On Preserving Sensitive Information of Multiple Aspect Trajectories In-House

Spyros Giotakis

University of Piraeus (UPRC), Greece, sgiotakis@gmail.com

Nikos Pelekis

Department of Statistics and Insurance Science, University of Piraeus (UPRC), Greece, npelekis@unipi.gr

ABSTRACT

Nowadays, human trajectories are enriched with semantic information having multiple aspects, such as by using background geographic information, by user-provided data via location-based social media, as well as by data coming from various kind of sensing devices. This new type of multiple aspects representation of personal movements as sequences of places visited by a person during his/her movement poses even greater privacy violation threats. This paper provides the blueprint of a semantic-aware Moving Object Database (MOD) engine for privacy-aware sharing of such enriched mobility data and introduces an attack prevention mechanism where all potential privacy breaches that may occur when answering a query, are prevented through an auditing methodology. Towards enhancing the user-friendliness of our approach, we propose a mechanism whose objective is to modify the user queries that cannot be answered due to possible privacy violation, to 'similar' queries that can be answered without exposing sensitive information.

CCS CONCEPTS

• Information systems • Information systems applications • Spatial-temporal systems

KEYWORDS

privacy, anonymity, semantic trajectories, multiple aspect trajectories, query auditing

1 Introduction

Nowadays, the vast amount of spatiotemporal 'fingerprints' (i.e. trajectories) of individuals may prove to be a dangerous tool in the hands of a malicious user. So far, the scientific community has proposed various approaches to protect individual's privacy ([1][2][8][10][15]). Most of these studies focus on the spatiotemporal nature of this data, handling them as sequences of points on a geometric space, without considering that such raw data are usually enriched by additional information from the application context. However, the increasing need of analyzing mobility data has led to an advanced representation of trajectories enriched with contextual data from external data sources, thus transforming raw trajectories to the so-called semantic trajectories. A semantically-annotated trajectory, in short *semantic trajectory*, is considered as a sequence of stop episodes (i.e. places where the object remains "static") and move episodes (i.e. parts of the object's trajectory in between two stops) [11]. Each of them may contain additional annotations (i.g. home, cinema, work, etc.). Recently an even more advanced notion of *multiple aspects trajectories* [7] further enriches this kind of data with information coming from any kind of sensing devices (e.g. weather data, measurements from health apps, etc.).

This enriched representation of trajectories may pose even greater privacy violation threats. Consider for example a malevolent user who is able to detect places of interest (POIs), where a moving object has visited (e.g. hospital, betting office, etc.). This additional knowledge allows the inference of personal sensitive information of this specific individual. On the one hand, analyzing semantically-enriched movement traces of users can aid decision making in a wide spectrum of applications, but on the other hand, the disclosure of such data to untrusted parties may expose the privacy of the users, whose movement is recorded. Sharing user mobility data for analysis purposes should be done only after the data has been protected against potential privacy breaches.

The trend in the literature aims at protecting users' privacy by releasing an anonymized version of the original dataset ([1][2][6][8][10][15]). These approaches assume that in the anonymized dataset a malevolent user will not be able to link a specific user with a movement. In this paper, we employ a more conservative approach regarding privacy by assuming that data stay in-house to the hosting organization in order to prevent any privacy breach. We design a query-based auditing mechanism that can effectively identify and block a range of potential attacks that could lead to user identification or tracking, thus controlling the requested information that is released to third parties and ensure privacy-aware data sharing.

In more detail, in this proposal we make the following assumptions to tackle the problem. An enriched trajectory database contains, apart from the 'raw' trajectories, a number of episodes in which these trajectories were 'fragmented'. Every trajectory corresponds to one or more episodes. Each episode consists of data, describing its spatial range, its time range, the 'kind' of episode (i.e. Stop/Move) and a potential number of tags (annotation) which semantically enrich it. The four different pieces of information mentioned above are the data on which every user is able to apply several criteria through his/her query. Every query is a sequence of one or more independent sub-queries and each of them includes at least one of the four pieces of information previously mentioned as criteria. Every user receives as an answer to his query only the number of the trajectories that fulfill the criteria of his query and every such query as well as its answer is stored in the database for possible future reuse. It is considered that every annotation (tag) which may exist in the database is potentially known to the malevolent user. The proposed mechanism provides an answer only if k-anonymity principle is not violated w.r.t the user's current history.

