Local Suppression and Splitting Techniques for Privacy Preserving Publication of Trajectories

Manolis Terrovitis, Giorgos Poulis, Nikos Mamoulis and Spiros Skiadopoulos
mter@imis.athena-innovation.gr, {poulis, spiros}@uop.gr, nikos@cs.uoi.gr

Abstract—We study the problem of preserving user 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 cards, RFID debit cards or NFC compliant devices. We show that, if such trajectories are published exactly (by only hiding the identities of persons that followed them), one can use partial trajectory knowledge as a quasi-identifier for the remaining locations in the sequence. We devise four intuitive techniques, based on combinations of locations suppression and trajectories splitting, and we show that they can prevent privacy breaches while keeping published data accurate for aggregate query answering and frequent subsets data mining.

Index Terms—Anonymity, Trajectories, Generalization, Suppression, Splitting.

1 Introduction

Consider a company *SmartCard* (e.g, a contactless card company, a bank) that issues electronic money cards (like the Octopus or the EZ-Link card) or credit/debit cards (e.g., VISA, MASTERCARD) to its users, to allow cashless payments. Assume that the company wants to publish sequences of transactions for analysis and querying purposes. We consider as a motivating example, the Octopus (http://www.octopuscards.com/) smart RFID card, commonly used by Hong Kong residents to pay for their transportation and for their transactions at PoS services (e.g., shops, vending machines). Transactions created by the usage of cards in physical locations have a very important spatiotemporal aspect; they reveal that a person was at a certain place in a specific time. *SmartCard* and similar companies accumulate vast amounts of transactional data that reveal user trajectories daily, and they are under pressure to publish or share them with third parties. Such data can be used to extract movement and behavioral patterns as well as the causality relationships between geographic locations. A similar problem can arise in the context of Location Based Social Networks, where the user trajectory is the sequence of her check-ins in predefined locations.

Direct publishing of this information, even after hiding the IDs of users, may easily result in privacy breaches, once combined with the partial trajectory information, known to an adversary. The adversaries in the aforementioned setting, are the companies that collaborate with *SmartCard*. Any company that accepts the cards of *SmartCard* has a small part of *SmartCard*'s database, i.e., it has partial knowledge of numerous trajectories. When a person uses his card to pay at a convenience store, the company that owns 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). Large companies that have chains of stores constitute powerful adversaries, since they might know more than one points of a published trajectory. Such companies are able to launch mass attacks against user privacy, i.e., they can re-identify users at a massive scale, by simply matching their own database to *SmartCard*'s database if that is available to them. At the same time, in the above setting, the data owner (i.e., *SmartCard*) knows the part of the database that each adversary has and can use this knowledge to anonymize the trajectory data before publication. This is exactly the focus of our proposed anonymization methodology.

Fig. 1a shows an example of a database $\mathcal{T}$ owned by *SmartCard*. For the ease of presentation, we model each trajectory by a sequence of locations, without temporal information (i.e., timestamps) on the elements. Each sequence element is a shop address, where the corresponding user had 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 and possible adversary A. Fig. 1b shows $\mathcal{T}^A$; the knowledge A has for the exact database of Fig. 1a. It is easy to see, that by matching this background knowledge about its customers from $\mathcal{T}^A$ to the published knowledge of $\mathcal{T}$, the adversary can infer additional locations visited by its customers. For instance, the adversary may link trajectories t_1 and t_1^A since they are the only ones in $\mathcal{T}$ and $\mathcal{T}^A$ respectively that contain a_1 and do not contain a_2 and a_3 and, thus, deduce that the user of t_1^A also visited locations b_2 and b_3.

In this paper, we propose a methodology that allows anonymizing trajectories, in a way that would prevent adversaries from using their partial knowledge to infer locations unknown to them. Our proposal adopts the l-diversity paradigm, to prevent adversaries from inferring the connection between their background knowledge about a person's whereabouts and specific locations. Our methods employ both suppression and splitting of trajectories, to sanitize the data in order to reduce the information loss. The

- *M. Terrovitis is with the IMIS Athena research center.*
- *G. Poulis and S. Skiadopoulos are with the University of the Peloponnese.*
- *N. Mamoulis is with the University of Ioannina.*

G. Poulis research is funded by Research Funding Program: Heraclitus II
N. Mamoulis research is funded by EU H2020 grant 657347 (LBSKQ)

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

(a) Original dataset $\mathcal{T}$

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

(b) A's knowledge $\mathcal{T}^A$

id	trajectory
t'_1	$a_1 \to b_2 \to b_3$
t'_2	$b_1 \to a_3$
t'_3	a_3
t'_4	$a_3 \to b_1$
t'_5	$a_1 \to b_1$
t'_6	$a_1 \to b_1$
t'_7	a_1
t'_8	$a_3 \to b_2 \to b_3$

(c) GSUP($\mathcal{T}$): Global suppression

id	trajectory
t'_1	$b_2 \to b_3$
t'_2	$a_2 \to a_3$
t'_3	$a_2 \to a_3$
t'_4	$a_2 \to a_3$
t'_5	$a_3 \to a_1$
t'_6	$a_3 \to a_1$
t'_7	$a_3 \to a_1$
t'_8	$b_2 \to b_3$

(d) LSUP($\mathcal{T}$): Local suppression

id	trajectory	id	trajectory
t'_1	$a_1 \to b_2 \to b_3$	t''_4	b_1
t'_2	b_1	t'_5	a_3
t''_2	$a_2 \to b_2$	t''_5	$a_1 \to b_1$
t'''_2	a_3	t'_6	$a_3 \to a_1$
t'_3	a_2	t''_6	b_1
t''_3	b_3	t'_7	$a_3 \to b_2 \to a_1$
t'''_3	a_3	t'_8	$a_3 \to b_2 \to b_3$
t'_4	$a_2 \to a_3$		

(e) SPLIT($\mathcal{T}$): The splitting method

id	trajectory	id	trajectory
t'_1	$a_1 \to b_2 \to b_3$	t''_4	b_1
t'_2	b_1	t'_5	a_3
t''_2	a_2	t''_5	$a_1 \to b_1$
t'''_2	b_2	t'_6	$a_3 \to a_1$
t''''_2	a_3	t'_7	$a_3 \to b_2 \to a_1$
t'_3	$a_2 \to a_3$	t'_8	$a_3 \to b_2 \to b_3$
t'_4	$a_2 \to a_3$		

(f) MIX($\mathcal{T}$): Mixing suppression and splitting

Fig. 1: The original dataset, an adversary's knowledge and the result of several anonymization methods

application of trajectory splitting is a novel data transformation that allows our method to handle large trajectories.

Our work makes the following contributions:

- We present a novel data transformation for trajectories, termed splitting, which splits original trajectories to new smaller ones.
- We develop four anonymization algorithms, which employ suppression of locations, splitting of trajectories or both, to assess the impact of each strategy.
- We experimentally demonstrate that our algorithms protects data, preserve data utility and are efficient.

This work extends the work of [**?**] by: *(a)* an improved global suppression algorithm, coined GSUP, *(b)* a local suppression algorithm, coined LSUP, *(c)* the trajectory splitting transformation and the two algorithms, coined SPLIT and MIX that exploit it.

2 PRELIMINARIES

In this section, we first define the concepts and the notation used throughout the paper. Then, we present an algorithm that identifies privacy threats according to our problem definition. Finally, we outline the anonymization techniques used by our approach and discuss its limitations.

Definition 1. Let $\mathcal{L}$ be a set of locations. A *trajectory* t of length n, is a sequence $t = l_1 \to \cdots \to l_n$ of n locations of $\mathcal{L}$, tracking the visits of a user. The size of a trajectory is denoted by $|t|$.

A *location* is a point of special interest on a map (e.g., hospital, store, bank, touristic site). Fig. 1a shows eight trajectories defined on locations $\mathcal{L} = \{a_1, a_2, a_3, b_1, b_2, b_3\}$.

Definition 2. Let $t = l_1 \to \cdots \to l_n$ and $s = \ell_1 \to \cdots \to \ell_\nu$ be two trajectories defined over locations set $\mathcal{L}$. Trajectory s is a *subtrajectory of* or is *contained in* trajectory t, denoted by $s \sqsubset t$, if and only if $|s| < |t|$ and there is a mapping f such that $\ell_1 = l_{f(1)}, \ldots, \ell_\nu = l_{f(\nu)}$ and $f(1) < \cdots < f(\nu)$ hold.

For example, trajectories a_1, b_2, $a_1 \to b_2$ and $a_1 \to b_3$ are some subtrajectories of trajectory $t_1 = a_1 \to b_2 \to b_3$.

Definition 3. Let $\mathcal{T}$ be a set of trajectories defined over locations set $\mathcal{L}$. An *adversary* A is a data owner (e.g., a store chain) controlling a subset of locations $\mathcal{L}^A \subset \mathcal{L}$. Adversary A can track any user visiting a location in $\mathcal{L}^A$.

For a set of potential adversaries $\mathcal{A}$, we assume that each adversary $A \in \mathcal{A}$ controls a non-empty set of locations $\mathcal{L}^A$ that is disjoint from the set of locations $\mathcal{L}^B$ controlled by any other adversary $B \in \mathcal{A}$. More formally, for any $A, B \in \mathcal{A}$ such that $A \neq B$ we have $\mathcal{L}^A \cap \mathcal{L}^B = \emptyset$. In Fig. 1a there are two adversaries A and B controlling locations $\mathcal{L}^A = \{a_1, a_2, a_3\}$ and $\mathcal{L}^B = \{b_1, b_2, b_3\}$ respectively.

Definition 4. Let $\mathcal{L}$ be a set of locations and A be an adversary controlling locations $\mathcal{L}^A \subset \mathcal{L}$. The *projection* of a trajectory $t = l_1 \to \cdots \to l_n$ to adversary A, denoted by $\pi_A(t)$, is a (potentially empty) subtrajectory of t, describing A's knowledge about trajectory t. More formally, $\pi_A(t) = \lambda_1 \to \cdots \to \lambda_\nu$ where $\pi_A(t) \sqsubset t$ and $\lambda_1, \ldots, \lambda_\nu$ are all locations in $l_1, \ldots, l_n$ that belong to $\mathcal{L}^A$.

In simple words, to construct $\pi_A(t)$ we remove from the original trajectory $t = l_1 \to \cdots \to l_n$ all locations $l_i \notin \mathcal{L}^A$ and preserve the order of the remaining locations. For example in Fig. 1a, the projection of trajectory $t_1 = a_1 \to b_2 \to b_3$ to adversary A (respectively, B) is $\pi_A(t_1) = a_1$ (respectively, $\pi_B(t_1) = b_2 \to b_3$).

Definition 5. Let $\mathcal{T}$ be a set of trajectories defined over locations $\mathcal{L}$ and A be an adversary controlling locations $\mathcal{L}^A$. The *knowledge of adversary* A, denoted by $\mathcal{T}^A$, is the projection of every trajectory $t \in \mathcal{T}$ to A, i.e., $\mathcal{T}^A = \{\pi_A(t) : t \in \mathcal{T}\}$.

For example, Fig. 1b illustrates adversary A's knowledge $\mathcal{T}^A$ controlling locations $\mathcal{L}^A = \{a_1, a_2, a_3\}$ for the original dataset $\mathcal{T}$ of Fig. 1a. The adversary A may explore $\mathcal{T}^A$ and the published dataset to associate locations unknown to A with the trajectories of $\mathcal{T}^A$. To this end, A will first associate

Algorithm: THREATID

Input: A dataset $\mathcal{T}$, identification probability threshold P_{br}
Output: List $\mathcal{Q}$ of problematic pairs.
The total number of problems N.
The number of problems $n@S_{\mathcal{T}}(\lambda, t^A)$ of every problematic pair (λ, t^A).

