

4 PRIVACY AND UTILITY ANALYSIS

In this section, the privacy of the system in the presence of dishonest entities is analyzed. Two kind of entities might behave dishonestly in our protocol: a user inside the group or the server. The legal authority $\mathcal{A}$ only participates in certain scenarios, it contains no trajectory information, and it is always considered as a trusted third party. In addition, the utility of the anonymized trajectory data is discussed.

4.1 Privacy analysis

The privacy of the proposed system is discussed in this section.

4.1.1 Dishonest user. The ElGamal cryptosystem is semantically secure under the Decisional Diffie-Hellman assumption. This means that a dishonest user cannot know if two different ciphertexts will result into the same cleartext after decryption.

Therefore, every time that a ciphertext c_i is transformed by a group member (i.e., remasked and permuted), the attacker can only link the result to c_i by random guessing, the intermediate re-maskings and permutations preventing her from finding the links between them. Hence, the probability of success is $1/(n(\tau + \tau'))$, since there are $n(\tau + \tau')$ ciphertexts involved in the process.

Let us consider the case in which a dishonest user successfully learns the message of another component of the group. This means that she is able to link one input of the permutation networks with one of the outputs. This attack may be conducted if the dishonest user knows the secret group key. In this case, the attacker can decrypt the queries at any step of the protocol.

Theorem 4.1. *The attacker can only recover the secret key if:*
(a) she compromises the other $n - 1$ members of the group, or
(b) she can find collisions in a cryptographic hash function.

Proof. Let us assume that we have an adversary that is able to recover secrets key in our system. The generation of the group key is distributed among the participants using the n-out-of-n threshold ElGamal key generation detailed in Section 2.4. The public key is computed as:

$$g^{\alpha} = g^{\alpha_1} \cdot \ldots \cdot g^{\alpha_n}$$

Hence, the secret key is $\alpha = \alpha_i + \ldots + \alpha_n$. This means that the adversary either knows or can compute the left part (α) of the equation or the terms of the right part ($\alpha_i, \ldots, \alpha_n$). In order to obtain α, the adversary must be able to

compute discrete logarithms. Currently, this is assumed to be computationally hard and therefore, not feasible for any adversary.

On the other hand, for the adversary to know $\alpha_i, \ldots, \alpha_n$ has already be proven to be unfeasible. One of the characteristics of the n-out-of$n$ threshold ElGamal encryption is that, if there is even a single honest user, the secret key cannot be reconstructed. Hence, the fulfillment of the theorem first condition (a) is inherent to the security of the n-out-of-n ElGamal scheme (Desmedt and Frankel 1990).

Another alternative in order to learn the secret key is to maliciously alter the key generation phase. In this phase, each user generates her share $y_i = g^{a_i}$, then she broadcasts a commitment to that share using a cryptographic hash function $\mathcal{H}(y_i)$ and, finally, she sends y_i in a new message. A dishonest user may change her choice of share after receiving the shares of the other participants, before sending her own. This dishonest user calculates her share $y'_j = g^{a_j} / \prod_{i=1}^{n-1} y_i = g^{a_j - a_1 - \cdots - a_{n-1}}$ and broadcasts it. As a result, the group key is computed as $y = g^{a_j}$ and, hence, the dishonest user knows the secret group key.

In order for this attack to be successful and remain undetected, the dishonest user must be able to find collisions in the hash function. This means that she must find a value y'_j for which her previous commitment is still valid (i.e., $\mathcal{H}(y_i) = \mathcal{H}(y'_i)$). Nowadays, the probability of finding a collision in a reasonable amount of time, using a cryptographic hash function such as SHA-2, is almost negligible. Therefore, the probability for an attacker to fulfill the theorem second condition (b) can be neglected. □

Thus, the probability of attacking the system successfully can be neglected, since it would imply either (1) forming a collusion with the other $n - 1$ users of the group (who are selected pseudo-randomly and, hence, do not know each other), or (2) finding a collision of a cryptographic hash function, which is computationally infeasible.