Given the above, this paper makes the following contributions:

- We trace various types of attacks and thus privacy violations that malevolent users may try by querying the original enriched trajectory database.
- We device a query-based auditing mechanism that can effectively identify and block the potential attacks that could lead to user identification or tracking.
- We propose the *LENS* algorithm aiming at increasing user friendliness of the proposed mechanism by modifying the (original) query posed that cannot be answered due to privacy violation, to the 'nearest' query that can be safely answered.

The rest of the paper is structured as follows: Section 2 presents related work. Section 3 introduces different types of attacks of a malevolent user. Section 4 provides the auditing mechanism that handles the previously described attacks as well as the *LENS* algorithm. Finally, Section 5 concludes the paper.

2 Related Work

The *k*-anonymity principle [14] is the most common approach that has been adopted for the anonymization of both relational and mobility data. For mobility data, it states that a dataset must be anonymized so that every trajectory is indistinguishable from at least *k*-1 other trajectories.

Hoh and Gruteser [5] presented a data perturbation algorithm that is based on path crossing. Terrovitis and Mamoulis [15] consider datasets as sequences of places visited by users and proposed a suppression technique that eliminates the least number of places from a user's trajectory, so that the remaining trajectory is *k-ano*nymous. Abul et al. [1] proposed a *k*-anonymity approach that relies on the inherent uncertainty of moving objects whereabouts, where a trajectory is considered as a cylinder. The anonymity algorithm employs space translation and generates clusters of at least *k* trajectories. Each cluster of *k* trajectories forms an anonymity region and the co-clustered trajectories can be released. To achieve space-time translation, the authors proposed W4M [2], which uses a different distance measure that allows time-warping.

Nergiz et al. [10] proposed a coarsening strategy to generate a sanitized dataset that consists of *k*-anonymous sequences. Monreale et al. [8] proposed another anonymization approach that is based on the combination of spatial generalization and *k*-anonymity principle, while in [9] authors faced the problem of anonymizing semantic trajectories. To release a safe version of a semantic trajectory dataset, they propose a method that generalizes sequences of visited places based on a privacy place taxonomy.

On the other hand, in several sharing scenarios, many consider that data should stay in-house to the hosting organization and only a specific number of authorized users should have access on them. In this way, the hosting organization would be able to confront several possible legal restrictions, it could record the individuals who use the database and it would update the database whenever appropriate. Despite all the

above, a mechanism is, in any case, needed in order to ensure that no sensitive information will be released during this process. Along this direction, methodologies have been proposed for disclosure control in statistical databases [3], but they support only count and/or sum queries.
In [4] the authors proposed an envisioned query engine where subscribed users had restricted access to the database to accomplish various analysis tasks and in [12] and [13] a privacy-aware query engine was introduced that can protect the trajectory database from potential attacks, while supporting popular queries for mobility data analysis. Both approaches deal with spatiotemporal trajectory databases not enriched, as in our case. This paper improves the approach in [6]. Finally, in [16] authors proposed a data stream management system, aiming at preserving users' privacy by enforcing Hippocratic principles.

3 Privacy Attacks

The main purpose of every attack of a malevolent user is to broaden his knowledge about an individual or a situation that interests him. This occurs when the attacker *raises his confidence* about an event that may be related to an individual, who is the 'target' or a situation for which he wishes to acquire more specific knowledge. Usually, a malevolent user has prior knowledge, i.e. time, place, type of episode and/or semantics (or any possible combination) about an individual.

3.1 Totally Overlapping Queries

Overlapping queries is a sequence of at least two queries posed by a user, having as a characteristic that the criteria of these successive queries are overlapping. We assume that either the queries have the same number of sub-queries and their criteria differ only in one dimension (space / time / semantics) or they differ in the number of sub-queries that each one contains keeping their 'common' sub-queries totally identical between them w.r.t. their criteria.

Spatial Overlapping. In this attack, the identity of a user can be revealed by posing overlapping queries, which differ in spatial dimension. The attacker poses a query and if the number of trajectories is at least k, he proceeds with one or more queries modifying each time only the spatial dimension, such that every time the new query contains the previous one. In fact, the most targeted way for a malevolent user to perform a 'successful' attack is to modify the spatial criterion of only one sub-query (or the same sub-query each time).

