Privacy Preservation in the Publication of Trajectories*

Manolis Terrovitis and Nikos Mamoulis
Department of Computer Science
University of Hong Kong
Pokfulam Road, Hong Kong
{rrovitis,nikos}@cs.hku.hk

Abstract

We study the problem of protecting privacy in the publication of location sequences. Consider a database of trajectories, corresponding to movements of people, captured by their transactions when they use credit or RFID debit cards. We show that, if such trajectories are published exactly (by only hiding the identities of persons that followed them), there is a high risk of privacy breach by adversaries who hold partial information about them (e.g., shop owners). In particular, we show that one can use partial trajectory knowledge as a quasi-identifier for the remaining locations in the sequence. We device a data suppression technique, which prevents this type of breach, while keeping the posted data as accurate as possible.

1. Introduction

Consider a card company, which keeps track of customer transactions. Assume that the company wants to publish sequences of transactions by the same person as trajectories, for analysis and querying purposes. As a motivating example, consider the Octopus[1] smart RFID card, commonly used by Hong Kong residents to pay for their transportation and for their transactions at point-of-sale services, such as shops, restaurants, parking spaces, vending machines, etc. The Octopus company accumulates vast amounts of trajectory data daily, which could be published in order to extract the movement and behavioral patterns of Hong Kong residents and the causality relationships between geographic spots on the city's map.

On the other hand, direct publishing of this information, even after hiding the IDs of users, may easily result in privacy breach, once combined with the partial trajectory information, known to an adversary. For example, when a person uses his Octopus card to pay at a convenience store, the store can also keep this transaction in its own database and associate it to the person's identity (e.g., via a loyalty/bonus card). Furthermore, if the store belongs to a chain (e.g., 7-Eleven), it can know multiple and different locations of the same person's activities (assuming that he visited shops of the same company multiple times), which constitute a subset of his complete trajectory.

Figure 1a shows an example of a database T owned by the publisher (e.g., Octopus).[2] Each sequence element is a shop address, where the corresponding user did his/her card transactions. Locations are classified according to the possible adversaries. For example, all places denoted by a_i (where i is any integer) are assumed to also be tracked by company A (e.g., 7-Eleven). Figure 1b shows T_A; the knowledge A has for the exact database of Figure 1a.

We now show how company A (acting as an adversary) could identify private information by combining its knowledge T_A with the database T, if T is published directly. We assume that company A can associate each t_i^A in its local database T_A to a real user identity and we want to prevent A from identifying the locations in t_i, which are unknown to A, with probability higher than P_{br} (breach probability). The first type of link attack is due to the direct mapping of a tuple in T_A to a tuple in T. For example, if T is published, A will know that t_5^A actually corresponds to t_5, since t_5 is the only trajectory that goes through a_1 and a_3, but no other location of company A. Therefore A is 100% sure that the user who followed t_5^A in its local database, visited b_1. Even for sequences that adversary A cannot map exactly to its own database, privacy breach can occur. Consider, for instance, sequence t_7^A, which A cannot distinguish whether it is t_6, t_7, or t_8.[3] Based on T, adversary A can infer that the user corresponding to t_7^A visited location b_2 with probability 66%. This is unacceptable, if $P_{br} = 50\%$. This type of breach is due to the lack of location diversity between trajectories that map to the same set of partial routes

*Work supported by grant HKU 7155/06E from Hong Kong RGC.

[1] http://www.octopuscards.com/

[2] For the ease of presentation, we model each trajectory by a sequence of locations, without temporal information (i.e., timestamps) on the elements. We will discuss about timestamped trajectories later in the paper.

[3] Note that the sequences are published in random order, so an attempt to map T_A to T based on the order of trajectories in T would fail.

in the attacker's database. By constructing the projected database T_B of adversary B, the reader can easily identify similar breaches of A-locations (e.g., the B-projection of t_8 is unique in T_B and adversary B can find out that the corresponding person visited a_3).

id	trajectory
t_1	$a_1 \rightarrow b_1 \rightarrow a_2$
t_2	$a_1 \rightarrow b_1 \rightarrow a_2 \rightarrow b_3$
t_3	$a_1 \rightarrow b_2 \rightarrow a_2$
t_4	$a_1 \rightarrow a_2 \rightarrow b_2$
t_5	$a_1 \rightarrow a_3 \rightarrow b_1$
t_6	$a_3 \rightarrow b_1$
t_7	$a_3 \rightarrow b_2$
t_8	$a_3 \rightarrow b_2 \rightarrow b_3$

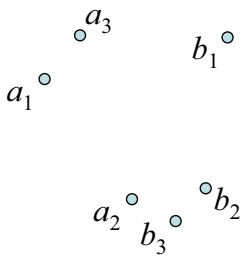
(a) exact data (T)

id	trajectory
t_1^A	$a_1 \rightarrow a_2$
t_2^A	$a_1 \rightarrow a_2$
t_3^A	$a_1 \rightarrow a_2$
t_4^A	$a_1 \rightarrow a_2$
t_5^A	$a_1 \rightarrow a_3$
t_6^A	a_3
t_7^A	a_3
t_8^A	a_3