1 $\mathcal{Q} = \emptyset, N = 0, n@S_{\mathcal{T}} = ()$ `// Initialize` $\mathcal{Q}$, N `and` $n@S_{\mathcal{T}}$
`// Compute` $n@S_{\mathcal{T}}$
2 **for** *every* $t \in \mathcal{T}$ *and every* $A \in \mathcal{A}$ **do**
3 **if** $\pi_A(t) \neq \emptyset$ **then**
4 **for** *every* $\lambda \in t \setminus \{\pi_A(t)\}$ **do**
5 $n@S_{\mathcal{T}}(\lambda, \pi_A(t))$++
`// Privacy threat identification`
6 **for** *every* $A \in \mathcal{A}$ **do**
7 **for** *every* $\lambda \in \mathcal{L} \setminus \mathcal{L}^A$ *and every* $t^A \in \mathcal{T}^A$ **do**
8 **if** $\frac{n@S_{\mathcal{T}}(\lambda, t^A)}{|S_{\mathcal{T}}(t^A)|} > P_{br}$ **then** `//` (λ, t^A) `is problematic`
9 Insert pair (λ, t^A) to list $\mathcal{Q}$
10 $N = N + n@S_{\mathcal{T}}(\lambda, t^A)$
11 **return** $(\mathcal{Q}, N, n@S_{\mathcal{T}})$

the projections in $\mathcal{T}^A$ with the actual trajectories of the published dataset as in the following definition.

Definition 6. A trajectory t in a published dataset *supports a projection* t^A of an adversary A, if $t^A = \pi_A(t)$. Reversely, the *support set* of a projection t^A of A with respect to a dataset $\mathcal{T}$, denoted by $S_{\mathcal{T}}(t^A)$, is defined as $S_{\mathcal{T}}(t^A) = \{t \in \mathcal{T} : \pi_A(t) = t^A\}$.

If a dataset $\mathcal{T}$ is published directly (as in Fig. 1a), adversary A is able to infer that $S_{\mathcal{T}}(a_3 \rightarrow a_1) = \{t_5, t_6, t_7\}$, i.e., projection $a_3 \rightarrow a_1$ is one of the following trajectories *(a)* $t_5 = a_3 \rightarrow a_1 \rightarrow b_1$, *(b)* $t_6 = a_3 \rightarrow a_1 \rightarrow b_1$ or *(c)* $t_7 = a_3 \rightarrow b_2 \rightarrow a_1$. In two of the above three cases, A is able to associate the user of projection $a_3 \rightarrow a_1$ with location b_1 that is not controlled by A. This type of attack is captured by the following definition.

Definition 7. Let $\mathcal{T}$ be a set of trajectories, A be an adversary controlling locations $\mathcal{L}^A$ and t^A be a projection of A. The *number of trajectories in the support set* $S_{\mathcal{T}}(t^A)$ *that contain a location* $\lambda \notin \mathcal{L}^A$, denoted by $n@S_{\mathcal{T}}(\lambda, t^A)$, is defined as $n@S_{\mathcal{T}}(\lambda, t^A) = \big|\{s : s \in S_{\mathcal{T}}(t^A) \wedge \lambda \in s\}\big|$. Moreover, the *identification probability*, denoted by $P_{\mathcal{T}}(\lambda, t^A)$, is the probability of associating a location $\lambda \notin \mathcal{L}^A$ to the projection t^A and can be computed by:

$$P_{\mathcal{T}}(\lambda, t^A) = \frac{n@S_{\mathcal{T}}(\lambda, t^A)}{|S_{\mathcal{T}}(t^A)|} \quad (1)$$

In simple words, $P_{\mathcal{T}}(\lambda, t^A)$ is the fraction of trajectories in the support set (e.g., $S_{\mathcal{T}}(t^A)$) that include λ.

Definition 8. A pair (λ, t^A) is called *problematic* if $P_{\mathcal{T}}(\lambda, t^A) > P_{br}$, where P_{br} is a user defined *probability threshold*. The *number of problems* of a problematic pair (λ, t^A) in $\mathcal{T}$ is captured by $n@S_{\mathcal{T}}(\lambda, t^A)$, i.e., it equals the number of trajectories in the support $S_{\mathcal{T}}(t^A)$ that contain λ. A projection t^A is called *problematic* if it participates in a problematic pair (λ, t^A).

Example 1. Consider Fig. 1a and projection a_1 of A. We have $S_{\mathcal{T}}(a_1) = \{t_1\}$, $n@S_{\mathcal{T}}(b_2, a_1) = 1$ and $P_{\mathcal{T}}(b_2, a_1) = 1$. Thus, if $P_{br} = 0.5$, pair (b_2, a_1) is problematic and its number of problems is $n@S_{\mathcal{T}}(b_2, a_1) = 1$. Similarly, pair $(b_1, a_3 \rightarrow a_1)$ is also problematic since $P_{\mathcal{T}}(b_1, a_3 \rightarrow a_1) = 2/3 = 0.66$.

(λ, t^A)	$n@S_{\mathcal{T}}(\lambda, t^A)$	$\lvert S_{\mathcal{T}}(t^A)\rvert$	$P_{\mathcal{T}}(\lambda, t^A)$
(b_2, a_1)	1	1	1
(b_3, a_1)	1	1	1
$(a_1, b_2 \rightarrow b_3)$	1	2	0.5
$(b_1, a_2 \rightarrow a_3)$	2	3	0.66
$(b_2, a_2 \rightarrow a_3)$	1	3	0.33
$(a_2, b_1 \rightarrow b_2)$	1	1	1
$(a_3, b_1 \rightarrow b_2)$	1	1	1
$(b_3, a_2 \rightarrow a_3)$	1	3	0.33
(a_2, b_3)	1	1	1
(a_3, b_3)	1	1	1
(a_2, b_1)	1	3	0.33
(a_3, b_1)	3	3	1
$(b_1, a_3 \rightarrow a_1)$	2	3	0.66
(a_1, b_1)	1	3	0.33
$(b_2, a_3 \rightarrow a_1)$	1	3	0.33
(a_1, b_2)	1	1	1
(a_3, b_2)	1	1	1
(b_2, a_3)	1	1	1
(b_3, a_3)	1	1	1
$(a_3, b_2 \rightarrow b_3)$	1	2	0.5

Fig. 2: Algorithm THREATID in operation; the problematic pairs (set $\mathcal{Q}$) are highlighted in gray

Definition 9. A dataset $\mathcal{T}$ is called *unsafe*, if it has one or more problematic pairs; otherwise is called *safe*. The *total number of problems* of an unsafe dataset $\mathcal{T}$ is the sum of the number of problems of every problematic pair in $\mathcal{T}$.

Now, we are ready to define the problem under consideration.

Definition 10. Given a dataset $\mathcal{T}$ of trajectories, a user defined probability threshold P_{br} and a set of adversaries $\mathcal{A}$, construct a safe dataset corresponding to $\mathcal{T}$, with minimum information loss.

We discuss a heuristic function to estimate the information loss in Section 4 and we present a series of information loss metrics in Section 9.

2.1 Identifying privacy threats

In this section, we will present Algorithm THREATID that identifies potential privacy threats. This algorithm takes as input a dataset $\mathcal{T}$, the identification probability threshold P_{br} and the background knowledge of adversaries, and returns *(a)* a list $\mathcal{Q}$ holding all problematic pairs (λ, t^A) (where λ is a location and t^A is a projection of adversary A), *(b)* the total number of problems N in dataset $\mathcal{T}$ and *(c)* the number of problems $n@S_{\mathcal{T}}(\lambda, t^A)$ by every problematic pair (λ, t^A).

After initializations, THREATID computes $n@S_{\mathcal{T}}(\lambda, t^A)$ by scanning $\mathcal{T}$ once (Lines 2–5). Specifically, for each tuple $t \in \mathcal{T}$, and for each adversary A, such that projection $\pi_A(t)$ is not empty, $n@S_{\mathcal{T}}(\lambda, \pi_A(t))$ is increased for each $\lambda \in t \setminus \{\pi_A(t)\}$. Next (Lines 6–10), the algorithm identifies, i.e., pairs (λ, t^A) for which $P_{\mathcal{T}}(\lambda, t^A) = \frac{n@S_{\mathcal{T}}(\lambda, t^A)}{S_{\mathcal{T}}(t^A)} > P_{br}$ (Line 8), inserts them to $\mathcal{Q}$ (Line 9) and increases the number of problems in $\mathcal{T}$ by $n@S_{\mathcal{T}}(\lambda, t^A)$ (Line 10).

Example 2. Let us execute Algorithm THREATID with input the dataset $\mathcal{T}$ of Fig. 1 and $P_{br} = 0.5$. In Lines 2–5, the

algorithm first considers trajectory $t_1 = a_1 \rightarrow b_2 \rightarrow b_3$. For t_1, adversary A knows projection a_1, thus, locations b_2 and b_3 are the locations in t_1 that do not belong to $\mathcal{L}^A$, so $n@S_{\mathcal{T}}(b_2, a_1)$ and $n@S_{\mathcal{T}}(b_3, a_1)$ become 1. Similarly, for t_1, adversary B knows projection $b_2 \rightarrow b_3$, thus, location a_1 does not belong to $\mathcal{L}^B$, so $n@S_{\mathcal{T}}(a_1, b_2 \rightarrow b_3)$ becomes 1. Following, THREATID scans the remaining trajectories and updates $n@S_{\mathcal{T}}$ for all locations and adversary projections; the results are illustrated in column $n@S_{\mathcal{T}}(\lambda, t^A)$ of Fig. 2. Following, Lines 6–10 consider all pairs in turn to identify the problematic ones. For the first pair (b_2, a_1) we have

$$P_{\mathcal{T}}(b_2, a_1) = \frac{n@S_{\mathcal{T}}(b_2, a_1)}{|S_{\mathcal{T}}(a_1)|} = \frac{1}{1} = 1 > 0.5 = P_{br}$$

so, (b_2, a_1) is problematic (see also Example 1). Thus, it is inserted into $\mathcal{Q}$ and the total number of problems is increased by 1 ($= n@S_{\mathcal{T}}(b_2, a_1)$). Similarly all other pairs are checked. In Fig. 2, the problematic pairs (set $\mathcal{Q}$) are highlighted in gray. The total number of problems N is 17; this may be verified by summing up $n@S_{\mathcal{T}}$ values of all problematic (in gray) pairs.

2.2 Eliminating privacy threats

The main idea behind our anonymizing algorithms 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 if a trajectory includes such a point. More specifically, to make dataset $\mathcal{T}$ safe, we will eliminate problematic pairs (λ, t^A) by employing suppression or splitting. *Suppression* deletes a location $\lambda \in t$ from a trajectory t. For instance, trajectory $t = l_1 \rightarrow \cdots \rightarrow \kappa \rightarrow \lambda \rightarrow \mu \rightarrow \cdots \rightarrow l_n$ after suppressing location λ becomes $t' = l_1 \rightarrow \cdots \rightarrow \kappa \rightarrow \mu \rightarrow \cdots \rightarrow l_n$. On the other hand, *splitting* trajectory t at location λ, results in trajectories $t' = l_1 \rightarrow \cdots \rightarrow \lambda$ and $t'' = \mu \rightarrow \cdots \rightarrow l_n$. In both cases, before applying a change, we take under consideration the privacy gain and the information loss..

When we suppress a location λ or split at a location λ term $n@S_{\mathcal{T}}(\lambda, t^A)$ is affected directly while term $S_{\mathcal{T}}(t^A)$ indirectly. In our setting, each location has two conflicting roles. It can either act as a *quasi identifier* or as a *sensitive item* for each adversary. Thus, suppressing a location or splitting a trajectory at a location may eliminate some but may also create some other privacy threats. The latter may happen if the suppression or the splitting operations reissues a previously eliminated threat. For example, assume that $a_1 \rightarrow b_1$ is the unique trajectory that violated the P_{br} and the algorithm has already suppressed a_1. If, at a latter stage, it examines $a_1 \rightarrow a_2 \rightarrow b_1$ the algorithm can re-create it by suppressing a_2, if it is not checked.