4.1.2 Dishonest server. The objective of the server is to obtain trajectories to store in its database. However, these requests do not disclose relevant information: when the proposed protocol is executed, the server cannot know if a certain sub-trajectory has been generated by the user who has submitted it. This happens because, when a user U_i executes the protocol, she submits her real sub-trajectories hidden among her fake sub-trajectories and real and fake sub-trajectories generated by other $n - 1$ users.

As remarked above, the objective of the server is to help users achieve k-anonymity. This is done by comparing the new sub-trajectories with already stored sub-trajectories, and letting the user know how unique their data are. Nevertheless, the k-anonymity might differ among users, since the k is a system parameter chosen by each user.

Regarding the bootstrapping, we assume that there is a part of the population who is not concerned about sharing the anonymized trajectory data (as in Figure 1(b)) even if the sub-trajectories are not common to other users ($k = 0$). However, for systems where privacy is more critical, the server is required to have a bootstrapping phase. During this phase, the server will store trajectories, but they will not be associated to any identifier. These "completely anonymous" trajectories have an expiration date, and can be removed whenever the server no longer needs them.

As the creation of the group is concerned, the steps presented in Section 3.3.1, adapted from (Lindell and Waisbard 2010), prevent the server from maliciously grouping users. The security of this protocol is analyzed by (Lindell and Waisbard 2010). The authors compute the probability that a bad grouping occurs, i.e., having $n - 1$ dishonest users together with a single honest party. Assuming that the server has t machines under its control, the probability of a bad grouping is:

$$\prod_{i=1}^{n-2} \frac{(t-i)}{(N-i)} \frac{N-t}{N-n+1} N.$$

Nevertheless, if we assume that $N \gg t$, then we have that this probability is approximately $\left(\frac{t}{N}\right)^{n-2} N$. For example, if one million users participate in the system, and the server controls one thousand, then the probability of a bad grouping is under 10^{-48}.

In the protocol presented in Section 3.3, the server is trusted to respond truthfully to requests. Nevertheless, for the sake of exhaustiveness, let us consider the consequences of a malicious server that provides fake responses. The main interaction with the server occurs in Step 4 of Section 3.3.5. In this step, the server sends a value ϕ that indicates how many times a similar trajectory appears in the database. If this value is lower than the user-chosen value k, then the trajectory is not shared. However, if a malicious server always sends very high values for ϕ, it is likely that users will share all their trajectories. In this case, their privacy is not protected. A possible solution is to use some anonymous signature scheme (Yang et al. 2006) that forces the server to prove that its response is real.

Nevertheless, using signatures consumes a large amount of time. For this reason, another solution that balances performance and security is to add redundancy to the servers. If several servers hold the same database, then users can send requests to more than one. The responses must be the same, otherwise, the user will no longer send her data. This solution is not completely secure, but it provides a certain probability of success if at least one server is honest.

4.2 Utility analysis

Enhancing privacy in trajectory data entails some modifications of the original data, which might decrease the corresponding utility. Generally, there is a trade-off between the level of privacy and the utility of the resulting data. Therefore, in order to evaluate our system, it is necessary not only to analyze privacy, but also to quantify how much utility has been reduced.

Measuring data utility is a tough task. Currently, no single utility measure is broadly accepted (Bertino et al. 2008). Consequently, in this section, we propose a utility measure adapted to our scenario. For each spatio-temporal point i, our system allows users to anonymize it by adjusting two parameters:

- *The radius* r_i*:* the radius of the circle that contains the Cartesian coordinates (x_i, y_i).
- *The time gap* γ_i*:* the time difference between a_i and b_i.

Therefore, the utility of resulting data directly depends on the choice the above parameters. This means that, in our system, the utility of the trajectory data will not be the same for every user in the final database, and that there is no upper bound for utility loss.