(b) A's knowledge (T_A)

id	trajectory
t'_1	$a_1 \rightarrow b_1 \rightarrow a_2$
t'_2	$a_1 \rightarrow b_1 \rightarrow a_2$
t'_3	$a_1 \rightarrow b_2 \rightarrow a_2$
t'_4	$a_1 \rightarrow a_2 \rightarrow b_2$
t'_5	$a_3 \rightarrow b_1$
t'_6	$a_3 \rightarrow b_1$
t'_7	$a_3 \rightarrow b_2$
t'_8	$a_3 \rightarrow b_2$

(c) transformed database (T')



(d) the map of locations

Figure 1. Trajectory database

In this paper, we study the problem of publishing the trajectories in a way that would prevent adversaries from using their partial knowledge (linked to a person's identity) to infer locations unknown to them. Our technique suppresses location information in T wherever privacy leaks occur and converts it to a secure, published database T'. The two conflicting goals of this data transformation are (i) the satisfaction of certain privacy requirements (i.e., P_{br}) by the published data and (ii) maximizing the utility of the published data measured by the similarity of the original trajectories in T to their corresponding transformations in T'.

For each adversary A and for every projected sequence $t_i^A \in T_A$ (corresponding to $t_i \in T$), we would like the adversary, not to be able to reconstruct any point in $p \in t_i, p \notin t_i^A$ with certainty higher than P_{br}. For example, if $P_{br} = 50\%$, then the exact publication of the database T of Figure 1a is not secure with respect to adversary A (who knows the data of Figure 1b), as already explained. Our method converts T to T' (see Figure 1c) after suppressing b_3 from t_2, a_1 from t_5, and b_3 from t_8. The privacy with respect to adversaries A and B is preserved with the publication of T'. For instance, each of the first four projections $a_1 \rightarrow a_2$ in T_A is mapped to four trajectories $t'_1 - t'_4$ in T', in which b_1 and b_2 appear with probability $50\% \leq P_{br}$. The same holds for the last three trajectories in T_A. Note that t_5^A cannot be mapped to any trajectory in T'. In the worst case, the adversary can infer that t_5^A must be one of t'_5 to t'_8, since the A-projection a_3 of these four sequences appears only three times in T_A (and a_1 could have been truncated from the original mapping of t_5^A in T). As we will discuss later in the paper, this case does not entail privacy threats. When selecting locations to truncate from sequences, our method considers both the satisfaction of the privacy constraints and also the introduced error with respect to the original trajectory. For example, the deleted points in the transformed database T' of Figure 1c are close to their neighboring points in the corresponding original sequences (as shown in the map of locations in Figure 1d).

The remainder of the paper is organized as follows. Section 2 summarizes related work. Section 3 formally defines the privacy preservation problem that we study in this paper. In Section 4 we present our algorithm, which is experimentally evaluated in Section 5. Finally, Section 6 concludes the paper with a discussion about future work.

2. Related work

This section provides background on privacy preservation in the publication of databases that contain sensitive information. Trajectories of moving objects are data sequences, therefore work on the related issue of anonymizing locations and sequences thereof is also reviewed.

2.1. k-anonymity and ℓ-diversity

An issue that has risen lately in database research is the publication of microdata (e.g., hospital records) that contain one or more sensitive attributes (e.g., disease). Organizations, such as hospitals, want to release such data (e.g., for research) without compromising the privacy of the patients. Simply hiding the explicit identity of persons (i.e., name, ID) before publication does not suffice as recent studies demonstrate [6, 19]. In particular, a set of non-sensitive attributes of a person (e.g., gender, age, zipcode) may act as a *quasi-identifier* to reveal the association of him/her with a published record. For instance, by joining a (public) voting registration database with anonymized medical records, one may easily infer the identity of a patient who has unique quasi-identifier compared to others in the voting registration table; if Bob is the only male of age 55 in his neighborhood, then any of his visits to the hospital (and even his decease) can be easily inferred from a published medical records database. To address this problem, k-anonymity [18, 17] transforms the database before publication, such that the values of attributes in the quasi-identifier are suppressed or generalized in some of the published records. In a k-anonymized published database, each tuple has the same quasi-identifier values with at least $k-1$ others (all these tuples form an *equivalence class*).

Research on k-anonymization initially focused on generalizing each attribute individually [2, 11]; e.g., all instances

of age=55 are generalized to the same range 53–57. Later, LeFevre et al. [12] noticed that the quality of anonymization can be improved if equivalent classes are defined by arbitrary, non-overlapping multi-dimensional ranges in the quasi-identifier space. In [1], k-anonymity is treated as a special clustering problem, and a constant factor approximation of the optimal solution is proposed. In the same direction, [20] propose more efficient agglomerative and divisive recursive clustering algorithms for this problem.