Let's assume that a user poses query Q_1 that contains n trajectories where $n \geq k$ and then Q_2 which returns as an answer $n+m$ trajectories, where $m<k$. The malevolent may conclude that the area corresponding to the difference between Q_1 and Q_2 contains m trajectories which is less than threshold k, thus privacy violation is occurred.

Consider the example depicted in Figure 1. A user poses query Q_1: *Find people starting from area A between [8.00-8.30am] and, then, stop at area B between [9.15-11.30pm].* This query contains two different sub-queries, each one able to provide an answer, if posed independently from the other. The same user poses query Q_2: *Find people starting from area A' between [8.00-8.30am] and, then, stop at area B between [9.15-11.30pm].* The answer of Q_1 contains 7 trajectories while the answer of Q_2 contains 8 trajectories. Thus, the malevolent can easily infer that only *one* person appears in the *area [A'-A] between [8.00-8.30am]* and subsequently stopped at *area B between [9.15-11.30pm].* By combining this knowledge with additional information, the malevolent can identify this person and extract the information that he was alone at that specific time and place.

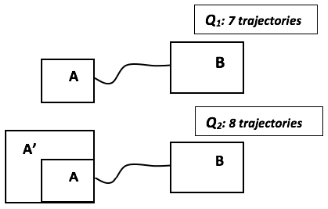

Figure 1: An example of two Overlapping Queries

The same reasoning, described above, can be followed regarding attacks applied by using two or more successive queries, which overlap in the temporal dimension.

Annotations Overlapping. In this case, a malevolent user may attempt to pose successive queries that take into account differences regarding only the semantic dimension. Let's examine the following example assuming that k=3. The user poses query Q_1 which consists of one sub-query and the answer corresponds to a specific spatiotemporal area that includes 7 *stop* episodes. During Q_2 the malevolent maintains the criteria of Q_1 but also adds *tag*='work'. The output of Q_2 contains 6 trajectories. The malevolent can conclude that only *one* entity was not working and he/she may be a workplace visitor.

Another example, using k=3, is this. Supposing that a malevolent user knows beforehand that a previously monitored/recorded person was found in a specific spatiotemporal area and had tag = 'fun' and this person claimed that he/she was accompanied by another (also monitored) person. The user poses query Q_1 regarding this area without adding any specific tag annotation and it returns 4 episodes. By posing Q_2 the user adds *tag*='work'. If the result of Q_2 contains 3 trajectories then the recorded person is lying.

Different Number of SQs. In this attack, a malevolent user poses a sequence of queries which differ only in the number of the sub-queries that each one contains. Between two consecutive queries, assume that Q_1 contains n sub-queries and Q_2 contains $n+m$, where $m \geq 1$. In order to achieve an attack the malevolent should know that his target 'participates' in the n sub-queries. Subsequently, the malevolent may compare the number of the trajectories corresponding to the query consisting of the n sub-queries in relation to the one with the $n+m$ sub-queries. If the difference of the number of trajectories that constitute the answer of these two queries is less than k, this may result in revealing sensitive information about the target.

Consider the following example. The malevolent is aware of a target's home and working address and he/she intends to learn if the target returned at his residence after work. Assume that k=4. The user poses the query Q_1 consisting of two sub-queries that definitely 'contain' the target's trajectory (how many people stayed during the night in area A and during the following day were working in area B). The number of trajectories that fulfill Q_1 is 5. Q_2 contains exactly the same sub-queries with Q_1 along with a new sub-query that asks for those that returned after work back to area A during the night. If the answer of Q_2 returns 4 trajectories, then there is an 80% probability that the target spent indeed the night at home. This constitutes a significantly increased certainty compared to the 'safety limit' defined by the k-anonymity threshold. In other words, the malevolent should never be more than 25% certain about any situation 'recorded' in a database protected by a k=4 threshold. If the result again was 5, then he would be absolutely certain that the target returned home. We reasonably understand that if the user performed (only) the Q_2 from the beginning and regardless of the outcome, the result would be the same, but this *would not increase* his or her certainty about the behavior of the target in any way.