2.3 Limitations

The proposed approach tackles the threats posed by adversaries who own a large portion of the published dataset, but at the same time they are known to the publisher. This is a practical scenario, but it also has limitations.

Ad-hoc adversaries and colluding adversaries. We assume that the background knowledge of the adversaries is known to the publisher. However, in some cases an adversary can hold an arbitrary subset of each trajectory. For example, a person physically followed by an adversary A can reveal to A a part of her trajectory, regardless the ownership of the PoS services the person uses. The proposed method does not protect against such ad-hoc adversaries. Similarly, the proposed method does not provide a privacy guarantee against colluding adversaries since it does not take into account their combined background knowledge. In order to apply our approach in the case of such collusions, the publisher should treat each disjoint group of colluding adversaries as a single adversary by merging all their data. Moreover, in the motivating setting, where card companies are data publishers and adversaries are commercial companies, collusion requires sharing of companies' customer data, which is not a common practice.

Timestamps and continuity. In this paper, we assume that timestamp information is removed from the trajectories and only sequences of locations are published. The proposed approach uses suppression and splitting as its main data transformation operators. Both operators are not suitable for trajectories of timestamped locations, since it is high likely that each timestamp will be unique and the algorithm will have to suppress it. If we use some simple generalization based preprocessing, i.e., we transform timestamps to timeslots, e.g., 22:23:45 will be reported as 22:00-0:00 timeslot, then each transaction can be treated efficiently by our algorithms as a spatiotemporal point in the trajectory. This is exactly the way we have treated the data from Gowalla network in Section 9. For example point a_1 might indicate a transaction at location a in timeslot 1 and point a_2 a transaction again in location a, but at timeslot 2.

The above modeling of timestamps, together with the model of predefined locations (which is natural for transaction points) does not lead to trajectories, which are continuous routes, but to semantic trajectories, which are sequences of discrete points. This model offers protection against minimality attacks by adversaries who could observe gaps due to suppression of points. While for continuous trajectories (e.g., trajectories created by GPS tracing) suppressing certain points, would still allow for accurate assumptions by adversaries due to the continuity of measurements, this is impractical for trajectories that are created based on arbitrary user actions, e.g., card transactions. In the latter case, any gap or discontinuity can be attributed to user behavior.

Splitting of long trajectories. Splitting is a novel data transformation operator. Record partitioning strategies have only rarely been employed in data anonymization literature, so there is not a large related literature to assess their impact on large records and large frequent patterns. In our setting, splitting helps the anonymization algorithm to better preserve the original data, as we experimentally demonstrate in Section 9. Still, it can cause the omission of long trajectories in the anonymized data. This effect cannot be denied in theory, but in practice, splitting helps preserve more frequent patterns than algorithms relying only on suppression. This happens because long trajectories are often rarer and they allow the identification of a person. In such cases, the

the anonymization algorithm has to eliminate the trajectory, independently of the data transforation operator that will be employed. If only suppression is available, then a large part of long trajectories is completely suppressed, whereas splitting allows to preserve parts of them unaffected. Moreover, large patterns are often a lot smaller than long trajectories in a given dataset, e.g., if we have trajectories with an average of 10 points a large pattern could have 5-6 points in practice. In these cases, splitting does not necessarily destroy the large patterns of a trajectory, since they are preserved in one of the sub-trajectories that are created. This is experimentally supported by the superior preservation of frequent sub-trajectories, as shown in Fig. 8.

3 Related work

There are numerous works on data anonymization and many of them focused on the anonymization of location data [?]. In the following, we survey the recent development in the field and position our work against them.

Clustering and perturbation. Several methods exist, using clustering and perturbation, which are applied to time-stamped trajectories. *Never Walk Alone* (*NWA*) [?] creates cylindrical volumes of radius δ, each one holding at least k trajectories, enforcing (k, δ)-anonymity. Each cylinder forms an anonymity group having indistinguishable trajectories. Domingo-Ferrer et al. [?], [?] showed that (k, δ)-anonymity does not effectively hide the original trajectory and proposed the *SwapLocations* algorithm to address this problem. *SwapLocations* use *microaggregation* to group trajectories together, and it then permutes the locations inside each cluster to guarantee anonymity. Kopanaki et al. [?], extend the work of [?] with personalized privacy parameters and with a technique that partitions trajectories to sub-trajectories. This partition, unlike our approach, is done as a preprocessing step and not as part of the main anonymization operator. Nergiz et al. [?], group and then reconstruct trajectories in order to protect from identity disclosure. The algorithm creates a group of size at least k, by matching locations inside each trajectory. For each anonymity group (i.e., a bounding box) a set of fake trajectories is reconstructed, that approximately preserves the distribution and covariance of the original group. Finally, Lin et al. [?] assume that the road-network data are apriori known to the data publishers, and appropriately cluster them to guarantee k-anonymity.

Location generalization and suppression. Yarovoy et al. [?] assume that for each user there is a different set of POIs that can be used as quasi identifiers. Thus, each trajectory is protected differently, using generalization in order to formulate non-disjoint anonymity groups. Poulis et al. [?], [?] propose methods that protect both from identity disclosure and attribute linkage, satisfying either the k^m-anonymity or the ℓ^m-anonymity principle [?]. Monreale et al. [?] presented the c-safety model, which is based on the notion of l-diversity. c-safety upper-bounds the probability of linking sensitive locations to trajectories by c. The proposed algorithm generalize POIs using a predefined locations taxonomy. If generalization cannot enforce c-safety, locations suppression is used. Cicek et al. [?] presents a similar notion, coined p-confidentiality, that limits the probability that a user has visited a sensitive location in a trajectory. The proposed method anonymizes the map first, by grouping sensitive points to generalized points. Chen et al. [?] focus on attribute linkage and proposed $(K, C)_L$-privacy: if an adversary knows up to L locations of a user's trajectory, a user is indistinguishable from at least $K-1$ users, while the probability of linking a user to its sensitive values is at most C.

Differential privacy. Methods based on differential privacy [?] release synthetic datasets, holding most attributes of the original datasets. These datasets are effective at supporting specific data analytic tasks, such as count query answering [?], [?], sequential event publishing [?] and frequent pattern mining [?]. Chen et al. [?] proposed a method that uses a *context-free*, *taxonomy* tree, to identify the set of counting queries that should be supported by the noisy summary. Bonomi et al. [?] use a prefix-tree to identify which location patterns will be used in the construction of the data summary. Andrés et al. [?] propose the notion of Geo-indistinguishability, a special case of differential privacy, to protect the exact location of a user in a location-based system. Their aim is not to completely avoid the inference of a user' location, but to limit the increase of the adversary's knowledge due to the observation. A significant difference from our approach is that it focuses on user's exact location and not on user trajectories. Jiang et al. [?] use differential privacy in the publication of trajectories, but they focus on trajectories in terms of exact coordinates, and not on semantic trajectories as we do. In [?] a technique based on differential privacy is proposed for publishing statistics from infinite location streams. Unlike our work, the focus is on publishing location counts. In a different approach, He et al. [?] create synthetic trajectories from a database of real trajectories, that share similar aggregate properties, while preserving the privacy of the users using differential privacy guaranties. Zhang et al. [?], investigate the utility of location data that have been anonymized with differently private methods, with respect to location recommendations. The work that lies closer to our is NGRAMS [?], that proposes an anonymization method based on differential privacy for sequential databases. NGRAMS extracts and publishes variable-length N-grams from the dataset, which in our context are variable length subtrajectories. NGRAMS allows the generation of a synthetic anonymous database, that enable a wide range of data analytics. In Section 9 empirically compare our method and NGRAMS [?].

4 A global suppression algorithm

We propose Algorithm GSUP, a greedy algorithm that iteratively suppresses locations, until the privacy constraint is met. The algorithm simulates the attack from any possible adversary, and then tackles the identified privacy breaches. GSUP first extracts the projected database $\mathcal{T}^A$ of each adversary $A \in \mathcal{A}$ (Line 1) and identifies the projections that lead to a privacy breach, by scanning $\mathcal{T}$ once, using Algorithm THREATID (Line 2). Then, GSUP runs a loop (Lines 3–11); while privacy breach problems have been identified, it attempts to unify a pair of projections (t_R^A, t_r^A) of the same adversary A, at least one of which is problematic.

Algorithm: GSUP

Input: A dataset $\mathcal{T}$, probability threshold P_{br}, locations of the background knowledge of each adversary in $\mathcal{A}$
Output: An anonymous version of $\mathcal{T}$
Local variable: Array $\mathcal{U}_{gain}$; $\mathcal{U}_{gain}(t_R^A, t_r^A)$ holds the cost for unifying projections t_R^A and t_r^A of the same adversary A.

1 Construct projection $\mathcal{T}^A$ for each adversary $A \in \mathcal{A}$
2 $(\mathcal{Q}, N, n@S_{\mathcal{T}})$ =THREATID($\mathcal{T}$, P_{br})
3 **while** $N > 0$ **do**
4 **for** *every* $A \in \mathcal{A}$ *and every* $t_R^A,\ t_r^A \in \mathcal{T}^A$ *such that* $t_r^A \neq t_R^A$ **do**
5 **if** t_R^A *or* t_r^A *is problematic and* $t_r^A \sqsubset t_R^A$ **then**
6 Calculate $\mathcal{U}_{gain}(t_R^A, t_r^A)$ for unifying t_R^A with t_r^A
7 Let mt_R^A and mt_r^A be the projections with the maximum $\mathcal{U}_{gain}(t_R^A, t_r^A)$
8 **for** *every* $t \in S_{\mathcal{T}}(mt_R^A)$ **do** `// trajectories supporting` t_R^A
9 Suppress all $\lambda \in t$, $\lambda \in mt_R^A$, $\lambda \notin mt_r^A$
10 Remove mt_R^A from $\mathcal{T}^A$
11 $(\mathcal{Q}, N, n@S_{\mathcal{T}})$ =THREATID($\mathcal{T}$, P_{br})
12 **return** $\mathcal{T}$

Two projections t_r^A and t_R^A can be *unified* only if one is a subtrajectory $t_r^A \sqsubset t_R^A$ of the other (we use t_R^A to denote the longer and t_r^A to denote the shorter). For example, $t_R^A = a_3 \rightarrow a_1$ (see t_5^A, t_6^A and t_7^A in Fig. 1b) can be unified with $t_r^A = a_1$ (see t_1^A). Technically, the unification results in the suppression of the points (e.g., a_3) in the longer projection (e.g., $a_3 \rightarrow a_1$) that are not contained in the shorter one (e.g., a_1) in all trajectories that support the former (e.g., t_5, t_6 and t_7). The unification is done in a way such that the resulting projection (e.g., a_1) does not violate the privacy of the data. In other words, if t_R^A is a problematic projection, then either t_R^A is not supported in the transformed database $\mathcal{T}'$ resulting from this unification, or $P_{\mathcal{T}'}(\lambda, t_R^A) \leq P_{br}$ for all $\lambda \notin \mathcal{L}^A$. In the example of Fig. 1, after the unification of $a_3 \rightarrow a_1$ with a_1, trajectories t_5, t_6 and t_7 become $t'_5 = t'_6 = a_1 \rightarrow b_1$ and $t'_7 = b_2 \rightarrow a_1$ and the problems of both $a_3 \rightarrow a_1$ and a_1 are resolved; $a_3 \rightarrow a_1$ is no longer supported in $\mathcal{T}'$ and a_1 does not map to any B-location with probability higher than 50% (see also Fig. 3b).

A unification between projections of the same adversary may also solve problems that other adversaries have. From the point of view of another adversary B, the appearances of points that do not belong to the unified projection are reduced. Subsequently, the confidence of B that a trajectory includes any of these points is also reduced. For example, when unifying $a_3 \rightarrow a_1$ with a_3 in Fig. 1, a_3 is removed from t_5 to t_7, and at the same time, a_3 appears fewer times in the trajectories where projection b_1 of adversary B is mapped to. To this end, we re-assess (Line 11) 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. Algorithm GSUP uses an array $\mathcal{U}_{gain}$, where each element $\mathcal{U}_{gain}(t_R^A, t_r^A)$ holds the benefit resulting from the unification of two projections t_R^A and t_r^A of the same adversary A. To measure $\mathcal{U}_{gain}(t_R^A, t_r^A)$, we consider two factors: *(a)* the anonymity gain and *(b)* the information loss.