A stronger version of k-anonymity (called ℓ-diversity [13]) requires the diversity of sensitive values (e.g., decease) that appear in each equivalence class. For example, if Bob's values in the quasi-identifier attributes (e.g., gender=male, age=55, zipcode=17728) fall in the same class (e.g., male, 53–57, 17001–18000) as $k = 10$ other persons and all these persons have diabetes, then even after k-anonymization, an adversary can infer that Bob had diabetes when he visited the hospital. ℓ-diversity prevents such attacks by ensuring that at least ℓ sensitive values are well-represented in each equivalence class.

Our problem has two major differences from the classic k-anonymity and ℓ-diversity problems. First, the sensitive information is not absolute, but relative to the adversary's knowledge. In other words, the part of a trajectory known to an adversary is not sensitive which respect to him; however, the remaining information should be hidden from him. We make no assumptions about the sensitivity degree of each location; we consider all equally sensitive. Second, the trajectory projections, used as quasi-identifiers are sequences of variable length. As a result, our model and privacy preservation techniques are fundamentally different than existing work on secure publishing of relational data.

2.2. Location and Time-series Anonymization

Location-based applications support user queries (e.g., nearest neighbor search) which have as input the location of the issuer. For example, GPS-enabled mobile phones can be used to search for nearby services, like restaurants and gas-stations. One problem when using such services is that the public server that is queried (e.g., Google Maps) can combine the location of the user with some external information (e.g., a residential addresses database) to infer the identity of the query issuer. As a result, the privacy of user activities may be at risk. To tackle this problem a similar idea to k-anonymity can be used. Location-based queries are sent from users to the public sender through a trusted *query anonymizer* which (i) hides the identity of the user and (ii) applies a *spatial-cloaking* technique that generalizes the user location (original query input) to a greater query region that contains at least k locations of different users the time of the query. As a result, the query issuer is indistinguishable from at least k other users and the privacy of his/her action is well-preserved. The query anonymizer retrieves the result of the cloaked query from the public server, which is a superset of the original query result. Finally, false hits are filtered by the anonymizer and the result of the actual query is returned to user. Spatial cloaking was first proposed in [8] and based on this idea, a system that supports different queries and various anonymization requirements was developed by Mokbel et al. in [15]. Ghinita et al. [7] showed how to apply this framework in a distributed environment, where users dynamically form and maintain the k-anonymized spatial regions.

A problem related to sequence anonymization has been addressed in [9]. Given a set of trajectories, the objective is to confuse an adversary who can associate trajectory parts to user identities (e.g., using a residential addresses database). This is achieved by perturbing trajectories at points where they spatially meet; for instance if the actual routes of two cars x and y cross at some point, then the segment of x (resp. y) before the meeting point is concatenated with the segment of y (resp. x) after the meeting point to form the published trajectories. An optimization problem is modeled, where the goal is to achieve the maximum privacy subject to a maximum allowed distance of trajectories at meeting points where perturbation takes place. On the other hand, quality constraints such as the difference between the original and published trajectories are not considered. In [3], the concept of historical k-anonymity is defined. Assuming that users issue sequences of spatial queries to a service provider, the objective is to generalize the locations of the queries, such that for each sequence of cloaked requests there are at least k users whose trajectories are covered by them. As a result, an adversary cannot link the sequence of requests to less than k users. In a recent study [16], the problem of perturbing the values of a time-series in order to introduce uncertainty has been studied. Noise added to the original data should have similar properties as the data and at the same time should be resistant to compressibility-based attacks. This work has similar objectives to our study, however, the data domain as well as the attack and utility models differ, since (i) we attempt to anonymize multiple sequences of discrete data (as opposed to adding noise in a continuous data domain), (ii) we assume that the adversary's knowledge (i.e., sequence parts) is known to the publisher, and (iii) suppression is used whenever necessary, as a means for protecting privacy.

3. Problem Definition

We model the original trajectories in the owner database as sequences of locations, which are precise points on a map. Unlike the common assumption in spatiotemporal data, we consider a discrete spatial domain, e.g., spatial information is given in terms of addresses in a city map. Such a case is closer to the reality for data that stem from commercial transactions. More formally, we define as trajectory

as follows:

Definition 1 *A trajectory t of length n is a sequence $t = [p_1, \ldots, p_n]$ of addresses taken from a set $\mathcal{P}$*

In our motivating example, $\mathcal{P}$ includes all addresses of shops which accept the Octopus cards. For simplicity, we assume that there is a 1-1 relationship between shops and addresses. Since commercial companies might have multiple branches, $\mathcal{P}$ can be partitioned in m disjoint non-empty sets of addresses $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_m$, such that each set contains all and only the addresses of the different branches of a company. Therefore, assuming that we wish to publish a database T of trajectories, where sequence elements take values from $\mathcal{P}$, there are m adversaries (i.e., companies). Each adversary $i \in \mathcal{V}$ controls addresses $\mathcal{P}_i$, such that $\forall i, j \in \mathcal{V}, \mathcal{P}_i \cap \mathcal{P}_j = \varnothing$ and $\bigcup_{i \in \mathcal{V}} \mathcal{P}_i = \mathcal{P}$. For each trajectory $t \in T$, every adversary $i \in \mathcal{V}$ holds a projection t^i, as defined below.