3.2 Multiple Intersecting Queries

This type of attack requires a sequence of, at least, 3 queries posed by a user. For simplicity reasons, let's assume that we are dealing with queries which comprise only one sub-query. The sub-query's criteria with which an attack can take place in this case are only space and time. We assume that the triple (at least) of the queries posed by the user differs either in space or in time each time, thus the malevolent user will be able to reach more specific and therefore alarming findings using the semantic database.

Consider the example depicted in Figure 2. Supposing that k=3, the user initially poses the query Q_1 whose spatial criterion corresponds to *area A* and it includes 5 episodes/trajectories. Then, he/she poses query Q_2 whose area corresponds to *area B* which covers part of *area A* along with some *extra area* and it returns

3 episodes. If a third query *Q3* (*area C*) corresponds exactly to the remaining part of *area A* that does not intersect with *area B* and it includes 3 episodes, then a privacy breach takes place. This is happening because, although none of the queries above do not overlap with each other, considering the fact that the 3 out of 5 episodes of *area A* are situated in *area C,* the user will be able to conclude that among the 3 episodes belonging to *area B* the two of them are situated within the *area A* and *only one episode* corresponds to the part of *area B* that does not intersect with *area A*.

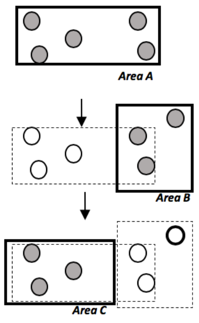

Figure 2: An example of three Intersecting Queries

Similarly, every time a user alters marginally only the temporal dimension between at least 3 successive intersecting queries and the spatial dimension remains the same, similar situations may arise.

4 Attack Prevention

To prevent the previously described attacks, an auditing mechanism is required to: (a) ensure that k-anonymity principle is not violated before answering each query; (b) protect certain episodes that may reveal sensitive information about entities and should be addressed in a specific way; (c) activate the *LENS* algorithm which allows the appropriate modification of the original query if k-anonymity principle is violated, in an attempt to make it acceptable; (d) allow the data owner to monitor and record constantly all the queries posed per user as well as the corresponding answers in order to compare them with every new query posed to the database.

4.1 Sensitive Episodes

Sensitive episodes correspond to known locations that contain particularly sensitive information and can possibly expose the identity of a user. We call these locations *sensitive* for a user as no such information should be disclosed to the attackers. In order to deal with user-defined sensitive episodes, the auditing mechanism initially does not count them as part of the query's answer set, but if the number of the non-sensitive episodes corresponds to at least *k* trajectories, then these sensitive episodes are allowed to be finally incorporated in the answer set.

4.2 LENS Algorithm

When a user poses a query to the database, he is willing to gain knowledge about whether there are semantic trajectories that are answering the query w.r.t. certain criteria. If the number of the semantic trajectories, composing the result set, is less than the anonymity threshold *k*, the query is not safe to be answered. This ensures a first level of privacy protection.

The main idea of the proposed approach is that, instead of not providing an answer whenever *k*-anonymity principle is violated, an auditing mechanism should try to answer the query posed by a user in any case. In other words, the mechanism will provide an answer of the most 'similar' query to the original, that fulfills *k*-anonymity principle by relaxing conditions via generalization. Query relaxation enlarges the search range to include additional information. The output of this process is like a generalized query in one or more possible dimensions. Put differently, a user seeks to query an area, but the mechanism resolves it for a zoomed-out area that is generalized up to a permissible degree of analysis.

The main goal of implementing such an approach is to increase user friendliness and improve database functionality. A user can gain information without posing consecutive queries expanding repeatedly the criteria set until an answer is provided, because in this case there is an extra computational cost to the database and the possibility the user might not be given the best possible answer. To achieve this, the mechanism allows the generalization of one or more criteria, of the sub-queries that constitute the original query. The generalization may occur in the spatial, the temporal or even the semantic dimension.

Let's assume that the potential answer of an initial query *Q* consists of less than *k* trajectories. The output of the algorithm is a modified query *Q'*. If the modification process is successful, then the execution of the query *Q'* will result in at least *k* trajectories. The number of the sub-queries in both *Q* and *Q'* is of the same size. Each sub-query of *Q'* is either the same or generalized based on the corresponding sub-query contained in *Q*.