Anonymity gain. Let N (respectively, N') be the total number of problems in the dataset before (respectively, after) the unification of t_R^A and t_r^A. The anonymity gain is expressed as the ratio $\frac{N-N'}{N}$. The *maximum* value of the ratio is 1 and, naturally, a greater ratio denotes a larger anonymity gain.

Heuristic for information loss.

The unification of t_R^A and t_r^A resolves some anonymity problems, but induces information loss. To compare different solutions we estimate different losses by using the $ploss$ heuristic function. $ploss$ quantifies the number of locations pairs in a trajectory that are removed due to unification. Specifically, if t is the original trajectory and t' is the trajectory that result in after the unification, $ploss$ is defined as $ploss(t, t') = 1 - p(t')/p(t)$, where p is the function that counts the pairs of locations of a trajectory. Since $p(t) = |t| \cdot (|t| - 1)/2$, we equivalently have

$$ploss(t, t') = 1 - \frac{|t'|(|t'| - 1)}{|t|(|t| - 1)}. \quad (2)$$

For instance in the example of Fig. 3, the unification of $a_3 \rightarrow a_1$ with a_1 turns trajectory $t_5 = a_3 \rightarrow a_1 \rightarrow b_1$ to $t'_5 = a_1 \rightarrow b_1$ (see also Figs. 3a and 3b) resulting a pairs lost of $ploss(t_5, t'_5) = 1 - \frac{2 \cdot 1}{3 \cdot 2} = 0.66$. It is clear that lower $ploss$ values indicate lower information loss. Several works in related literature use variations of the trajectory distortion metric [?], [?], [?], [?], which sums the point-wise distances between the original and anonymized trajectories [?]. Others, like [?] use as a heuristic the result of mining for maximal frequent patterns, but their heuristic is not invoked frequently. We opted for $ploss$ instead because: *(a)* it is computationally cheap (slightly cheaper than distortion), *(b)* it is meaningful for measuring information loss caused both by suppression and splitting and *(c)* experimentation showed that it provides superior results for GSUP, than trajectory distortion. It should be noted, that the basic requirements for a heuristic is to distinguish between a good and a bad solution and to be computationally cheap, since it is frequently used in the anonymization algorithm. The detailed evaluation of the quality of the anonymization output can be performed with more detailed and computationally expensive metrics (as it is done in Section 9).

Overall gain. To compute the overall benefit $\mathcal{U}_{gain}(t_R^A, t_r^A)$ of unifying t_R^A with t_r^A, we consider the trajectories that are affected by the unification and combine the anonymity gain and the information loss using the following formula:

$$\mathcal{U}_{gain}(t_R^A, t_r^A) = \frac{N - N'}{N} \cdot \frac{1}{\sum_{t \in \mathcal{S}} ploss(t, t')} \quad (3)$$

where *(a)* N and N' is the total number of problems in the dataset before and after the unification, respectively, *(b)* $\mathcal{S} \subseteq \mathcal{T}$ is the set of affected trajectories by the unification of t_R^A and t_r^A, *(c)* t' is the trajectory corresponding to t after the unification and *(d)* $ploss(t, t')$ is information loss metric of Equation 2. Note that the gain is maximized when the anonymity gain (i.e., $\frac{N-N'}{N}$) is maximized and the information loss metric (i.e., $\sum_{t \in \mathcal{S}} ploss(t, t')$) is minimized.

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 safe publication of $\mathcal{T}$.

Theorem 1. Algorithm GSUP derives a safe counterpart of a dataset $\mathcal{T}$.

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

(a) Original dataset $\mathcal{T}$

id	trajectory
t'_1	$a_1 \to b_2 \to b_3$
t'_2	$b_1 \to a_2 \to b_2 \to a_3$
t'_3	$a_2 \to b_3 \to a_3$
t'_4	$a_2 \to a_3 \to b_1$
t'_5	$a_1 \to b_1$
t'_6	$a_1 \to b_1$
t'_7	$b_2 \to a_1$
t'_8	$a_3 \to b_2 \to b_3$

(b) Unifying $a_3 \to a_1$ with a_1

id	trajectory
t'_1	$a_1 \to b_2 \to b_3$
t'_2	$b_1 \to a_3$
t'_3	a_3
t'_4	$a_3 \to b_1$
t'_5	$a_1 \to b_1$
t'_6	$a_1 \to b_1$
t'_7	a_1
t'_8	$a_3 \to b_2 \to b_3$

(c) GSUP($\mathcal{T}$)

t	loc	$gain$
t_1	b_2	0.71
t_2	b_1	**0.94**
t_3	b_3	0.35
t_4	a_2	0.71
t_5	a_3	0.71
t_6	a_3	0.71
t_7	b_2	0.71
t_8	a_3	0.35

(d) $LocMaxGain(t) = (loc, gain)$

id	trajectory
t'_1	$b_2 \to b_3$
t'_2	$a_2 \to a_3$
t'_3	$a_2 \to a_3$
t'_4	$a_2 \to a_3$
t'_5	$a_3 \to a_1$
t'_6	$a_3 \to a_1$
t'_7	$a_3 \to a_1$
t'_8	$b_2 \to b_3$

(e) LSUP($\mathcal{T}$)

Fig. 3: Algorithms GSUP (a-c) and LSUP (d-e) in operation

Proof. First, Algorithm GSUP correctly identifies the privacy breaches, using THREATID. Each time a unification is performed between two trajectories t_R^A and t_r^A, where $t_r^A \sqsubset t_R^A$, the breach for at least one of them is resolved. On the other hand, there is no way that after a unification, some $P(\lambda, t^A, \mathcal{T}')$ 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 adversary A, may only cause the reduction of the numerator in Equation 1 for any other adversary B, thus the P_{br} of any adversary B can only be reduced when removing points from an adversary 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 safe publication. ∎

A subtle thing to note is that after the unification of t_R^A with t_r^A, since t_R^A is no longer supported in the result, the adversary A can infer that t_R^A is unified with t_r^A. However, this does not allow inferring anything more specific that violates the privacy guaranty. Since t_r^A will not have any privacy problem, i.e., no point $\lambda \notin \mathcal{L}^A$ violates P_{br} for projection t_r^A, then nothing additional can be inferred for t_R^A, which is supported by the same trajectory set $S_{\mathcal{T}}(t_r^A)$, after the unification.

Let us now exemplify Algorithm GSUP with input the dataset $\mathcal{T}$ of Fig. 1a and the identification probability threshold $P_{br} = 0.5$ (similarly to Example 2). The process is summarized in Fig. 3; for illustration purposes $\mathcal{T}$ is also presented in Fig. 3a. Initially, the algorithm constructs the projection of adversaries A and B (the projection of A is illustrated in Fig. 1b). Then, GSUP (Line 2) uses THREATID to get the list of problematic pairs $\mathcal{Q}$ (highlighted in gray in Fig. 2), the total number of problems $N = 17$ of the dataset and the number of problems $n@S_{\mathcal{T}}(\lambda, t^A)$ of every problematic pair (λ, t^A) (also shown in Fig. 2). Next, the algorithm iterates over all possible unifications to identify that unifying $a_3 \to a_1$ with a_1 results in the greatest gain (Lines 4–6) and applies this unification (Lines 7–9). This unification results in the suppression of a_3 from trajectories t_5, t_6 and t_7 (illustrated in gray in Fig. 3a). The result after the unification is illustrated in Fig. 3b. In a similar manner, GSUP, unifies $a_2 \to a_3$ with a_3 which results in the suppression of a_2 in trajectories t_2, t_3 and t_4 (illustrated in gray in Fig. 3b). Following, the algorithm unifies *(a)* $b_1 \to b_2$ with b_1, *(b)* b_3 with $\emptyset$ and *(c)* b_2 with $\emptyset$ in turn to produce the anonymized dataset (depicted in Fig. 3c).

Multiple unifications per loop. In order to improve the efficiency of Algorithm GSUP, we perform multiple unifications at each **while** loop (Lines 3–11). At each loop, we select the top-s independent unifications, in terms of gain, 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 related to an adversary A, also affects problematic projections of all other adversaries B, where $A \neq B$.

4.1 Computing $\mathcal{U}_{gain}$

The computation of $\mathcal{U}_{gain}$ is crucial in Algorithm GSUP since, in every **while** iteration (Lines 3–11), it is calculated for every adversary and every problematic pair (Line 6). According to Equation 3, computing $\mathcal{U}_{gain}(t_R^A, t_r^A)$ requires to compute $\sum_{t \in S_{\mathcal{T}}} diff(t, t')$, N and N'. Computing the sum is reduced to simple arithmetic operations and can be performed efficiently. To compute N and N' we may use Algorithm THREATID twice, with input the dataset before and after the unification respectively. However, since the unification of two projections t_R^A and t_r^A, affects the problems that pertain to the trajectories that support t_R^A and t_r^A (which are in general a fraction of the dataset), we device a more efficient computation which:

- subtracts from the total number of problems the number of problems of all pairs (λ, t_R^A), as projection t_R^A is no longer supported.
- recalculates the problems of all pairs (λ, t_r^A). These pairs are affected, as trajectories supporting projection t_R^A now support projection t_r^A.
- recalculates all pairs (λ, t^B), where λ is every location suppressed from the unification of between t_R^A and t_r^A.

This computation drastically improves efficiency, as it considers only the affected trajectories, and users the initial calculation of Algorithm THREATID for unaffected trajectories.

5 A LOCAL SUPPRESSION ALGORITHM

Algorithm GSUP (Section 4) uses global suppression to unify two projections t_R^A and t_r^A. Every location that needs to be suppressed for this unification is globally suppressed, i.e., *all* its appearances are removed. To further reduce information loss, we propose a *local* suppression approach, which suppresses a location λ from a single trajectory t at a time.

Any location λ may act at the same time as a sensitive location for an adversary A and as a quasi identifier for some other adversary (Section 2.2). Thus, suppressing a location λ from a *single* trajectory t (unlike global suppression of a location), does not guarantee a reduction in the privacy threats. To evaluate the effect of suppressing location λ in trajectory t of a dataset $\mathcal{T}$, we introduce the *suppression effect gain*, denoted by $\mathcal{P}_{gain}(\lambda, t)$ that is defined as:

$$\mathcal{P}_{gain}(\lambda, t) = \frac{N - N'}{N} \cdot \frac{1}{ploss(t, t')}, \tag{4}$$

Algorithm: LSUP

Input: A dataset $\mathcal{T}$, probability threshold P_{br}, locations of the background knowledge of each adversary in $\mathcal{A}$
Output: An anonymous version of $\mathcal{T}$
Local variables: Set Π holds trajectories $t \in \mathcal{T}$ that have problematic pairs
Array $LocMaxGain$; for every trajectory t, $LocMaxGain(t) = (loc, gain)$ holds the location loc of t that when removed from t gives the maximum $gain$

1 $(\mathcal{Q}, N, n@S_{\mathcal{T}})$=THREATID$(\mathcal{T}, P_{br})$
2 **while** $N > 0$ **do** // There are still problems
3 $\Pi = \emptyset$, $LocMaxGain = ()$
4 Insert into Π every trajectory $t \in \mathcal{T}$ that has problematic pairs
5 **for** *every* $t \in \Pi$ **do**
6 **for** *every* $\lambda \in t$ **do**
7 Calculate $\mathcal{P}_{gain}(\lambda, t)$ for suppressing location λ from trajectory t
8 Let λ' be the location such that $\mathcal{P}_{gain}(\lambda', t)$ is maximum
9 $LocMaxGain(t) = (\lambda', \mathcal{P}_{gain}(\lambda', t))$
10 Let t_m be the trajectory where $LocMaxGain(t_m)$=$(loc_m, gain_m)$ stores the maximum $gain_m$
11 **if** $gain_m \leq 0$ **then return** GSUP$(\mathcal{T}, P_{br})$
12 Suppress location loc_m from trajectory t_m in $\mathcal{T}$
13 $(\mathcal{Q}, N, n@S_{\mathcal{T}})$=THREATID$(\mathcal{T}, P_{br})$ // Reassess dataset $\mathcal{T}$
14 **return** $\mathcal{T}$

where N and N' are the total number of problems before and after the suppression of location λ, respectively, and $ploss(t, t')$ is the pairs lost metric (Equation 2). Negative values of $\mathcal{P}_{gain}$ denote that suppressions increase anonymity threats. In general, greater values of $\mathcal{P}_{gain}$ indicate better suppression choices. More specifically, for a given trajectory t (where N is fixed), $\mathcal{P}_{gain}(\lambda, t)$ is maximized for the location λ, which when suppressed from t, solves the larger number of problems, i.e., when N' is minimized. When we compare different trajectories, $\mathcal{P}_{gain}(\lambda, t)$ takes into account the number of initial problems N and the pairs lost metric $ploss$, promoting trajectories t having high N and low $ploss$. Promoting low $ploss$ values is a natural choice; we also promote trajectories with many problems N since they are more likely to change in order to fulfill the anonymity guaranty than trajectories with fewer problems.