Definition 2 *A projection of a trajectory $t = [p_1, \ldots, p_n]$ with respect to an adversary A, who owns $\mathcal{P}_A \subset \mathcal{P}$ is a (potentially empty) trajectory $t^A = [p_1^A, \ldots, p_k^A]$, $\{p_j^A \mid p_j^A \in t \wedge p_j^A \in \mathcal{P}_A\}$. The order of elements $p_j^A \in t$ is preserved in t^A.*

Simply speaking, the projection t^A of a $t \in T$ is the subtrajectory of t that contains only and all the points of $\mathcal{P}_A$ in t. Therefore, each adversary (e.g., A) holds a local database (e.g., T_A) with the projections of all $t \in T$ with respect to $\mathcal{P}_A$. The adversary has no knowledge about trajectories having empty projection; therefore, T_A can be smaller than the database of the publisher. A trajectory may appear multiple times in T and more than one trajectories may have the same projection with respect to $\mathcal{P}_A$. The most important property of a t^A is that adversary A can directly link it to the identities of all persons that pass through it, in its local database (e.g., loyalty program). We are now ready to formalize the definition of our problem.

Definition 3 (Problem Definition) *Given a database T of trajectories, where locations take values from $\mathcal{P}$, construct a transformed database T', such that if T' is public, for all $t \in T$, every adversary A cannot correctly infer any location $\{p_j \mid p_j \in t \wedge p_j \notin t^A\}$ with probability larger than P_{br}.*

In simple words, we do not want an adversary A who has a projection t^A of a trajectory t in the original database to correctly infer any other locations of the trajectory after accessing T'. This problem is similar to the ℓ-diversity problem defined in [13]. In our case, t^A acts like a quasi-identifier that links the remaining points to a person identity in A's local database. The main differences from database publication problems studied before are (i) the quasi-identifiers are variable-length sequences of locations and (ii) there can be multiple sensitive values (i.e., locations) per trajectory in T and these values are different from the perspectives of different adversaries (i.e., shop chains). The second difference requires the algorithm which transforms T to T' to consider attacks to different sensitive values from different adversaries.

3.1. Computation of Threats

Before we describe the derivation of privacy threats, we define the key concept of *support* that is used by our methodology. We say that trajectory $t \in T$ supports a projection t^A, held by adversary A, if the projection of t with respect to $\mathcal{P}_A$ is identical to t^A. For example, consider a database T and two adversaries A and B, that divide $\mathcal{P}$ in two sets of locations (i.e., shop branches) $\mathcal{P}_A = \{a_1, a_2, \ldots\}$ and $\mathcal{P}_B = \{b_1, b_2, \ldots\}$. Trajectory $t = [a_1, b_1, a_2]$ supports projection $[a_1, a_2]$, but does not support $[a_1]$, $[a_1, a_2, a_3]$, or $[a_2, a_1]$. Reversely, the support set $S(t^A, T)$ of a projection t^A with respect to T is defined by the set of trajectories in T that support it.

Let us now elaborate on the computation of the probability $P(p_j, t^A, T')$ for adversary A to associate a location $p_j \notin \mathcal{P}_A$ to a real person, whose trajectory is projected as t^A in T_A. Assume that t^A is supported by a set $S(t^A, T')$ of trajectories in the published database T'. Then, $P(p_j, t^A, T')$ can be computed as follows.

$$P(p_j, t^A, T') = \frac{\left|\{t' \mid t' \in S(t^A, T') \wedge p_j \in t'\}\right|}{|S(t^A, T')|} \quad (1)$$

In simple words, $P(p_j, t^A, T')$ is the fraction of trajectories in the support $S(t^A, T')$ which include p_j. If $P(p_j, t^A, T') > P_{br}$, then we say that the publication of T' is insecure with respect to t^A and p_j. For example, in Figure 1, $P(b_2, t^7, T) = 66\%$ and pair (b_2, t^7) violates $P_{br} = 50\%$ if T is published exactly. However, if T is published as T' then $P(b_2, t^7, T') = 50\%$ and there is no privacy breach. Our goal is to derive a transformation T' for which there exists no (p_j, t^A) pair for which $P(p_j, t^A, T') > P_{br}$.

3.2. Utility

The recipient of the transformed database T' may be interested in analyzing it in various ways. For example he could apply some data mining operation (like clustering [10] or sequence pattern mining [5]) to the resulting trajectories, or perform aggregate queries of the form "How many people traveled from region X to region Y and then to region Z?". Given that different recipients might be interested in different properties of the data, we do not make explicit assumptions on their use, but use a general measure for the quality of the published dataset T'. We measure utility by the average difference between the original trajectories in T and the published ones in T'.