An obvious approach would be to apply the aforementioned method only in case that a query cannot be answered marginally w.r.t. to *k*-anonymity threshold. A threshold should then be defined. The mechanism, based on it, would be activated every time the query could not be answered. But in case the mechanism is activated when only few trajectories are missing from the answer set, a privacy breach may occur. A malevolent user can easily assume that the modified query does contain certain number of additional trajectories within the returned extra area. An obvious solution is to apply *LENS* algorithm, regardless of the number of trajectories needed to reach *k*-anonymity principle.

The goal of the algorithm is to modify one or more sub-queries, so that more episodes will be included per sub-query. To help the algorithm decide which episode is preferable to be included in the answer of the modified query, it should be able to compare the distortion that is caused on each sub-query, when trying to select between two or more candidate episodes. A unit, that calculates the distortion caused due to the generalization of one or more dimensions on each sub-query, is required. The distortion should be as low as possible to limit the generalization to the necessary level w.r.t. the initial query posed by the user.

The *LENS* algorithm takes as input a semantic trajectory database, the original query posed by a user that cannot be answered, the anonymity threshold *k,* the distortion limit value *dist* and a matrix *H*. The output of the algorithm is the modified query along with the corresponding sub-queries.

The algorithm after the initialization process (lines 1-2) continues with a loop phase where each sub-query of the original query is executed individually and the trajectories that comprise each one are retrieved (line 6). Each trajectory id of these trajectories is inserted into a matrix *(H)* along with the frequency indicating its appearance *(freq)* in all sub-queries (line 7). When a trajectory id first enters the matrix receive 1 as the value of *freq* and its frequency is increased by 1 every time the same trajectory (through its episodes) is found within the answer set of the sub-queries subsequently executed. Thus, *H* is a set of tuples each of which contains trajectory id (*traj_id*), frequency (*freq*) and the sub-queries (SQ_i). The maximum value that the counter *(freq)* can take in each record is equal to the number of the sub-queries. Consider as an example a query with three sub-queries. The counter for each trajectory in matrix *H* will receive a value ranging from 1 up to 3. If the counter receives the maximum value, it means that the episodes of this trajectory have been identified in all sub-queries, thus this trajectory can be returned as an answer to the overall modified query.

Based on matrix *H*, the algorithm detects the trajectory or trajectories with frequency (i.e., number of sub-queries), less than the maximum possible frequency, but at the same time with the highest value among the other trajectories in the matrix (line 9). Subsequently, a loop starts that ensures that, if no episode is found, the algorithm will search the trajectory that has the subsequent smaller frequency. This loop ends either when permissible episodes can be integrated, or if all the remaining trajectories from the matrix have been investigated and no episode is found (line 15). To define the most appropriate candidate episodes, the algorithm employs a process called *Compute_Distortion_Units*. A metric function calculates the distortion caused in a sub-query in order to be modified and be able to include an episode from the trajectory. In case we have a distortion unit greater than a distortion limit (user-defined), the episode takes the tag INF and the algorithm proceeds with the next trajectory. Under these conditions the algorithm selects as preferable the episode that has the lowest distortion (line 13).

As a next step, the sub-query is modified in one or more dimensions to contain the episode that minimizes the distortion (line 17). The repetition ends (line 19) either if *k* trajectories have frequency equal to the number of sub-queries or if no episodes were integrated.

Algorithm 1. *LENS*

Input: (1) anonymity threshold k, (2) initial query with sub-queries Q = <SQ_1, SQ_2,…, SQ_n>, (3) a semantic trajectory database D, (4) distortion limit *dist*, (5) array H[tr_id, freq, SQ_1, SQ_2, …, SQ_n]
Output: Q'= <SQ'_1, SQ'_2,…, SQ'_n>

```
1.  Q' ← Q; H ← ∅
2.  N_tr ← Count(Q')
3.  repeat
4.      Something_Changed ← False;
5.      for i=1 to n do
6.          Execute_Query(in SQ'_i out tr_ids)
7.          Fill_Help_Table(in H, tr_ids out H)
8.      end for
9.      Find_Freq_Position(in H, n out i)
10.     episode_found ← False
11.     repeat
12.         Compute_Distortion_Units(in out episode_found, H, i)
13.         Select_best_candidate_episode(in H, dist, i out tr_id, ep_id)
14.         i ← i+1
15.     until episode_found or EOF
16.     if episode_found then
17.         Embed_New_Episode(in H, tr_id, ep_id, in out SQ'_i, Something_Changed)
18.         N_tr ← Count(Q')
19. until (not Something_Changed) or (N_tr=k)
20. Compute_Random_Number(in Rmin, Rmax out R)
21. Compute_New_Episodes(in Q', R out Q')
22. return Q'
```