Our local anonymization method, illustrated in Algorithm LSUP, takes as input an unsafe dataset $\mathcal{T}$ and iteratively removes a location until dataset $\mathcal{T}$ becomes safe. LSUP algorithm uses set Π storing trajectories with problematic pairs. The algorithm also uses array $LocMaxGain$ that, for every trajectory t, stores at $LocMaxGain(t)$ a pair $(loc, gain)$ where $gain$ is the maximum suppression effect gain for trajectory t, and loc the location corresponding to that gain.

Initially, LSUP calls THREATID to compute the list of problematic pairs $\mathcal{Q}$, the total numbers of problems N and array $n@S_{\mathcal{T}}$. If problems exist in $\mathcal{T}$ (i.e., $N > 0$), Line 4 stores in set Π the trajectories with problematic pairs (trajectories with no problematic pairs are considered safe and are excluded). Then, LSUP scans all trajectories $t \in \Pi$, to compute array $LocMaxGain$ that stores the maximum gain and the corresponding location (Lines 5–9). Specifically, for every location λ of trajectory t, LSUP computes the gain $\mathcal{P}_{gain}(\lambda, t)$ of suppressing λ in t (Line 7). The maximum gain and the corresponding location of each trajectory t is stored at $LocMaxGain(t)$ (Lines 8–9). Following, the algorithm finds the trajectory t_m for which the corresponding pair $LocMaxGain(t_m) = (loc_m, gain_m)$ has maximum $gain_m$ (Line 10). If there are no possible suppressions decreasing the total number of problems, i.e., $gain_m \leq 0$ (Line 11), LSUP calls Algorithm GSUP to fix the remaining problematic pairs and produce the anonymized dataset. As shown in Section 4 and Theorem 1, GSUP always provides a safe counterpart of a dataset. If $gain_m > 0$ (Line 12), Algorithm LSUP suppresses the location loc_m from t_m. Additionally, it recalls THREATID to reassess the number of problems and update list $\mathcal{Q}$ (Line 13). LSUP repeats the above process (Lines 2–13) until all problems are resolved and returns a safe dataset $\mathcal{T}$.

Let us execute Algorithm LSUP having as input dataset $\mathcal{T}$ of Fig. 1a and identification probability threshold $P_{br} = 0.5$ (similarly the previous examples). The process is summarized in Figs. 3d–3e; for illustration purposes $\mathcal{T}$ is also presented in Fig. 3a. Initially, Algorithm LSUP (Line 1) uses THREATID to get the list of problematic pairs $\mathcal{Q}$ (highlighted in gray in Fig. 2), the total number of problems $N = 17$ of the dataset and the number of problems $n@S_{\mathcal{T}}(\lambda, t^A)$ of every problematic pair (λ, t^A) (also shown in Fig. 2). Following, the algorithm (Line 4) inserts to Π all trajectories of $\mathcal{T}$(since all have problematic pairs). Next, LSUP evaluates the first trajectory $t_1 = a_1 \rightarrow b_2 \rightarrow b_3$ and computes $\mathcal{P}_{gain}(a_1, t_1) = \frac{17-15}{17}\frac{1}{0.33} = 0.35$, $\mathcal{P}_{gain}(b_2, t_1) = \frac{17-13}{17}\frac{1}{0.33} = 0.71$ and $\mathcal{P}_{gain}(b_3, t_1) = \frac{17-13}{17}\frac{1}{0.33} = 0.71$. Thus, the algorithm sets $LocMaxGain(t_1) = (b_2, 0.71)$. In a similar manner, Algorithm LSUP calculates $LocMaxGain(t)$ for the remaining trajectories in Π (the final result is depicted in Fig. 3d) and Algorithm LSUP (Line 12) suppresses from trajectory t_2 location b_1 having maximum information gain (Fig. 3). Algorithm LSUP proceeds in a similar manner to produce the anonymized result of Fig. 3e.

Multiple suppressions per loop. In order to improve the efficiency of Algorithm LSUP, we perform multiple suppressions at each loop of Lines 2–13. Specifically, we select the top-s trajectories, in terms of gain, and provided that the gain is greater than 0, we simultaneously suppress the corresponding locations. If none such trajectory exists, we revert to Algorithm GSUP.

Computing $\mathcal{P}_{gain}$. We calculate $\mathcal{P}_{gain}$ without revisiting the data, as we did for $\mathcal{U}_{gain}$ in Section 4.1. This computation considers the suppression of location λ from trajectory t (resulting in trajectory t') and calculates:

- the effect of suppression for adversary A controlling location λ. Suppressing λ from t reduces the support of $\pi_A(t)$ by 1 ($\pi_A(t)$ is no longer supported by t). On the other hand, the support of $\pi_A(t')$ is increased by 1. Thus, we only recalculate problems for the pairs of projections $\pi_A(t)$ and $\pi_A(t')$.
- the effect for every other adversary B. As suppressing location λ from t only affects pair $(\lambda, \pi_B(t))$, our computation simply recalculates the number of problems for this affected pair.

6 A SPLITTING ALGORITHM

In Sections 4 and 5 we have described two anonymization methods that are based on suppression of locations. Although these transformations produce safe datasets, they need to suppress an increasingly large percentage of the dataset's locations, as the trajectories grow in size. With this problem in mind, we developed a third method that

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

(a) Original dataset $\mathcal{T}$

id	trajectory	id	trajectory
t'_1	a_1	t''_4	b_1
t''_1	$b_2 \rightarrow b_3$	t'_5	$a_3 \rightarrow a_1$
t'_2	b_1	t''_5	b_1
t''_2	a_2	t'_6	$a_3 \rightarrow a_1$
t'''_2	b_2	t''_6	b_1
t''''_2	a_3	t'_7	a_3
t'_3	a_2	t''_7	b_2
t''_3	b_3	t'''_7	a_1
t'''_3	a_3	t'_8	a_3
t'_4	$a_2 \rightarrow a_3$	t''_8	$b_2 \rightarrow b_3$

(b) Naive split of $\mathcal{T}$

Fig. 4: Splitting

keeps the distributions and the number of locations intact, by employing trajectory splitting.

A naive approach would split trajectories in a way that every trajectory contains only locations of the same adversary. For instance, for the original dataset of Fig. 4a the naive method would have resulted in the dataset of Fig. 4b. Although this dataset is safe, it does not withstand the whereabouts of users. Instead, we propose a method that measures the effects of each trajectory split and selects splits that solve anonymity problems and, at the same time, minimize trajectory changes.

Initially, to evaluate the effect of splitting a trajectory t at location λ of a dataset $\mathcal{T}$ with respect to the anonymity gain, we introduce the *splitting effect gain* ($\mathcal{S}_{gain}(\lambda, t)$) defined as:

$$\mathcal{S}_{gain}(\lambda, t) = \frac{N - N'}{N}, \tag{5}$$

where N and N' are the total number of problems before and after splitting trajectory t at location λ. Naturally, greater values of $\mathcal{S}_{gain}$ indicate better splitting choices and negative values denote that the splittings increase anonymity threats.

Similarly to suppression, spiting also affects the co-appearances of distinct locations in the same trajectory. To measure this distortion, we extend the $ploss(t, t')$ metric (Equation 2) as follows. If a trajectory t is split into trajectories t' and t'', we define $ploss(t, t', t'')=1 - (p(t')+p(t''))/p(t)$ where $p(t)$ is the a function that counts the pairs of locations in a trajectory. Equivalently, we have:

$$ploss(t, t', t'') = 1 - \frac{|t'|(|t'|-1)+|t''|(|t''|-1)}{|t|(|t|-1)}. \tag{6}$$

For instance, splitting trajectory $t_1 = a_1 \rightarrow a_2 \rightarrow b_2$ at location a_2 results in two trajectories $t'_1 = a_1 \rightarrow a_2$ and $t''_1 = b_2$ and a pairs lost metric of $ploss(a_2, t_1) = \frac{3-1-0}{3} = 0.66$.

Our splitting method, illustrated in Algorithm SPLIT, takes as input an unsafe dataset $\mathcal{T}$, and iteratively splits trajectories until dataset $\mathcal{T}$ is safe. SPLIT uses set Π that holds the trajectories that have problematic pairs. It also uses array $SplitMaxGain$ that, for every trajectory t, stores at $SplitMaxGain(t)$ a pair $(loc, gain)$ where $gain$ is the maximum splitting effect gain $\mathcal{S}_{gain}$ for trajectory t, and loc the location corresponding to that gain. Also SPLIT takes as input a parameter s which use will be clarified shortly.

Initially, SPLIT calls THREATID to compute the list of problematic pairs $\mathcal{Q}$, the total numbers of problems N and array $n@S_{\mathcal{T}}$. If there are problems in $\mathcal{T}$ (i.e., $N > 0$), Line 3 stores in Π the trajectories with problematic pairs. Then, Algorithm SPLIT scans all trajectories $t \in \Pi$, to compute

Algorithm: SPLIT

Input: A dataset $\mathcal{T}$, probability threshold P_{br}, locations of the background knowledge of each adversary in $\mathcal{A}$
Output: An anonymous version of $\mathcal{T}$
Local variables: Set Π holds trajectories $t \in \mathcal{T}$ that have problematic pairs
Array $SplitMaxGain$; for every trajectory t, $SplitMaxGain(t)=(loc, gain)$ holds the location loc of t that when t is split at loc gives the maximum $gain$
Parameter: Number s

```
1  (Q, N, n@S_T)=ThreatId(T, P_br)
2  while N > 0 do                                   // There are still problems
3      Π = ∅, SplitMaxGain = ()
4      Insert into Π every trajectory t ∈ T, |t| ≥ 2 that has problematic pairs
5      for every t ∈ Π do
6          for every λ ∈ t do
7              Calculate S_gain(λ, t) for splitting trajectory t at location λ
8          Let λ' be the location such that S_gain(λ', t) is maximum
9          SplitMaxGain(t) = (λ', S_gain(λ', t))
10     Let t_i, 1 ≤ i ≤ s, be s trajectories where SplitMaxGain(t_i) = (loc_i, gain_i) stores the top-s values for gain_i
11     if all the above trajectories have gain_i < 0, 1 ≤ i ≤ s then return Gsup(T, P_br)
12     Let t_m, 1 ≤ m ≤ s, with SplitMaxGain(t_m)=(loc_m, gain_m) be the trajectory such that (a) gain_m > 0 and (b) when t_m is split at loc_m results in the minimum pair loss
13     Split trajectory t_m at location loc_m in T
14     (Q, N, n@S_T)=ThreatId(T, P_br)    // Reassess dataset T
15 return T
```

array $LocMaxGain$ that stores the maximum gain and the corresponding location (Lines 5–9). Specifically, for every location λ of trajectory t, LSUP computes the gain $\mathcal{S}_{gain}(\lambda, t)$ of suppressing λ in t (Line 7). The maximum gain and the corresponding location of each trajectory t is stored at $SplitMaxGain(t)$ (Lines 8–9). Following, the algorithm finds s trajectories (where s is the user's parameter) that their corresponding $SplitMaxGain(t_i) = (loc_i, gain_i)$ pairs have the top-s values for $gain_i$ (Line 10). If none of these top-s values is positive, SPLIT reverts to GSUP (Line 11) since splitting cannot improve the anonymity of dataset $\mathcal{T}$. Otherwise, SPLIT chooses the trajectory t_m, $1 \leq m \leq s$, with $SplitMaxGain(t_m) = (loc_m, gain_m)$ satisfying that *(a)* $gain_m > 0$ and *(b)* when t_m is split at loc_m results in the minimum pair loss (Line 12). Following, the algorithm splits t_m at loc_m (Line 13) and recomputes $\mathcal{Q}$, N and $n@S_{\mathcal{T}}$ (Line 14). SPLIT repeats the above process (Lines 2–14) until all problems are resolved and returns a safe dataset $\mathcal{T}$ (Line 15).