The difference $diff(t, t')$ between a $t \in T$ and its corresponding transformation $t' \in T'$ is measured as fol-

lows. Let t'_s and t'_e be the starting and ending points of t', respectively, corresponding to t_s and t_e in t, respectively. Each point $p \in t$, contributes one component to the distance. If p is before t_s (after t_e), then the corresponding component is $dist(p, t_s)$ ($dist(p, t_e)$, respectively). If p is in-between t_s and t_e then the error for p is $dist(p, (\overline{p, t'}))$, where $\overline{p, t'}$ is the projection of p on trajectory t'. By summing the squares of these components and taking the root of the sum we derive the difference between the trajectories. Figure 2 shows an exemplary trajectory $t = a_2 \rightarrow a_1 \rightarrow a_3 \rightarrow b_1$, which is approximated by $t' = a_1 \rightarrow b_1$ (after the suppression of a_2 and a_3). The difference between t and t' is computed by summing the squares of components $dist(a_2, a_1)$, $dist(a_1, a_1)$, $dist(a_3, (\overline{a_3, t'}))$, and $dist(b_1, b_1)$, and then taking squared root, i.e., $diff(t, t')$=$\sqrt{dist(a_2, a_1)^2 + dist(a_3, (\overline{a_3, t'}))^2}$. In the special case, where $t' = \emptyset$ (i.e., the original t has been completely suppressed in T'), each component of $diff(t, t')$ is set as the maximum distance between two points on the map.

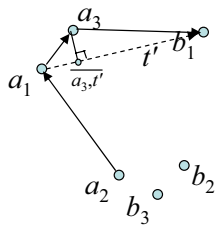


Figure 2. Distance between trajectories

4. The Algorithm

The anonymization algorithm carries the basic concept of value generalization to the spatiotemporal field. In our context, the quasi-identifiers are projections of trajectories held by the attackers. The main idea behind our anonymizing algorithm is to transform long and detailed projections to smaller and simpler ones. In doing so, we are able to (i) increase the supports of projections and (ii) diversify the locations that are not monitored by adversaries, making thus impossible for them to infer with high certainty whether a trajectory includes such a point. The mechanism we use for generalizing the trajectories is to suppress the existence of certain points in them, taking under consideration the benefit in terms of privacy and the deviation from the main direction of the trajectory.

Finding the optimal set of points to delete from T in order to derive a secure T' and achieve the minimum possible information loss is harder than the simpler k-anonymity problem in relational databases, discussed in Section 2, which is already shown to be NP-hard [14]. Therefore, we propose a greedy algorithm that iteratively suppresses locations, until the privacy constraint is met. The algorithm simulates the attack from any possible adversary, and then solves the identified privacy breaches. Algorithm 1 is a pseudocode of this method.

In the first stage, the projected database T_i of each attacker $i \in \mathcal{V}$ is extracted. Then, the algorithm identifies the projections that lead to a privacy breach, by scanning T once, according to Algorithm 2. For each tuple $t \in T$, and for each adversary i, such that projection t^i is not empty, a counter $sup(p_j, t^i, T)$ is increased for each $p_j \in t, p_j \notin t^i$. After scanning T, pairs (p_j, t^i) for which $P(p_j, t^i, T) = \frac{sup(p_j, t^i, T)}{S(t^i, T)} > P_{br}$ are identified and t^i is marked as a problematic projection for adversary i. Then Algorithm 1 runs a loop; while privacy breach problems have been identified, we attempt to unify a pair of projections (t^i_x, t^i_y), of the same adversary i, at least one of which is problematic.

Two projections can be unified only if one is a subtrajectory of the other, i.e., the larger projection contains all the points of the smaller one, and in the right order. For example, $a_1 \rightarrow a_3$ can be unified with a_3 (see t^A_5 and t^A_6 in Figure 1). Technically, the unification results in the suppression of the points (e.g., a_1) in the longest projection (e.g., $a_1 \rightarrow a_3$) that are not contained in the shorter one (e.g., a_3) in all trajectories that support the former (e.g., t_5). The unification is done in a way such that the resulting projection (e.g., a_3) does not violate the privacy of the data. In other words, if t^i_x is a problematic projection then either t^i_x is not supported in the transformed database T' resulting from this unification, or $P(p_j, t^i_x, T') \leq P_{br}$ for all $p_j \notin \mathcal{P}_i$. In the example of Figure 1, after the unification of $a_1 \rightarrow a_3$ with a_3, trajectory t_5 becomes $t'_5 = a_3 \rightarrow b_1$ and the problems of both $a_1 \rightarrow a_3$ and a_3 are fixed; $a_1 \rightarrow a_3$ is no longer supported in T' and a_3 does not map to any B-location with probability higher than 50%.

The unification happens only between the projections of the same attacker. Still, the consequences of the unification are not limited to that attacker, as problems that other adversaries have may be solved. The effect here is different; instead of increasing the supports of projections in order to decrease the transition probabilities to sensitive locations, the instances of some points that do not belong to the projection are reduced, and so is the adversary's certainty that a trajectory includes any of these points. For example, by removing a_1 from t_5 in Figure 1, the problems of $a_1 \rightarrow a_3$ and a_3 are resolved and, at the same time, a_1 appears fewer times in the trajectories where projection b_1 of adversary B is mapped to. To this end, we re-assess the existing problems after performing the unification and, while there are more breaches, we repeat the unification process.