In order to calculate the utility of anonymized trajectory data belonging to two different users, we propose Equation 1. Informally, this expression calculates the inverse function of the volume of each *uncertain point* as represented in Figure 1(b), weighted by the number of points that have been removed from the original trajectory. If all the points in the original trajectory appear in the anonymized trajectory, then the utility is the inverse function of the sum of volumes of each uncertain point. If some points of the original trajectory are completely removed, then utility is penalized and consequently reduced. An exponential function is used in order to bound the results between 0 and 1.

$$\text{Utility} = 1 - \exp\left(-\frac{1}{\sum_{i=1}^{|T'|} \pi r_i^2 \gamma_i} \frac{|T'|}{|T|}\right). \tag{1}$$

Let us illustrate how this formula works with a simple example. We consider a user U_i with the following trajectory:

$$(59.9145, 10.7532, 2015/03/01\ 14{:}03)$$
$$\downarrow$$
$$(59.9045, 10.7722, 2015/03/15\ 7{:}30)$$
$$\downarrow$$
$$(60.2067, 11.1134, 2015/04/01\ 17{:}17)$$
$$\downarrow$$
$$(50.7345, 7.1134, 2015/04/15\ 20{:}41)$$
$$\downarrow$$
$$(50.0966, 8.6235, 2015/05/01\ 1{:}22).$$

Now, we consider that, after anonymizing the trajectory and removing the sub-trajectory (based on the server response) corresponding to $(59.9045,\ 10.7722,\ 2015/03/15\ \ 7{:}30) \rightarrow (60.2067, 11.1134, 2015/04/01\ \ 17{:}17)$, the final anonymized trajectory is the following:

$$(59.9160, 10.7569, 0.3, 2015/03/01\ 13{:}08, 2015/03/01\ 14{:}12)$$
$$\downarrow$$
$$(50.7301, 7.1186, 0.8, 2015/04/15\ 20{:}19, 2015/04/15\ 21{:}15)$$
$$\downarrow$$
$$(50.1001, 8.7763, 0.4, 2015/05/01\ 13{:}13, 2015/05/01\ 14{:}55).$$

Then, using the utility expression from Equation 1, we obtain:

$$\text{Utility} = 1 - \exp\left(-\frac{1}{(\pi \cdot 0.3^2 \cdot 1.06) + (\pi \cdot 0.8^2 \cdot 0.93) + (\pi \cdot 0.4^2 \cdot 1.7)}\frac{3}{5}\right) = 0.18.$$

For a different parameter selection, the value may increase or decrease. For example, if we use smaller radius and time gaps than for the previous anonymized trajectory, we may obtain the following anonymized trajectory:

$$(59.9130, 10.7369, 0.1, 2015/03/01\ 13{:}48, 2015/03/01\ 14{:}17)$$
$$\downarrow$$
$$(50.7371, 7.1122, 0.3, 2015/04/15\ 20{:}25, 2015/04/15\ 20{:}53)$$
$$\downarrow$$
$$(50.1000, 8.6263, 0.2, 2015/05/01\ 13{:}09, 2015/05/01\ 13{:}38),$$

whose corresponding utility can be computed as follows:

$$\text{Utility} = 1 - \exp\left(-\frac{1}{(\pi \cdot 0.1^2 \cdot 0.48) + (\pi \cdot 0.3^2 \cdot 0.46) + (\pi \cdot 0.2^2 \cdot 0.48)} \cdot \frac{3}{5}\right) = 0.94.$$

The utility is considerably higher, since the radius of the three points are shorter and the time gaps have also decreased. This utility measure can be used by data collectors in order to filter data that do not reach a utility threshold. Alternatively, it can be used to calculate the average utility of a complete dataset.

5 SIMULATIONS WITH REAL DATA

In order to test the proposed system, we have implemented the protocol and we present results regarding trajectory anonymization in this section. More specifically, the results show how many anonymized trajectories are submitted to the server for different parameter configurations.

For this purpose, we have used real data extracted from GeoLife GPS trajectories (Zheng et al. 2009). This GPS trajectory dataset contains sequences of time-stamped points collected from 182 users, employing GPS loggers and GPS-phones, in a period of over five years (from April 2007 to August 2012). The dataset contains 17,621 trajectories with a total distance of about 1.2 million kilometers and a total duration of 48,000+ hours.

The proposed protocol requires two components: the central server and the user application. These two components have been implemented using the Java programming language in order to allow application portability.