For example, let us execute Algorithm SPLIT with input the dataset $\mathcal{T}$ of Fig. 1a, the identification probability threshold $P_{br} = 0.5$ (similarly to the previous examples), and $s = 2$. The process is summarized in Fig. 5; for illustration purposes $\mathcal{T}$ is also presented in Fig. 5a. Initially, SPLIT uses THREATID to get the problematic pairs $\mathcal{Q}$ (highlighted in gray in Fig. 2), the total number of problems $N = 17$ and the number of problems $n@S_{\mathcal{T}}(\lambda, t^A)$ of every problematic pair (λ, t^A) (also shown in Fig. 2). Then, the algorithm (Line 4) inserts to Π all trajectories of $\mathcal{T}$(since all have problematic pairs and their size is greater than 2). Following, SPLIT evaluates the first trajectory $t_1 = a_1 \rightarrow b_2 \rightarrow b_3$ and computes $\mathcal{S}_{gain}(a_1, t_1) = \frac{17-15}{17} = 0.12$ and $\mathcal{S}_{gain}(b_2, t_1) = \frac{17-17}{17} = 0$. Thus, the algorithm sets $SplitMaxGain(t_1) = (a_1, 0.12)$ (Line 9). In a similar manner, Algorithm SPLIT calculates $SplitMaxGain$ for the remaining trajectories in Π; the final result is depicted in Fig. 5b. Next, in Line 10, algorithm selects trajectories t_5 and t_6 and calculates their respective $ploss$ (depicted in Fig. 5b). Since for both cases the pair loss

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

(a) Original dataset $\mathcal{T}$

t_i	loc_i	$gain_i$	$ploss$
t_1	a_1	0.12	-
t_2	b_1	0.24	-
t_3	b_3	0.06	-
t_4	a_2	0.12	-
t_5	a_3	**0.41**	0.66
t_6	a_3	**0.41**	0.66
t_7	b_2	0.18	-
t_8	a_3	0.12	-

(b) $SplitMaxGain(t)$=$(\lambda, gain_m)$ and $ploss$

id	trajectory
t'_1	$a_1 \to b_2 \to b_3$
t'_2	$b_1 \to a_2 \to b_2 \to a_3$
t'_3	$a_2 \to b_3 \to a_3$
t'_4	$a_2 \to a_3 \to b_1$
t'_5	a_3
t''_5	$a_1 \to b_1$
t'_6	$a_3 \to a_1 \to b_1$
t'_7	$a_3 \to b_2 \to a_1$
t'_8	$a_3 \to b_2 \to b_3$

(c) First step of SPLIT

id	trajectory
t'_1	$a_1 \to b_2 \to b_3$
t'_2	b_1
t''_2	$a_2 \to b_2$
t'''_2	a_3
t'_3	a_2
t''_3	b_3
t'''_3	a_3
t'_4	$a_2 \to a_3$

id	trajectory
t''_4	b_1
t'_5	a_3
t''_5	$a_1 \to b_1$
t'_6	$a_3 \to a_1$
t''_6	b_1
t'_7	$a_3 \to b_2 \to a_1$
t'_8	$a_3 \to b_2 \to b_3$

(d) SPLIT($\mathcal{T}$): The splitting method

Fig. 5: Algorithm SPLIT in operation ($P_{br} = 0.5$ and $s = 2$)

is the same Algorithm SPLIT choose one randomly and (Line 13) splits trajectory t_5 at location a_3 having minimum $ploss$. The result is depicted in Fig. 5c. SPLIT proceeds in a similar manner; the final result is illustrated in Fig. 5d.

Multiple splittings per loop. In order to improve the efficiency of Algorithm SPLIT, we perform multiple suppressions at each loop of Algorithm SPLIT (Lines 2–14) similarly to the suppression cases.

Computing $\mathcal{S}_{gain}$. In a nutshell, splitting trajectory t decreases the support of $\pi_A(t)$ by one (trajectories t' and t'' do not support $\pi_A(t)$) and increases the support of projections $\pi_A(t')$ and $\pi_A(t'')$ by one (projections $\pi_A(t')$ and $\pi_A(t'')$ are supported by trajectories t' and t'' respectively). Thus our computation simply recalculates the problems for all pairs of projections $\pi_A(t)$, $\pi_A(t')$ and $\pi_A(t'')$.

7 A MIXED ALGORITHM

SPLIT has the advantage of publishing the same number of locations as the original dataset, but in some cases LSUP solves problems with less information loss. To combine the advantages of both strategies, we developed Algorithm MIX, where its main difference to SPLIT algorithm is in Step 13. In more details, at each iteration MIX finds the best splitting location λ of a trajectory t as in Step 12 of SPLIT. Next, it calculates the problems of trajectory t after suppressing location λ. If the final problems of trajectory t are resolved with suppression then MIX suppresses location λ from t, otherwise it splits trajectory t at λ. The intuition behind this approach, is that splitting a trajectory t to t' and t'', decreases the number of total problems, but also introduces two new trajectories t' and t'' which may be problematic. On the other hand, suppressing λ from t results in a problems free trajectory.

8 COMPLEXITY ANALYSIS

Let $\mathcal{A}$ be the set of adversaries, $\mathcal{T}^A$ be the subset of the trajectories in $\mathcal{T}$ owned by an adversary A, t be a trajectory in $\mathcal{T}^A$, and N be the number of anonymity problems.

Algorithm THREATID iterates over all locations of all trajectories of all adversaries. This results in an overall complexity $\mathcal{O}(|\mathcal{A}| \cdot |\mathcal{T}^A| \cdot |t|)$ time. The most expensive loop of Algorithm GSUP is done in Lines 3–6. In the worst case, GSUP iterates over all anonymity problems N, over all adversaries in $\mathcal{A}$ and over all pairs of trajectories in $\mathcal{T}^A$ to compute $\mathcal{P}_{gain}$ (that requires scanning $\mathcal{T}^A$). In total, GSUP requires $\mathcal{O}(N \cdot |\mathcal{A}| \cdot |\mathcal{T}^A|^3)$ time. Algorithms LSUP, SPLIT and MIX in the worst case execute GSUP, thus, they also have an $\mathcal{O}(N \cdot |\mathcal{A}| \cdot |\mathcal{T}^A|^3)$ time complexity. Note, that this is worst case complexity, in practice the algorithms behave a lot better. As we see in the results for our real-life dataset Fig. 11, the behavior of all algorithms is almost linear to the size of D. Moreover, we can regulate the actual performance of the algorithms by adjusting m (the number of trajectories that are transformed in each iteration).

9 EXPERIMENTAL EVALUATION

All algorithms were implemented in C++ and tested on an Intel Core i7 at 2.2 GHz with 6 GB of RAM. We also compared our methods with the NGRAMS approach from [**?**], which is based on differential privacy.

Datasets. We used two datasets: *Oldenburg* and *Gowalla*. *Oldenburg* was created using Brinkhoff's data generator [**?**] and contains synthetically generated trajectories of objects moving on Oldenburg's city map. *Oldenburg* contains 18,143 trajectories passing from 100 distinct locations and having average length 4.72. *Gowalla* contains real user check-in data from the Gowalla social network [**?**]. The *Gowalla* dataset contains 59,994 trajectories of average length is 7.95, connecting 513 distinct locations in New York city. Each trajectory corresponds to the check-ins of a user within 24 hours. The timestamp granule is set to hours; i.e., each spatiotemporal point of a trajectory corresponds to a check-in to a certain location at a certain hour, i.e., a location visited in different hours would create different points and a location visited k times within the same hour it would create k duplicate entries of the same point.

9.1 Measuring utility

Estimating the utility of the anomymized dataset is challenging, since it highly depends on its intended use. Thus to evaluate our methods, we use different utility metrics that capture trajectory changes but also measure the impact on count queries and frequent patterns. The latter two, are the basis for utility metrics in several related works [**?**], [**?**], [**?**].

Average locations appearance ratio. The first metric measures the level of suppression performed by each method. We calculate the change in the number of appearances of locations in the original and the anonymized dataset. For every location l, we define the *locations appearance ratio* $\mathcal{R}(l)$, as the ratio of the appearance of l in the original dataset $\mathcal{T}$ to the appearance of l in the anonymized counterpart $\mathcal{T}'$. The values of $\mathcal{R}(l)$ are in $[0, 1]$; a value of 0 means that l was completely suppressed in the anonymized dataset while a value of 1 means that l was unaffected.

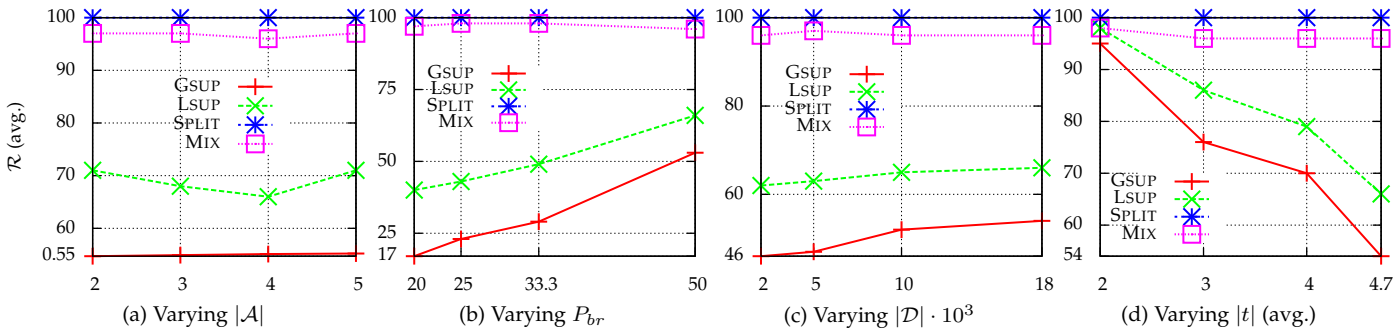


Fig. 6: Average locations appearance ratio $\mathcal{R}$ (higher is better)

Frequent sequential patterns published. We also evaluate the performance of our methods in sequential pattern mining. We use Prefixspan [**?**], to obtain the frequent sequential patterns (i.e., the sequential patterns having support larger than a threshold) in the original and the anonymous versions of the dataset. Then, we calculate the percentage of the frequent sequential patterns of the original dataset that are preserved in the anonymous counterpart.