Since it is likely that more than one projections are problematic, at each loop we choose the one, which is speculated

to be the most beneficiary. Therefore, each possible unification is given a *cost* with respect to the information loss it entails. This cost is quantized by summing the distances of the transformed trajectories to the corresponding original ones, had the unification been committed, minus the corresponding cost before the unification. Distances between original and transformed trajectories are measured using the method described in Section 3.2.

To facilitate the efficient execution of our algorithm we keep explicit links from each trajectory to the supported projections and vice-versa. The following theorem ensures that the algorithm terminates to a secure publication of T.

Theorem 1 *Algorithm 1 derives a secure database T'.*

Proof. First, Algorithm 2 correctly identifies the privacy breaches, by directly applying Equation 1 in the supporting trajectories to the projections of each adversary. Each time unification is performed between two trajectories t_x^i and t_y^i, where $t_y^i \subset t_x^i$, the breach for at least one of them is resolved. On the other hand, there is no way that after a unification, some $P(p_j, t^i, T')$ will become higher than P_{br} since points are only removed from trajectories and a unification that would result in a breach is never committed. The removal of points of an attacker A, can cause only the reduction of the numerator in Eq. 1 for any other attacker B, thus the P_{br} of any attacker B can only be reduced when removing points from an attacker A. In addition, a unification of a projection with the empty projection is possible (if it is unavoidable), so the algorithm will always terminate to a secure publication. □

A subtle thing to note is that after the unification of t_x^i with t_y^i, since t_x^i is no longer supported in T', the adversary i can infer that t_x^i is unified with t_y^i, and implicitly unify the supports of the two. However, this is not an issue for t_x^i, because the unification resolves the privacy problem of t_y^i; in other words, since no point $p_j \notin \mathcal{P}_i$ violates P_{br} for projection t_y^i it will also not violate P_{br} for t_x^i, which is supported by the same trajectory set $S(t_y^i, T')$. For example, unifying $a_1 \rightarrow a_3$ with a_3 in the example of Figure 1, fixes the problem of a_3, and therefore the problem of $a_1 \rightarrow a_3$, if the adversary assumes that the latter is mapped to the same set of tuples (i.e., t'_5 to t'_8).

4.1. Multiple Unifications per Loop

In order to improve the efficiency of our algorithm, we perform multiple unifications at each loop of Algorithm 1. Since not all projections and their problems are affected by a single unification, it makes sense to perform many independent ones in parallel. That is, each projection participates in at most one unification pair. At each loop, we select the top-s, in terms of cost, independent unifications that resolve problems and apply them simultaneously. Intuitively, this will decrease the cost of the algorithm proportionally. On the other hand, some of these unifications could be avoided if they were performed one at a time, since a unification $U(t_x^i, t_y^i)$, related to attacker i, affects problematic projections of adversaries j, where $j \neq i$.

Algorithm 1 Anonymization algorithm

Anonymization(T, P_{br})
1: Construct projection DB T_i for each attacker $i \in \mathcal{V}$
2: *Breach_identification*(T, P_{br}) ▷ call Alg. 2
3: T':=T ▷ Initialize output database
4: **while** there are still privacy breaches **do**
5: $U(t_x^i, t_y^i)$:=lowest cost unification ($t_y^i \subset t_x^i$, t_x^i and/or t_y^i are/is problematic)
6: **for all** $t' \in S(t_x^i, T')$ **do** ▷ traj. in T' supporting t_x^i
7: Suppress all $p \in t'$, $p \in t_x^i, p \notin t_y^i$
8: remove t_x^i from T_i ▷ t_x^i not supported anymore in T'
9: *Breach_identification*(T', P_{br})

Algorithm 2 Identification of privacy breaches

Breach_identification(T, P_{br})
1: **for all** i in $\mathcal{V}$ **do** ▷ Initialization
2: **for all** $p_j \notin \mathcal{P}_i$ **do**
3: **for all** $t^i \in T_i$ **do**
4: $sup(p_j, t^i, T)$:=0;
5: **for all** t in T **do** ▷ DB scan to update counters
6: **for all** i in $\mathcal{V}$ **do**
7: t^i:=projection of t in T_i
8: **if** $t^i \neq \emptyset$ **then**
9: **for all** p_j, $p_j \in t, p_j \notin \mathcal{P}_i$ **do**
10: $sup(p_j, t^i, T)$++;
11: **for all** i in $\mathcal{V}$ **do** ▷ Initialization
12: **for all** $p_j \notin \mathcal{P}_i$ **do**
13: **for all** $t^i \in T_i$ **do**
14: **if** $\frac{sup(p_j, t^i, T)}{S(t^i, T)} > P_{br}$ **then**
15: Mark (p_j, t^i) as problematic