The server $\mathcal{S}$ is a process (daemon) that listens to user requests in a fixed TCP port. After receiving n requests, $\mathcal{S}$ creates a new group and sends a message to each member with the IP addresses and ports to be used. It also sends a large prime p and $g \in Z_p^*$.

5.1 Testing methodology

The proposed protocol has several parameters that need to be configured for the simulations:

- **The number of simulations**. In order to avoid showing results based on an accident or coincidence, all the simulations have been performed three times for every parameter configuration. Consequently, all the presented results are in fact the arithmetic mean of three measurements.
- **The radius r_i and the time gap γ_i to anonymize each trajectory point**. Normally, users should choose these parameters for each point of their trajectories. However, since our system has not been deployed yet, for the simulations, we are forced to generate them at random. More specifically, for each point, we generate a random value for the radius between 0 and 1 km, and a random value for the time gap between 0 and 60 minutes. Since the point anonymization is random, computing utility as described in Section 4.2 is pointless. Nevertheless, in the future, we intend to deploy our system in a real environment and obtain values for r_i and γ_i from real users.
- **The number of sub-trajectories to extract (τ) and maximum number of points in each sub-trajectory (μ)**. For the sake of exhaustiveness, we have decided to extract, for each trajectory, all existing minimal sub-trajectories. This means that we consider all existing two-point ordered combinations of spatio-temporal points. For example, for a trajectory $A \rightarrow B \rightarrow C \rightarrow D$, we extract the sub-trajectories: $A \rightarrow B$, $A \rightarrow C$, $A \rightarrow D$, $B \rightarrow C$, $B \rightarrow D$, $C \rightarrow D$.

  As a result, from the 17,621 trajectories in the GeoLife dataset, we obtain 242,581 sub-trajectories associated to 182 different users.
- **The maximum distance to consider two points similar in terms of location (θ_L) and time (θ_T)**. We consider that two sub-trajectories are similar if both points are closer in distance and time than θ_L and θ_T, respectively. In order to test the effect of being more or less restrictive with these parameters, we have carried out experiments with the following two configurations:
  (1) $\theta_L = 1$ km, $\theta_T = 60$ minutes.
  (2) $\theta_L = 5$ km, $\theta_T = 180$ minutes.
- **k value for the k-anonymization**. With a sample of only 182 users, we have decided to simulate the system for its minimum value, $k = 2$. This means that a user will submit a sub-trajectory only if it already appears in the database once. This is the minimum value that provides uncertainty about whether that trajectory belongs to that user. All the simulations have been performed with $k = 2$ except for those appearing on Section 5.2.1, that show how increasing k affects the results.

- **Bootstrapping**. We call bootstrapping to the first phase of the simulations necessary to fill the database with some initial trajectories. We require a group of users, the bootstrappers, that are assumed to openly share their trajectories, i.e., they voluntarily renounce their anonymity ($k = 0$). Bootstrappers are necessary to let other users obtain k-anonymity for $k > 1$. For our simulations, we assume three different values for the quantity of bootstrappers: 10, 40, and 80 users. However, none of the results observed for bootstrappers are used to calculate the average results shown in this paper (e.g., the ratio of submitted anonymized sub-trajectories).

5.2 Results

Table 1 shows the percentage of sub-trajectories that were submitted to the database for each parameter configuration. Results show a low ratio of submitted anonymized sub-trajectories. This is, in part, due to the reduced number of available data, and it means that users inside the dataset do not share many common locations. However, we can observe that the percentage of submissions increases with the number of bootstrappers and with less restrictive θ_T and θ_L.

Table 1. Ratio of submitted sub-trajectories for different configurations

	Percentage of submitted sub-trajectories
bootstrap: 10 $\theta_L = 1$ km $\theta_T = 60$ min	1.89%
bootstrap: 40 $\theta_L = 1$ km $\theta_T = 60$ min	2.64%
bootstrap: 80 $\theta_L = 1$ km $\theta_T = 60$ min	5.11%
bootstrap: 80 $\theta_L = 5$ km $\theta_T = 180$ min	7.12%

Table 2 shows the number of users who have at least submitted one sub-trajectory to the database and, for each of them, the percentage of sub-trajectories that they have submitted.