During the generalization process of the sub-queries, a privacy breach may occur. Let's assume that the spatial dimension of the area that the query covers is enlarged, so as to contain exactly *k* episodes. The spatial generalization should be the minimum possible in order to keep the distortion caused from this process as low as possible. To achieve this, most of the episodes that are added will appear in the borders of the modified area. The malevolent user, thus, will be more confident that, at least one episode exists, between the query posed and the modified query that is finally answered. In order to avoid such a violation, the modified query is expanded on each side by a randomly generated percentage *R* (line 21). Finally, we get as output the final modified query along with its sub-queries (line 22).

4.3 Query Auditing

The main goal of the *Query-Auditing* algorithm is to prevent the semantic database from answering a query to the user when it is possible that he may acquire further knowledge of a situation or a specific individual whose trajectories are stored on the database than he might have had before posing the query. The only way to avert this is to deny answering the query. In order to apply such a mechanism, it is necessary to keep the past user's queries along with the corresponding answers stored in the database.

Related works do not provide any answer when two queries posed by the same user are overlapping, aiming to prevent any privacy violation. In order to increase user friendliness and system functionality, we argue that the previous approach is very conservative and the algorithm should proceed to further examination before denying an answer. In other words, we find that it may be very strict to deny an answer without previously examining the extent that every query, ready to be answered, is overlapping to every query previously posed by the same user. More specifically, since each user has the right to receive an answer every time the k-anonymity principle is not violated, we argue that he still has the same right even if his current query overlaps a previous one, as long as the difference of these two queries' answers (i.e. the number of trajectories contained in each answer) equals or is greater than *k* threshold.

However, after applying this policy to the auditor, this may lead to a new type of potential privacy breach. More specifically, let's consider the following example assuming that a *k*=2 threshold is applied. Supposing there is a query Q_1 having only one sub-query and its temporal criterion defined between [8.00-

11.00am] and it contains 5 trajectories. Next, if the same user poses a query *Q_2*, whose temporal dimension is [8.00-9.00am] and it contains 2 trajectories, the auditor should allow it to be answered, because the difference of these two answers is greater than *k*. Subsequently, if the same user tries a third query *Q_3* whose temporal dimension is [9.00-10.00am] which contains also 2 trajectories, it seems that it should also be answered because although it overlaps (only) with *Q_1*, their difference is still greater than *k*. If so, a privacy violation will take place because this specific user will eventually get to know that between [10.00-11.00am] only one entity has been recorded.

The solution we propose is to create dummy queries stored in the database along with the actual queries previously posed by the user and, in any case, the database will treat them as real queries posed in the past by him. Regarding the previous example, when the auditor replies to query *Q_2*, we create a dummy query with temporal criterion defined between [9.00-11.00am]. This would prohibit the user from querying *Q_3* successfully. Then, we present specific examples, regarding the different types of overlapping queries that the auditor has to deal with.

Spatiotemporal Overlapping. Let's assume that *k*=3. A user poses initially the query Q_1: A→B→C (i.e. the query consists of 3 sub-queries) and subsequently the query Q_2: A→B→C'. The sub-queries C and C' are overlapping. The auditor examines the number of trajectories corresponding to the aforementioned queries and, let's assume, that Q_1 is satisfied by 7 semantic trajectories while the Q_2 by 4 ones. The difference of these two queries is equal to *k,* so the user will be allowed to receive answers for these two successive queries. However, in order to avoid future violation by the specific user, the auditor has to construct a fake query that will be stored in the database and corresponds to the spatiotemporal difference of the two queries Q_1 & Q_2 regarding their last sub-queries C & C', before releasing the final answer (i.e. Q_{fake}: A→B→[C∩C']). Thus, if the same user attempts a third actual query (4th, including the fake one) Q_3: A→B→C'' and C'' totally overlaps [C∩C'], the same comparison between the number of trajectories of Q_{fake} & Q_3 will take place and if k-anonymity is still satisfied, then it will be answered too and a new fake query will be created, and so on. A similar approach to this also applies to multiple intersecting queries.