AREL. The *Average Relative Error* (AREL) measure [**?**] estimates the average number of trajectories that are retrieved incorrectly due to anonymization, as part of the answers to a workload of `COUNT` queries. Lower AREL scores indicate that the anonymized data estimate more accurately the number of co-occurrences of locations. AREL is used in many anonymity methods [**?**], [**?**]. To measure the AREL, we issue a query workload on the original and the anonymous datasets. Our query workload selects the 200 most frequent ordered subsequences. Such queries are of the following form:

```
SELECT COUNT(*) FROM trajectories t WHERE t
contains locations a, b and in that order.
```

Pairs lost. Our algorithms use *ploss* to heuristically measure the effect of a trajectory change. We calculate *ploss* on the dataset level, but taking into account the *difference in the total number of pairs in the original and the anonymized dataset*. Unfortunately, *ploss* on the dataset level is not that informative, since it only reflects the sum of all pairs lost in the anonymization process, independently of their value for data analysis. Note, for instance, that the removal of a very frequent pair would not affect the statistical distribution of the dataset as the removal of a less frequent pair. The removal of a very rare pair is frequently not important either, since it does not reveal a trend. To this end, we do not use *ploss* as a main evaluation metric, but for completeness, we report it in Fig. 9a.

9.2 Results

We evaluate our algorithms by varying the following:

- The probability threshold P_{br} (default value 50%).
- The number of adversaries $|\mathcal{A}|$ (default value 4). We have examined two models for the adveraries. In the Oldenburg adversaries are equal; all locations of the original dataset have been uniformly distributed to the adversaries. In the case of *Gowalla*, we partitioned all locations of the original dataset to adversaries, according to skewed, zipfian distribution.
- The size of the dataset $|D|$. We have created smaller datasets, by taking parts of the original ones, e.g., the first 5k trajectories, the first 10k trajectories etc.
- The average size of the trajectory $|t|$. We create datasets by truncating all trajectories of the original dataset at a certain point, e.g., we only keep the 7 first points in each trajectory. With some experimentation we can regulate accurately the resulting average trajectory size.
- The number of trajectories m that are changed (unified or split) in each algorithm iteration (default value 10).

We first evaluate our algorithms using the *Oldenburg* dataset. Figs. 6a–6d show the average appearance ratio $\mathcal{R}$ for variable number of adversaries, probability breach threshold, dataset size and average trajectories length, respectively. Algorithm SPLIT, that is based on splitting, has $\mathcal{R}(l) = 100\%$ (Fig. 6a) that means that it published the exact number of locations as in the original dataset. Algorithm MIX that favors splitting but may also use suppression, achieves an $\mathcal{R}$ that is on average 5.11% lower than SPLIT. Algorithms GSUP and LSUP, that are exclusively based on suppression, have lower $\mathcal{R}$. Specifically, LSUP (that is based on local suppression) publishes on average 39.56 more locations than GSUP (that is based on global suppression). Both these algorithms publish on average 53.65% fewer locations than SPLIT.

Next, we test the AREL scores of our methods (Figs. 7), varying the number $|\mathcal{A}|$ of adversaries (Fig. 7a). AREL is low for very few adversaries, because there are more projections per adversary, and thus, our methods have more flexibility in selecting the most beneficiary problematic pair. As the number of adversaries increases, AREL scores gradually increase and then decrease again, since if there are many adversaries, the chances of finding problematic projections decrease. Between our methods, MIX has the lowest AREL, as splitting preserves the number of published locations, while suppression reduces the number of iterations in order to produce the anonymized dataset. The AREL scores of MIX are on average 52.34%, 43.64% and 6.18% better than GSUP, LSUP and SPLIT respectively. We then evaluate AREL against the identification probability threshold P_{br} (Fig. 7b). As expected, increasing P_{br} leads to less problematic pairs and thus, in better AREL scores. Similarly to the previous experiment, MIX is slightly better than SPLIT, but significantly better than GSUP and LSUP. In Fig. 7c we observe that AREL scores improve for larger datasets. The reason is that as $|\mathcal{D}|$ increases, the initial supports of the projections increase in size (i.e., the probability that two

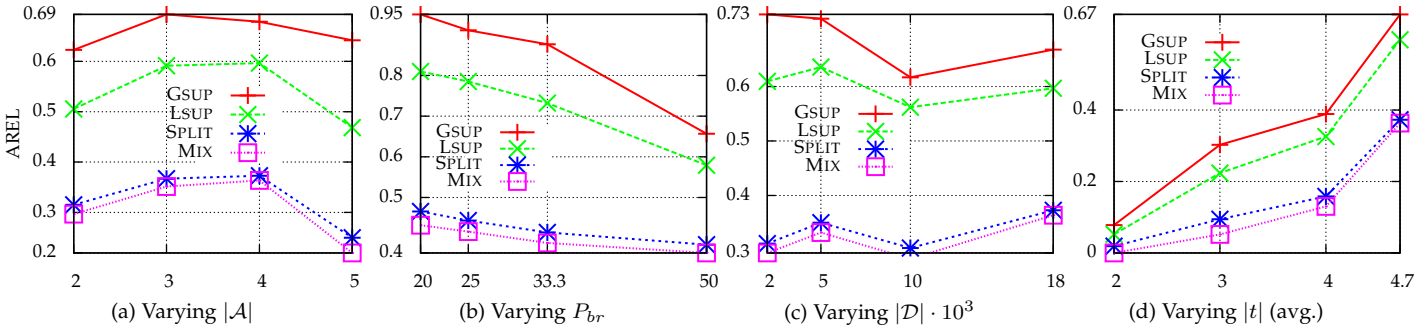


Fig. 7: AREL evaluation (lower is better)

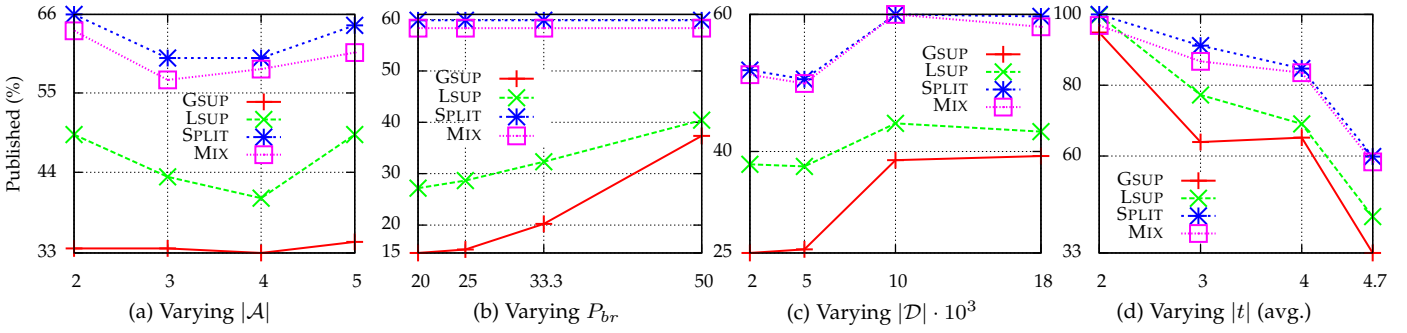


Fig. 8: Top-2% frequent sequences published (higher is better)

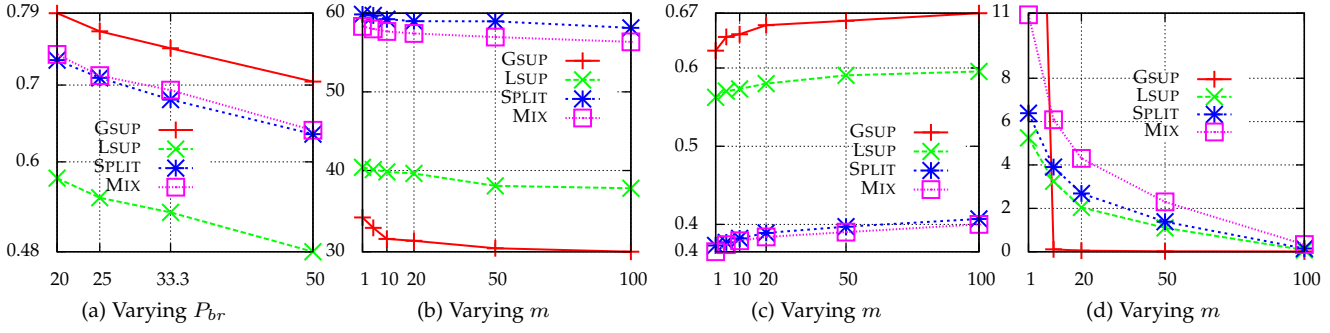


Fig. 9: (a) *ploss* for varying P_{br}(b) Top-2% frequent sequences , (c) AREL and (d) Efficiency (sec·10^2) for varying m

trajectories have the same projection increases) and, thus, the number of problematic pairs is reduced. Finally, AREL scores decrease as the average trajectories length increases (Fig. 7d), as a dataset comprised of small trajectories has fewer problematic pairs (i.e., each trajectory holds only a small combination of locations).

Following in Fig. 8, we evaluate the percent of the published frequent sequential patterns. In our experiments, we set the frequent patterns threshold to 2% that considers as frequent the sequential patterns that appear at least in the 2% of trajectories of the original dataset. In Fig. 8a, we report the percentage of preserved frequent patterns for various values of $|\mathcal{A}|$. As seen in the previous experiment, more adversaries decrease the probability of finding problematic pairs, thus, the number of frequent published subsequences increases. Algorithms SPLIT and MIX have the best preservation percentage, LSUP follows and GSUP comes last. Fig. 8b presents the percent of the published frequent sequential patterns for variable P_{br}. Larger values of P_{br}, result in fewer problematic pairs. Thus, all methods manage to preserve more frequent sequential patterns for larger P_{br}. SPLIT preserved on average 4.45% more frequent sequential patterns than MIX algorithm and up to 2.3 and 4.34 times more frequent sequential patterns that LSUP and GSUP respectively. In a similar manner, SPLIT and MIX outperform the other algorithms for varying dataset sizes and average trajectories length (Figs. 8c–9d).

Finally, in Fig. 9a we present the algorithms performance with respect to pair loss (*ploss*), when we vary P_{br}. We calculated dataset *ploss* based on *the difference in total number of pairs in the original and the anonymized dataset*. LSUP appears as the algorithm which preserves *ploss* better, since *ploss* penalizes splitting more than suppression. At the same time, as the previous experiments show, LSUP provides less accurate answers to count queries and preserves less frequent itemsets.

Effect of m. Figs. 9b-9d highlight the importance of parameter m in the performance of the proposed algorithms. We see that increasing m up to 100 only decreases utility by 4.5% on average. On the other hand, all methods are at least 100 times faster for $m = 100$ (Figs. 9d). Thus, we can use

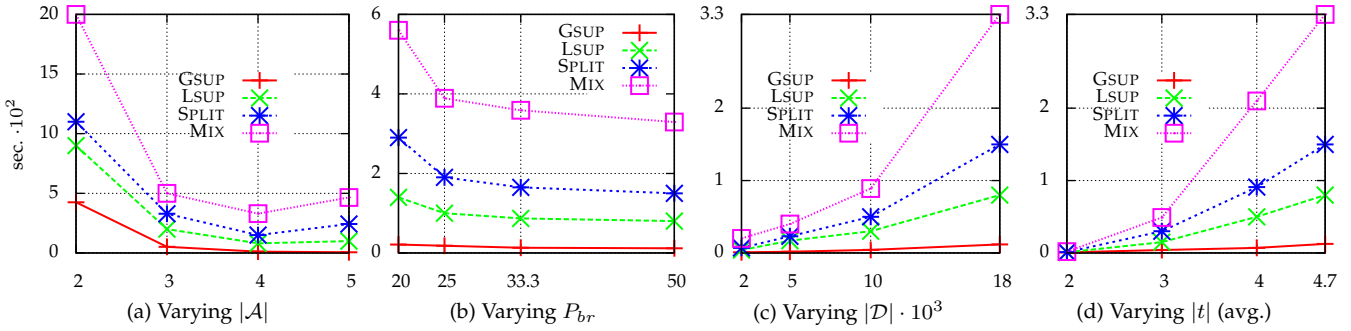


Fig. 10: Efficiency for varying (a) $|A|$, (b) P_{br}, (c) dataset size and (d) average trajectories length

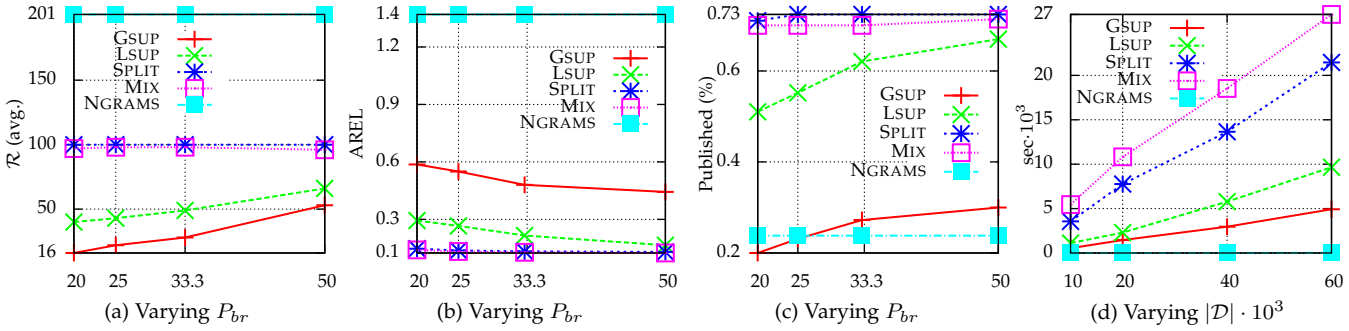


Fig. 11: (a) Average locations appearance ratio, (b) ARE, and (c) Top frequent sequences published for varying P_{br}

large values of m to significantly improve efficiency, without compromising utility.