Results are different depending on how restrictive the parameter configuration is. In the first configuration, only 29% of the users submit at least one sub-trajectory, and the total percentage of their submitted sub-trajectories is 2.22%. However, these results increase when the parameter configuration is less restrictive. In the least restrictive configuration, nearly 60% of the users submitted at least one sub-trajectory and the total percentage of their submitted sub-trajectories is 7.21%.

5.2.1 Increasing the value of k. For the sake of exhaustiveness, we have performed simulations with higher values for the k parameter of k-anonymity. The objective of these results is to allow a comparison with those shown in Tables 1 and 2. Without loss of generality, we have decided to use the parameter configuration which provides better results, i.e., bootstrap: 80, $\theta_L = 5$ km and $\theta_T = 180$ minutes.

The results are shown in Table 3, for $k = 3$ and $k = 4$. The first column is the percentage of submitted sub-trajectories, which is to be compared to the 7.12% of submitted sub-trajectories of Table 1.

Table 2. Information about users that have submitted at least one sub-trajectory

	Number of users that have submitted at least one sub-trajectory	Average percentage of submitted sub-trajectories for users that have submitted at least one
bootstrap: 10 θ_L = 1 km θ_T = 60 min	51 out of 172	3.22%
bootstrap: 40 θ_L = 1 km θ_T = 60 min	53 out of 142	5.95%
bootstrap: 80 θ_L = 1 km θ_T = 60 min	61 out of 102	7.08%
bootstrap: 80 θ_L = 5 km θ_T = 180 min	68 out of 102	10.84%

Table 3. Results comparison for $k = 2$, $k = 3$ and $k = 4$

	Percentage of submitted sub-trajectories	Number of users who have submitted at least one sub-trajectory	Average percentage of submitted sub-trajectories for users that have submitted at least one
$k = 2$	7.12%	68 out of 102	10.84%
$k = 3$	5.85%	58 out of 102	8.81%
$k = 4$	3.94%	56 out of 102	6.23%

As expected, the results given in Table 3 show that higher values of k result into a smaller number of submissions. Restricting the value of this parameter decreases the percentage of submissions from 7.12%, to 5.85% ($k = 3$), and to 3.94% ($k = 4$). The number of users that submit at least one sub-trajectory is also reduced, as does the percentage of submitted sub-trajectories for them.

Having less submissions with a higher k is not unexpected, since this is a more restrictive parameter configuration that can be satisfied less frequently. Nevertheless, in order to provide a more thorough analysis, we have studied the response ϕ provided by the server: the number of sub-trajectories in the database that are similar to the one queried by the user. Sub-trajectories with $\phi < k$ are not finally submitted to the server by its original user. For the previous simulations with $k = 3$, we have observed that the server returned $\phi = 0$ in 86.76% of the cases. This means that for most sub-trajectories, no similar sub-trajectory was found. Then, we have observed that, for those queries that received $\phi > 0$, the average number of similar sub-trajectories in the database was 3.4. Consequently, we can infer that, for those sub-trajectories which are not unique, they appear more than three times in the database, on average.

These results indicate that for larger values for k , the number of submissions will linearly decrease. The main reason for such a decrease is that the GeoLife database is not representative enough of a real population. With only 182 different users, requesting that at least $k = 3$ of the 182 users share the same sub-trajectory is the same as requesting that 1.64% of the population share it. For $k = 4$, a 2.2% of the population should share the same sub-trajectory in order to be submitted. For future work, we plan to work with larger databases, most likely extracte from social network location-aware posts, which will allow to increase the value of k and have more representative results for a real population.

5.2.2 Performance. In our scenario, it is difficult to provide relevant results regarding performance (e.g., anonymization cost, computation and communication delay). First of all, users can share their trajectories from different devices (smartphones, tablets, GPS systems, etc.) which have different power and effectiveness. Moreover, as stated in the Introduction (Section 1), the purpose of our system is to allow past trajectory data collection from a server. Users are not sending data from the device at a certain frequency. Instead, a server sends a request (e.g., to create a database for a data mining process of traffic analysis), and when the user is idle or has low task consumption, she executes the protocol.