Annotations Overlapping. Our approach for a friendlier handling of queries in the case of overlapping tags is (see Sec. 3):

$$\|Q[\text{tag is null}]\| - \sum \|Qs[\text{tag is not null}]\| \geq k$$

Before answering the query, the auditor compares the number of trajectories corresponding to the query, without containing any specific tag, with the sum of the number of trajectories corresponding to all the other queries which contain tag(s) in their criteria and have already been answered. Obviously, all of the aforementioned queries are totally identical except only for the annotation criterion of only one sub-query and we note that the inequality also includes the query the auditor is examining at that time. If this inequality is true, then the auditor proceeds in answering the current query. There is, also, no need for generating a fake query when answering such queries.

Algorithm 2. Query-Auditing

Input: (1) anonymity threshold k, (2) initial query with sub-queries Q = <SQ_1, SQ_2,…, SQ_n>, (3) a semantic trajectory database D, (4) distortion limit dist, (5) user id u_{id}

Output: F_Q

```
1.  F_Q ← Q; Q_D ← ∅
2.  Remove_Sensitive_Episodes(in out F_Q)
3.  if ||F_Q||<k then
4.     LENS(in k, F_Q, D, dist, H out F_Q)
5.     if ||F_Q||<k then
6.        return false
7.  for each Q_i ∈ D where user_id=u_id do
8.     for each SQ_j ∈ Q_i do
9.        for each SF_Qm ∈ F_Q do
10.          if SF_Qm overlaps SQ_j then
11.             Check_Suspicious_Query(in F_Q, Q_i out Is_Suspicious, Dummy_Query_Needed)
12.             if Is_Suspicious then
13.                if $| \|F_Q\| - \|Q_i\| | \geq k$ then
14.                   if Dummy_Query_Needed then
15.                      Create_dummy_query(in F_Q, Q_i out Q_D)
16.                else
17.                   return false
18.       end for
19.    end for
20. end for
21. Add_Sensitive_Episodes(in out F_Q)
22. D ← D U Q_D
23. return F_Q
```

Different number of SQs. Let's assume that *k*=3 and a user poses the query *Q_1*: A→B→C→D and it is satisfied by 4 semantic trajectories. The same user poses *Q_2*: A→B→C and the answer contains 7 trajectories. Since the difference of these two queries that corresponds to the query A→B→C→not [D] is equal to *k,* no privacy violation can be caused. In fact, in this case there is no need for generating a fake query.

Algorithm's Description. The input of the algorithm is the anonymity threshold *k*, the initial query of the user along with the corresponding sub-queries, a semantic trajectory database *D*, the distortion limit value *dist* and the *id* of the user posing the query. The algorithm first executes query *Q* and gets the number of trajectories that make up the answer set (line 1). Then, the episodes that are considered as sensitive are defined and removed from it (line 2). If the number of trajectories is less than *k*, *LENS* algorithm, previously described, is called to modify the original query in an effort to provide an answer (lines 3, 4). If *LENS* algorithm cannot manage to modify the query w.r.t. a distortion threshold, no answer is provided to the user and the algorithm ends (lines 5, 6). On the contrary, if the original query *Q* or the modified query *F_Q* has equal or more than *k* trajectories, the auditing mechanism continues to further investigate this query based on user's history.

The auditor proceeds by comparing every sub-query of *F_Q* with all the sub-queries that belong to queries posed from the user in the past. Every time two sub-queries are overlapping w.r.t their spatial, temporal or semantic dimension (including the case where these two are just intersecting each other) or they are totally identical, a procedure *Check_Suspicious_Query* is called to examine whether query *F_Q* is, in fact, a suspicious query compared to the current query Q_i which is a previously posed query stored in the Database (lines 10-11). If so, the auditor calculates the difference of the number of trajectories belonging to the corresponding queries (lines 12-13). If it is equal or greater than *k* (which means that the query will be answered in any case) and if the procedure *Check_Suspicious_Query* has previously determined the need of creating a dummy query, the auditor calls the procedure *Create_dummy_query* which creates the query *Q_D* (lines 14-15). Otherwise, the algorithm ends and no answer is provided to the user (line 17). Finally, if the query is to be executed, any sensitive episodes, that have been initially excluded, are added to the answer, the dummy query is stored in the database and the final answer is returned to the user (lines 21-23).