4.2. Extensibility

The trajectories have been regarded so far as simple sequences of locations that do not include timestamps in their elements. In practice, the publisher may also want to include timestamps in the published information. Our model can easily incorporate this extension, as follows. The temporal domain is discretized into intervals (e.g., hour of the day) and the locations in the sequences are replaced by (location, timestamp) pairs. For example, sequence $a_1 \rightarrow b_1 \rightarrow a_2$ is replaced by $(a_1, 10-11a.m.) \rightarrow (b_1, 1-2p.m.) \rightarrow (a_2, 3-4p.m.)$. The projections are also sequences of composite elements. Our algorithm can be seamlessly applied for such data, since the set of trajectories that support a specific timestamped projection and the corresponding threats can be identified as described in Section 3, the only difference being the replacement of locations by the composite elements. For example, projec-

tions $(a_1, 10-11a.m.) \rightarrow (a_2, 3-4p.m.)$ and $(a_1, 10-11a.m.) \rightarrow (a_2, 5-6p.m.)$ are considered different in the extended model, while they are identical (i.e., $a_1 \rightarrow a_2$) in the basic one. The discretization level of the temporal domain is determined based on the number of events that happen at an interval (very fine granularity makes privacy breaches easier and affects the utility of the published data).

5. Experimental Evaluation

We conducted a set of experiments to validate the effectiveness of our anonymization technique. For this purpose, we generated synthetic trajectories of moving objects using Brinkhoff's generator [4]. After generating routes on the Oldenburg map and normalizing them so that all coordinates take values in a $[1, 1000]^2$ map, we simulated sequences of customer transactions corresponding to these routes as follows. The map was divided into 100 regions using a uniform grid, and each region was given an adversary-id in random. An object that visits a sequence of regions is assumed to have performed one transaction per region in the visiting order. The centroids of these regions model the locations in the trajectories of T. The average length of the generated trajectories is 4.5. The default number of regions (i.e., addresses) in $\mathcal{P}$, trajectories in T, and adversaries in $\mathcal{V}$ are 100, 10000, and 5, respectively. The default number of unifications performed in parallel at each loop of Algorithm 1 is $s = 50$. The breach probability P_{br} is set to 50%, by default.

In the first experiment (Table 1), we vary the number s of concurrent unifications per loop and observe the utility (i.e., cost) C of the resulting database T', the number L of locations suppressed from the trajectories in T, the running time of anonymization (in seconds), and the total number U of committed unifications. Table 1 shows the result. As s increases, the time to anonymize the database drops fast. On the other hand, the information loss (modeled by C) increases with s, but not as significantly compared to the decrease in the execution time. The independent unifications, performed in parallel, do not have great effect on the quality (but result in great improvement in terms of speed). The fluctuations in the quality are due to the approximate nature of our greedy search; some unification decisions that are not deemed the most appropriate when fewer unifications are done at each loop, may eventually prove beneficial. In terms of deleted points (L) and total number of unifications (U), we do not see great difference as s increases. In general, the increase of s causes more points to be deleted from the database, but fewer unifications. This is because, with large s, the unifications performed should be independent and due to the lack of choices the algorithm selects many projections to completely suppress (i.e., unify them with the empty projection). The result of this is that fewer problems remain in latter iterations (smaller number of total unifications and removal in projection points), but the total number of deleted points from the database may increase due to the high supports of these suppressed projections. In summary, large values of s pay off as they reduce the anonymization time significantly, without high effect on the quality.

s	C	L	$time$	U
1	200	18564	530.74	1563
5	218	18916	107.58	1563
20	244	19726	26.55	1565
50	287	20664	12.59	1563
100	274	21415	6.36	1491

Table 1. Experiments with variable s

$\|T\|$	C	L	$time$	U
2000	303	4950	1.34	691
5000	296	11805	4.141	1075
10000	286	20664	12.59	1563
15000	250	29998	20.81	1891

Table 2. Experiments with variable $|T|$

In the second experiment (Table 2) we evaluate the scalability and robustness of our technique for different database sizes. Note that the time increases proportionally with the database size. On the other hand, the percentage of points that are deleted from the database decreases. The reason is that as T increases the initial supports of the projections increase in size (i.e., the probability that two trajectories have the same projection increases). This reduces the probability that a projection is problematic and makes the anonymization problem easier. The quality, increases with T, as fewer points are deleted from the trajectories.