Nevertheless, we can analyze the performance of our protocol from a theoretical point of view. In order to do so, we have studied the two elements that may have a higher impact on the performance: the communication and the computation costs.

First of all, we have measured the communication cost based on the message complexity. In order words, we have counted how many messages a user must send during the protocol. Note that we consider sending a long message (for example a list of ciphertexts) as only one message. We do not deal with the fact that the underlying routing protocol might have to split them into several chunks. On the other hand, we consider a broadcast of one message to n users as n different messages. We do not assume a flooding system that might reduce this cost. There are three steps of the protocol in which the user sends messages:

- *Creation of the anonymization group*: The user sends one message to the server containing a hash of her IP address, public key and a random number. Then, she broadcasts the information contained in that hash to the rest of the group. Therefore, the number of messages is computed as $1 + (n - 1)$, and the results is that each user sends n messages in this step of the protocol.
- *Distribution of sub-trajectories*: This is the multi-party step of the protocol and, therefore, the most expensive one in terms of communication. First of all, for the group key generation, each user must broadcast a commitment to her share, and then her share, therefore $2(n - 1)$ messages. Then, all the users have to send their encrypted sub-trajectories to the first user, which means one more message for every user (except for the first one). After that, each user must perform some operations and forward the resulting list of ciphertexts to the next user. This adds one more message to each of them except for the last one, who has to broadcast the list of cleartexts and therefore, she sends $n - 1$ messages. At this point, all of them have to send one message with the list of sub-trajectories to the server, and then broadcast the results to the other members the group ($n - 1$ messages). So, for this step of the protocol, each user sends $2(n - 1) + 1 + 1 + 1 + (n - 1) = 3n$ messages, except for the first user who sends $3n - 1$ messages, and the last user who sends $4n - 2$ messages.
- *Anonymized trajectory trimming*: At the end of this step, each users sends her anonymized trajectory to the server in one message.

Consequently, computing all the steps together, each user sends $4n + 1$ messages. There are two exceptions: the first user sends one message less ($4n$), and the last user sends $5n - 1$ messages. In a scenario with groups of $n = 3$ users, the first user would send 12 messages, the last user would send 14 messages, and the second user 13 messages.

Regarding the computation cost, in our protocol this is inherent to cryptographic operations, specially modular exponentiations. Analyzing the key generation, encryption and decryption of the protocol presented in Section 3.3, the total number of exponentiations is:

$$1 + 2(\tau + \tau') + n(\tau + \tau').$$

The work presented in (Canard et al. 2012) analyzes the cost of executing a cryptographic protocol on a restricted device such a smartphone. For example, for a Samsung Galaxy S2 smartphone with a Dual-core Exynos 4210 1.2 GHz processor ARM Cortex-A9 with the Android OS, v2.3 (Gingerbread), the authors state that an exponentiation takes 42 ms. In order to compare it to a more modern device, we have performed the same test for a OnePlus 5 smartphone with a Qualcomm® Snapdragon™ 835 Octa-core, 10nm, up to 2.45 GHz processor with the OxygenOS based on Android™ Nougat. In this device, an exponentiation takes 0.19 ms.

Consequently, if we consider an illustrative scenario for a $n = 4$ group size, a user who wants to share $\tau + \tau' = 1000$ sub-trajectories, the total cost of the computation, without considering communications costs and other simpler operations costs, would be of 252 seconds approximately (somewhat more than 4 minutes) in the Samsung Galaxy S2 smartphone, and only 1.14 seconds in the OnePlus 5 smartphone.

6 CONCLUSIONS AND FUTURE WORK

Trajectory anonymization has attracted more and more attention over the last few years. Existing solutions in the literature focus on developing privacy-preserving techniques to *publish* trajectories. Nevertheless, we claim that trajectory data would rather be protected in the client end, before they are stored in the server. To the best of our knowledge, this work is the first one to introduce trajectory anonymization in the user's device. The proposed solution gives users control over how much information they send to the server and how accurate it is. By executing a distributed cryptographic protocol, users can decide which parts of the trajectory they share in order to obtain k-anonymity. Additionally, through adjusting a set of parameters, users can distort the time interval and the exact position of any point of their trajectory, and even completely remove the most sensitive points. We have implemented the system and performed simulations with real data from 182 users. The results show how different parameter configurations allow sending more information to the server.