Time performance. Fig. 10a shows the efficiency of our methods for various numbers of adversaries. As expected, increasing the number of adversaries, reduces the runtime of our methods. Also, GSUP is the fastest between our methods, as it employees global suppression, solving more problems at each loop. On the other hand, Algorithm LSUP is slightly faster than SPLIT, as suppression eliminates more problems than splitting, with a great cost in utility. Finally, MIX is the slowest of our methods, as it has to evaluate every location for both splitting and suppression. Lower runtime cost was achieved for higher values of P_{br} (Fig. 10b) as well. Again, GSUP performs best, being on average 2, 2.1 and 3 times faster that LSUP, GSUP and MIX respectively. Finally, we evaluate the runtime of our methods for various input dataset sizes and average trajectory lengths (Figs. 10c-10d). Increasing the dataset size or the average trajectories length, results in higher runtime, as our methods has to evaluate more and larger trajectories. Again, GSUP algorithm is the fastest algorithm, as the global suppression scheme it entails favours efficiency. Algorithm LSUP appears 22.14% faster than SPLIT on average, while MIX is 1 to 3 times slower than the other methods.

***Gowalla* and comparison with NGRAMS** We complement our empirical evaluation with experiments on the *Gowalla* dataset and a comparison with NGRAMS, a method based on differential privacy. For NGRAMS we have used as parameters the default parameters of [**?**]: $l_{max} = 20$, $n_{max} = 5$ and $e = 0.1$. For *Gowalla*, we use the same default parameters as for *Oldenburg*, but we experiment with a skewed (zipfian) distribution of the locations to adversaries. The first result from Fig. 11 is that the behavior of the algorithms on *Gowalla* is consistent with their performance on *Oldenburg*. MIX remains the best algorithm and the comparative picture of the algorithms remains the same. Because the real data are more skewed, AREL and frequent pattern results are better than for *Oldenburg*.

An important reason we opted for a syntactical anonymity approach is the sparse multidimensional nature of our data, which forces methods based on differential privacy to add substantial noise to the result. At the same time differential privacy is not immune to attacks [**?**]. The comparison with NGRAMS backs up our decision, since it shows the superiority of our approach in terms of utility. Almost all algorithms in all settings outperform the NGRAMS algorithm for all utility metrics. NGRAMS adds significant noise; as shown in Fig. 11a, it doubles the points in the dataset. NGRAMS's results, in terms of AREL and preserved frequent itemsets, are also inferior to all proposed algorithms. This behavior is expected, since differential privacy cannot easily preserve accurate counts when subtrajectories are considered. The sparse multidimensional nature of the data results to low counts for subtrajectories and NGRAMS adds substantial noise to preserve the privacy guaranty.

Summary of the evaluation. The experimental evaluation confirms that the performance of methods that rely solely on generalization and suppression greatly decreases as size of the trajectory grows (which is consistent with most anonymization methods). On the other hand, the relevant experiments in (Figs. 6.d, 7 and 8) show that SPLIT, manages to address this problem, and moreover that MIX efficiently combines the best behavior of SPLIT and LSUP.

10 Conclusions

In this paper, we studied the problem of protecting datasets, holding user movements, from adversaries who can use their partial knowledge to infer locations unknown to them. We proposed four anonymization algorithms, employing locations suppression, trajectories splitting, or both suppression and splitting, to protect such datasets. Finally, we experimentally showcased the effectiveness of our algorithms, in terms of data utility preservation and efficiency.

References

[1] O. Abul, F. Bonchi, and M. Nanni. Never walk alone: Uncertainty for anonymity in moving objects databases. In *ICDE*, pages 376–385, 2008.

[2] M. E. Andrés, N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi. Geo-indistinguishability: differential privacy for location-based systems. In *CCS*, 2013.

[3] L. Bonomi and L. Xiong. A two-phase algorithm for mining sequential patterns with differential privacy. In *CIKM*, pages 269–278, 2013.

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

[5] Y. Cao and M. Yoshikawa. Differentially private real-time data release over infinite trajectory streams. In *MDM*, 2015.

[6] R. Chen, G. Acs, and C. Castelluccia. Differentially private sequential data publication via variable-length n-grams. In *CCS*, pages 638–649, 2012.

[7] R. Chen, B. Fung, N. Mohammed, B. Desai, and K. Wang. Privacy-preserving trajectory data publishing by local suppression. *Inf. Sci.*, 231:83–97, 2013.

[8] R. Chen, B. C. Fung, N. Mohammed, B. C. Desai, and K. Wang. Privacy-preserving trajectory data publishing by local suppression. *Information Sciences*, (0), 2011.

[9] E. Cho, S. A. Myers, and J. Leskovec. Friendship and mobility: User movement in location-based social networks. In *KDD*, pages 1082–1090, 2011.

[10] A. E. Cicek, M. E. Nergiz, and Y. Saygin. Ensuring location diversity in privacy-preserving spatio-temporal data publishing. *VLDB J.*, 23(4):609–625, 2014.

[11] C. Clifton and T. Tassa. On syntactic anonymity and differential privacy. *Trans. Data Privacy*, 6(2):161–183, 2013.

[12] J. Domingo-Ferrer and R. Trujillo-Rasua. Microaggregation- and permutation-based anonymization of movement data. *Journal of Information Sciences*, 208:55–80, 2012.

[13] C. Dwork. Differential privacy. In *ICALP*, pages 1–12, 2006.

[14] G. Ghinita. *Privacy for Location-based Services*. Synthesis Lectures on Information Security, Privacy, and Trust. Morgan & Claypool Publishers, 2013.

[15] X. He, G. Cormode, A. Machanavajjhala, C. M. Procopiuc, and D. Srivastava. DPT: differentially private trajectory synthesis using hierarchical reference systems. *PVLDB*, 8(11):1154–1165, 2015.

[16] K. Jiang, D. Shao, S. Bressan, T. Kister, and K.-L. Tan. Publishing trajectories with differential privacy guarantees. In *SSDBM*, 2013.

[17] G. Kellaris, S. Papadopoulos, X. Xiao, and D. Papadias. Differentially private event sequences over infinite streams. *Proceedings of the VLDB Endowment*, 7(12):1155–1166, 2014.

[18] D. Kopanaki, V. Theodossopoulos, N. Pelekis, I. Kopanakis, and Y. Theodoridis. Who cares about others' privacy: Personalized anonymization of moving object trajectories. In *EDBT*, 2016.

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

[20] D. Lin, S. Gurung, W. Jiang, and A. Hurson. Privacy-preserving location publishing under road-network constraints. In *DASFAA*, pages 17–31, 2010.

[21] G. Loukides, A. Gkoulalas-Divanis, and B. Malin. COAT: Constraint-based anonymization of transactions. *Knowl. Inf. Systems*, 28(2):251–282, 2011.

[22] A. Monreale, G. L. Andrienko, N. V. Andrienko, F. Giannotti, D. Pedreschi, S. Rinzivillo, and S. Wrobel. Movement data anonymity through generalization. *TDP*, 3(2):91–121, 2010.

[23] M. E. Nergiz, M. Atzori, and Y. Saygin. Towards trajectory anonymization: a generalization-based approach. In *SPRINGL*, pages 52–61, 2008.

[24] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, and M. Hsu. Mining sequential patterns by pattern-growth: the prefixspan approach. *KDE*, 16(11):1424–1440, 2004.

[25] G. Poulis, G. Loukides, A. Gkoulalas-Divanis, and S. Skiadopoulos. Anonymizing data with relational and transaction attributes. In *ECML/PKDD (3)*, pages 353–369, 2013.

[26] G. Poulis, S. Skiadopoulos, G. Loukides, and A. Gkoulalas-Divanis. Select-organize-anonymize: A framework for trajectory data anonymization. In *ICDM*, pages 867–874, 2013.

[27] G. Poulis, S. Skiadopoulos, G. Loukides, and A. Gkoulalas-Divanis. Apriori-based algorithms for k^m-anonymizing trajectory data. *TDP*, 7(2):165–194, 2014.

[28] M. Terrovitis and N. Mamoulis. Privacy preservation in the publication of trajectories. In *MDM*, pages 65–72, 2008.

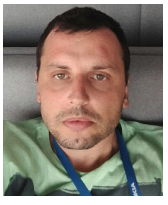
[29] M. Terrovitis, N. Mamoulis, and P. Kalnis. Local and global recoding methods for anonymizing set-valued data. *The VLDB Journal*, 20(1), 2011.

[30] R. Trujillo-Rasua and J. Domingo-Ferrer. On the privacy offered by (k, δ)-anonymity. *Information Systems*, 38(4):491–494, June 2013.

[31] R. Yarovoy, F. Bonchi, L. V. S. Lakshmanan, and W. H. Wang. Anonymizing moving objects: how to hide a mob in a crowd? In *EDBT*, pages 72–83, 2009.

[32] J. Zhang, G. Ghinita, and C. Chow. Differentially private location recommendations in geosocial networks. In *MDM*, 2014.

Manolis Terrovitis is an associate researcher at the Institute for the Management of Information Systems (IMIS) of the Research and Innovation Centre in Information, Communication and Knowledge Technologies Athena. He received his PhD in 2007 from the National Technical University of Athens. His main research interests lie in the areas of data privacy, indexing and query evaluation.



Giorgos Poulis received a bachelor in Informatics from University of Piraeus, Greece, a M.Sc. from National Technological University of Athens, Greece and a PhD from University of Peloponnese, Greece. He is currently a software engineer in Nokia Networks, Athens, Greece. His main research interests lie in the areas of data privacy.

Nikos Mamoulis received a diploma in Computer Engineering and Informatics in 1995 from the University of Patras, Greece, and a PhD in Computer Science in 2000 from the Hong Kong University of Science and Technology. He is currently an associate professor at the Department of Computer Science, University of Ioannina, which he joined in 2014. His research focuses on management and mining of complex data types, privacy and security in databases, and uncertain data management.

Spiros Skiadopoulos is currently a Professor at the Dept. of Informatics and Telecommunications at University of Peloponnese. He received a diploma and a PhD degree from the National Technical University of Athens and a MPhil degree from Manchester Institute of Science and Technology (UMIST). He has worked in a variety of areas including data management, knowledge representation and reasoning. His current research interests include anonymity and big data management.