Next, we test the performance of our algorithm varying the number $|\mathcal{V}|$ of adversaries (Table 3) . If the locations are distributed to a small number of adversaries, then the anonymization time increases, as the projections at each adversary are many and long and there are many privacy breaches. This affects the number of unifications and the running time, accordingly. On the other hand, the number of points deleted from the database is not very sensitive to $|\mathcal{V}|$. As $|\mathcal{V}|$ increases, the supports of the projections in T decrease, because there is a larger number of location permutations (distinct projections) that correspond to each attacker, which appear rarely in T and cause more breaches. Therefore, each unification causes few changes to T (i.e., few point deletions). Reversely, if there are many adversaries, the number of projections per adversary is small, but the support for each projection is large. Therefore, a unification results in the deletion of many points from T. The information loss is low for very few attackers because there are more projections per attacker and more flexibility in choosing a potential pair to unify. As a result, the choices made are good in terms of quality. As $|\mathcal{V}|$ increases,

$\|\mathcal{V}\|$	C	L	$time$	U
2	163	18333	511.36	4219
3	170	20540	65.81	2966
4	219	23369	18.14	2072
5	287	20664	12.59	1563
6	240	23620	6.55	1240
7	241	23617	4.30	986

Table 3. Experiments with variable $|\mathcal{V}|$

P_{br}	C	L	$time$	U
30%	1003	38150	9.82	1844
40%	466	28250	9.79	1729
50%	287	20664	12.59	1563
60%	170	15578	12.13	1530
70%	89	10691	9.95	1456

Table 4. Experiments with variable P_{br}

the quality gradually drops and then increases again, since if there are many attackers, the chances of finding problematic projections decrease.

Finally, we ran the algorithm for various values of P_{br} (Table 4). As expected, with the increase of P_{br} fewer points are deleted and the quality of the published database improves. The CPU time is not affected by this parameter.

Summing up, our technique performs best if the number of unifications s per iteration is large, the database size $|T|$ is large and the points are distributed to many adversaries. The first condition is a tunable parameter of our algorithm, while the other two happen in practice; typically, databases are large and the vendors who offer electronic payment in transactions are many. Therefore, our technique is expected to be practical in real problems.

6. Conclusion

This is, to our knowledge, the first study of an interesting problem; how to transform a database of trajectories to a format that would prevent adversaries, which hold a projection of the data, from inferring locations missing in their projections with high certainty. In practice, these missing locations could be associated to a real person and threaten his/her privacy. We propose a technique that iteratively suppresses selected locations from the original trajectories until a privacy constraint is satisfied. The effectiveness of our method is validated by experimentation.

In the future, we plan to study alternative sequence transformation techniques to solve the same problem. For example, instead of suppressing locations from the trajectories, we could replace them with larger (generalized) spatial regions that contain more addresses in them. Other techniques to be studied include sequence perturbation and inserting fake trajectories in the database.

References

[1] G. Aggarwal, T. Feder, K. Kenthapadi, S. Khuller, R. Panigrahy, D. Thomas, and A. Zhu. Achieving anonymity via clustering. In *PODS*, pages 153–162, 2006.

[2] R. J. Bayardo Jr. and R. Agrawal. Data privacy through optimal k-anonymization. In *ICDE*, pages 217–228, 2005.

[3] C. Bettini, X. S. Wang, and S. Jajodia. Protecting privacy against location-based personal identification. In *Secure Data Management*, pages 185–199, 2005.

[4] T. Brinkhoff. A framework for generating network-based moving objects. *GeoInformatica*, 6(2):153–180, 2002.

[5] H. Cao, N. Mamoulis, and D. W. Cheung. Mining frequent spatio-temporal sequential patterns. In *ICDM*, pages 82–89, 2005.

[6] A. Froomkin. The death of privacy. *Stanford Law Review*, 52(5):1461–1543, 2000.

[7] G. Ghinita, P. Kalnis, and S. Skiadopoulos. Prive: anonymous location-based queries in distributed mobile systems. In *WWW*, pages 371–380, 2007.

[8] M. Gruteser and D. Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *MobiSys*, 2003.

[9] B. Hoh and M. Gruteser. Protecting location privacy through path confusion. In *SecureComm*, 2005.

[10] J.-G. Lee, J. Han, and K.-Y. Whang. Trajectory clustering: a partition-and-group framework. In *SIGMOD Conference*, pages 593–604, 2007.

[11] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Incognito: Efficient full-domain k-anonymity. In *SIGMOD Conference*, pages 49–60, 2005.

[12] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Mondrian multidimensional k-anonymity. In *ICDE*, page 25, 2006.

[13] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. In *ICDE*, page 24, 2006.

[14] A. Meyerson and R. Williams. On the complexity of optimal k-anonymity. In *PODS conference*, 2004.

[15] M. F. Mokbel, C.-Y. Chow, and W. G. Aref. The new casper: Query processing for location services without compromising privacy. In *VLDB*, pages 763–774, 2006.

[16] S. Papadimitriou, F. Li, G. Kollios, and P. S. Yu. Time series compressibility and privacy. In *VLDB*, pages 459–470, 2007.

[17] P. Samarati. Protecting respondents' identities in microdata release. *IEEE Trans. Knowl. Data Eng.*, 13(6):1010–1027, 2001.

[18] P. Samarati and L. Sweeney. Generalizing data to provide anonymity when disclosing information (abstract). In *PODS*, page 188, 1998.

[19] L. Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5):557–570, 2002.

[20] J. Xu, W. Wang, J. Pei, X. Wang, B. Shi, and A. W.-C. Fu. Utility-based anonymization using local recoding. In *KDD*, pages 785–790, 2006.