However, there are still some interesting open research problems that need to be addressed in the future. One of the limitations of our work is that it does not consider all the possible background knowledge that an attacker may use. For example, an attacker could use map data or transport schedules in order to reduce the anonymization cylinder, and increase the probabilities of finding the exact location of her victim. Moreover, we are currently working on using social network data in order to improve our system. The idea is to create groups of n users that are close in terms of space and time (information extracted from social network publications) and share their trajectory information together. This will also allow to collect historical social network locations in order to create trajectories, anonymize them together, and put them in the database. With this solution, we can omit the bootstrapping period, for which we assumed that some users would share their trajectories with $k = 0$. Additionally, we intend to deploy our system in a controlled environment with real users. Obtaining data from real users will allow to compute utility and compare it with other existing proposals.

ACKNOWLEDGMENTS

This work was partly funded by the Spanish Government through grant TIN2014-57364-C2-2-R "SMARTGLACIS".

REFERENCES

Abe, M., 1999. Mix-networks on permutation networks. In: in Computer Science, L. N. (Ed.), Advances in Cryptology – Asiacrypt'99. Vol. 1716. pp. 258–273.

Abul, O., Bonchi, F., Nanni, M., 2008. Never walk alone: Uncertainty for anonymity in moving objects databases. In: Proceedings of the 2008 IEEE 24th International Conference on Data Engineering. ICDE '08. IEEE Computer Society, Washington, DC, USA, pp. 376–385.
URL http://dx.doi.org/10.1109/ICDE.2008.4497446

Andrés, M. E., Bordenabe, N. E., Chatzikokolakis, K., Palamidessi, C., 2013. Geo-indistinguishability: Differential privacy for location-based systems. In: Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security. ACM, pp. 901–914.

Beresford, A. R., Stajano, F., 2004. Mix zones: User privacy in location-aware services. In: Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops. PERCOMW '04. IEEE Computer Society, Washington, DC, USA, pp. 127–.
URL http://dl.acm.org/citation.cfm?id=977405.978634

Bertino, E., Lin, D., Jiang, W., 2008. A survey of quantification of privacy preserving data mining algorithms. In: Privacy-preserving data mining. Springer, pp. 183–205.

Canard, S., Desmoulins, N., Devigne, J., Traoré, J., 2012. On the implementation of a pairing-based cryptographic protocol in a constrained device. In: International Conference on Pairing-Based Cryptography. Springer, pp. 210–217.

Chen, R., Fung, B., Desai, B. C., Sossou, N. M., 2012. Differentially private transit data publication: a case study on the montreal transportation system. In: Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, pp. 213–221.

Desmedt, Y., Frankel, Y., 1990. Threshold cryptosystems. In: in Computer Science, L. N. (Ed.), Advances in Cryptology – CRYPTO'89. Vol. 335. pp. 307–315.

Dittler, T., Tschorsch, F., Dietzel, S., Scheuermann, B., Nov 2016. Anotel: Cellular networks with location privacy. In: 2016 IEEE 41st Conference on Local Computer Networks (LCN). pp. 635–638.

Dwork, C., McSherry, F., Nissim, K., Smith, A., 2006. Calibrating noise to sensitivity in private data analysis. In: Theory of Cryptography Conference. Springer, pp. 265–284.

Dwork, C., Naor, M., Pitassi, T., Rothblum, G. N., 2010. Differential privacy under continual observation. In: Proceedings of the forty-second ACM symposium on Theory of computing. ACM, pp. 715–724.

Dwork, C., Naor, M., Reingold, O., Rothblum, G. N., 2014. Pure differential privacy for rectangle queries via private partitions. In: International Conference on the Theory and Application of Cryptology and Information Security. Springer, pp. 735–751.

ElGamal, T., 1985. A public-key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory 31, 469–472.