5 Conclusions

In this paper, we proposed an envisioned query engine able to provide safe answers to queries posed by users in semantic trajectory databases. Different types of privacy attacks have been addressed and an effective auditing mechanism able to prevent privacy breaches has been proposed. We subsequently introduce *LENS* algorithm which is able to modify an initially inacceptable query to the 'closest' one that can be safely answered, thus increasing the user friendliness of the engine. In the future, we plan to support multiple users access of the MOD.

ACKNOWLEDGMENTS

This work was partially supported by project MASTER (Marie Sklowdoska-Curie agreement N. 777695) that has received funding from the EU Horizon 2020 Program.

REFERENCES

[1] O. Abul, F. Bonchi, M. Nanni, 2008. Never walk alone: Uncertainty for anonymity in moving objects databases. In Proceedings of ICDE.

[2] O. Abul, F, Bonchi, M, Nanni, 2010. Anonymization of moving objects databases by clustering and perturbation. Information Systems, 35(8):884-910.

[3] N.R. Adam, J. C. Worthmann, 1989. Security-control methods for statistical databases: A comparative study. ACM Computing Surveys, 21(4):515-556.

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

[5] B. Hoh, M. Gruteser, 2005. Protecting location privacy through path confusion. In Proceedings of SecureComm.

[6] D. Kopanaki, N. Pelekis, 2017. Privacy Preservation of Semantic Trajectory Databases using Query Auditing Techniques. In Proceedings of BIGQP @ EDBT/ICDT 2017).

[7] R. Mello, V. Bogorny, L. Alvares, Z. Santana, H. Luiz, C. Ferrero, A. Frozza, G. Schreiner, C. Renso. MASTER: A Multiple Aspect View on Trajectories. Transactions in GIS Journal, to appear.

[8] A. Monreale, G.L. Andrienko, N.V. Andrienko, F.,Giannotti, D. Pedreschi, S., Rinzivillo, S. Wrobel, 2010. Movement data anonymity through generalization. Transactions on Data Privacy, 3(2), pp. 91-121.

[9] A. Monreale, R. Trasarti, D. Pedreschi, C. Renso, V. Bogorny, 2011. C-safety: a framework for the anonymization of semantic trajectories. Transactions on Data Privacy, 4(2), pp.73-101.

[10] M. E. Nergiz, M. Atzori, Y. Saygin, 2008. Towards trajectory anonymization: a generalization-based approach. In Proceedings of Security and Privacy in GIS and LBS, SIGSPATIAL.

[11] C. Parent, S. Spaccapietra, C. Renso, G. Andrienko, N. Andrienko, V. Bogorny, M.L. Damiani, A. Gkoulalas-Divanis, J. Macedo, N. Pelekis, Y. Theodoridis, 2013. Semantic trajectories modeling and analysis. ACM Computing Surveys (CSUR), 45(4), p.42.

[12] N. Pelekis, A. Gkoulalas-Divanis, M. Vodas, D. Kopanaki, Y. Theodoridis, 2011. Privacy-aware querying over sensitive trajectory data. In Proceedings CIKM.

[13] N. Pelekis, A. Gkoulalas-Divanis, M. Vodas, A. Plemenos, D. Kopanaki, Y. Theodoridis 2012. Private-HERMES: a benchmark framework for privacy-preserving mobility data querying and mining methods. In Proc. of EDBT.

[14] L. Sweeney, 2002. k-anonymity: a model for protecting privacy. International Journal on Uncertainty, Fuzziness and Knowledge Based Systems, 10(5):557-570.

[15] M. Terrovitis, N. Mamoulis, 2008. Privacy preservation in the publication of trajectories. In Proceedings of MDM.

[16] H. Wu, S. Xiang, W.S. Ng, W. Wu, M. Xue, 2014. HipStream: A Privacy-Preserving System for Managing Mobility Data Streams. In Proc. of MDM.