ElSalamouny, E., Gambs, S., 2016. Differential privacy models for location-based services. Transactions on Data Privacy 9 (1), 15–48.

Gkoulalas-Divanis, A., Verykios, V. S., 2008. A privacy-aware trajectory tracking query engine. ACM SIGKDD Explorations Newsletter 10 (1), 40–49.

Gruteser, M., Liu, X., Mar 2004. Protecting privacy, in continuous location-tracking applications. IEEE Security Privacy 2 (2), 28–34.

He, X., Cormode, G., Machanavajjhala, A., Procopiuc, C. M., Srivastava, D., 2015. Dpt: differentially private trajectory synthesis using hierarchical reference systems. Proceedings of the VLDB Endowment 8 (11), 1154–1165.

Hilbert, D., 1891. Ueber die stetige abbildung einer line auf ein flächenstück. Mathematische Annalen 38 (3), 459–460.

Jiang, K., Shao, D., Bressan, S., Kister, T., Tan, K.-L., 2013. Publishing trajectories with differential privacy guarantees. In: Proceedings of the 25th International Conference on Scientific and Statistical Database Management. ACM, p. 12.

Li, M., Zhu, L., Zhang, Z., Xu, R., 2017. Achieving differential privacy of trajectory data publishing in participatory sensing. Information Sciences 400, 1–13.

Lindell, Y., Waisbard, E., 2010. Private web search with malicious adversaries. In: Privacy Enhancing Technologies. Springer, pp. 220–235.

Mazimpaka, J. D., Timpf, S., 2016. Trajectory data mining: A review of methods and applications. Journal of Spatial Information Science 2016 (13), 61–99.

Pensa, R. G., Monreale, A., Pinelli, F., Pedreschi, D., 2008. Pattern-preserving k-anonymization of sequences and its application to mobility data mining. In: PiLBA. pp. 1–10.

Reumers, S., Liu, F., Janssens, D., Cools, M., Wets, G., 2013. Semantic annotation of global positioning system traces: Activity type inference. Transportation Research Record: Journal of the Transportation Research Board (2383), 35–43.

Romero-Tris, C., Megías, D., 2015. User-centric privacy-preserving collection and analysis of trajectory data. In: Data Privacy Management, and Security Assurance - 10th International Workshop, DPM 2015, and 4th International Workshop, QASA 2015, Vienna, Austria, September 21-22, 2015. Revised Selected Papers. pp. 245–253.
URL http://dx.doi.org/10.1007/978-3-319-29883-2_17

Samarati, P., Sweeney, L., 1998. Generalizing data to provide anonymity when disclosing information (abstract). In: Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems. PODS '98. ACM, New York, NY, USA, pp. 188–.
URL http://doi.acm.org/10.1145/275487.275508

Tanuja, V., Govindarajulu, P., 2016. Application of trajectory data mining techniques in crm using movement based community clustering. International Journal of Computer Science and Network Security (IJCSNS) 16 (11), 20.

Terrovitis, M., Mamoulis, N., 2008. Privacy preservation in the publication of trajectories. In: Mobile Data Management, 2008. MDM'08. 9th International Conference on. IEEE, pp. 65–72.

To, H., Ghinita, G., Shahabi, C., Jun. 2014. A framework for protecting worker location privacy in spatial crowdsourcing. Proc. VLDB Endow. 7 (10), 919–930.
URL http://dx.doi.org/10.14778/2732951.2732966

Yang, G., Wong, D. S., Deng, X., Wang, H., 2006. Anonymous signature schemes. In: International Workshop on Public Key Cryptography. Springer, pp. 347–363.

Yarovoy, R., Bonchi, F., Lakshmanan, L. V. S., Wang, W. H., 2009. Anonymizing moving objects: How to hide a mob in a crowd? In: Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology. EDBT '09. ACM, New York, NY, USA, pp. 72–83. URL http://doi.acm.org/10.1145/1516360.1516370

Zheng, Y., Zhang, L., Xie, X., Ma, W.-Y., 2009. Mining interesting locations and travel sequences from gps trajectories. In: Proceedings of the 18th international conference on World wide web. ACM, pp. 791–800.