Data & Knowledge Engineering 94 (2014) 89–109

Contents lists available at ScienceDirect

Data & Knowledge Engineering

journal homepage: www.elsevier.com/locate/datak

Editorial

Privacy-preserving trajectory stream publishing

Khalil Al-Hussaeni [a], Benjamin C.M. Fung [b,*], William K. Cheung [c]

[a] CIISE, Concordia University, Montreal H3G 1M8, Canada
[b] School of Information Studies, McGill University, Montreal H3A 1X1, Canada
[c] Department of Computer Science, Hong Kong Baptist University, Hong Kong

ARTICLE INFO

Article history:
Received 26 November 2013
Received in revised form 14 September 2014
Accepted 19 September 2014
Available online 28 September 2014

Keywords:
Data sharing
Data mining
Privacy protection
Spatio-temporal databases
Data stream

ABSTRACT

Recent advancement in mobile computing and sensory technology has facilitated the possibility of continuously updating, monitoring, and detecting the latest location and status of moving individuals. Spatio-temporal data generated and collected on the fly are described as *trajectory streams*. This work is motivated by the concern that publishing individuals' trajectories on the fly may jeopardize their privacy. In this paper, we illustrate and formalize two types of privacy attacks against moving individuals. We devise a novel algorithm, called *Incremental Trajectory Stream Anonymizer* (*ITSA*), for incrementally anonymizing a sequence of sliding windows on trajectory stream. The sliding windows are dynamically updated with joining and leaving individuals. The sliding windows are updated by using an efficient data structure to accommodate massive volume of data. We conducted extensive experiments on simulated and real-life data sets to evaluate the performance of our method. Empirical results demonstrate that our method significantly lowers runtime compared to existing methods, and efficiently scales when handling massive data sets. To the best of our knowledge, this is the first work to anonymize high-dimensional trajectory stream.

1. Introduction

The improvement of information technology in the past years has facilitated sharing data among organizations, firms, and to the public. Location-aware devices, such as GPS and mobile phones, *constantly* report spatio-temporal data of a moving object or the individual carrying this object. In many cases, it is important to publish the automatically-collected data on the fly for various purposes, such as traffic analysis, live monitoring of moving objects, and mining recent events in a data stream. This process becomes of vital importance especially when it is essential to take immediate actions or follow certain detection or prevention measures. Nevertheless, releasing the automatically-collected raw data by a data holder for analysis and service improvement may compromise individuals' privacy. We assume that recipients of a published data stream are untrustworthy, and they may attempt to identify target victims or infer their sensitive information. In this paper, we study the challenges in anonymizing a stream of trajectories, and propose an efficient algorithm to anonymize a trajectory stream with the goal of minimizing data distortion.

Fig. 1 shows an overview of the trajectory stream environment. A trajectory stream S is a continuous sequence of *triples*, in which each *triple* has the form $\langle id, loc, t\rangle$, indicating a person p with id is at location loc at timestamp t. A combination of loc and t is called a doublet. We assume that the trajectory stream S is published for stream mining [16] or the trajectory paths are simply displayed on screen. We propose a trajectory stream anonymization method based on a *sliding window* [18,5]. The literature has

* Corresponding author at: 3661 Peel St., Montreal, QC, Canada H3A 1X1.
E-mail addresses: k_alhus@ciise.concordia.ca (K. Al-Hussaeni), ben.fung@mcgill.ca (B.C.M. Fung), william@comp.hkbu.edu.hk (W.K. Cheung).

http://dx.doi.org/10.1016/j.datak.2014.09.004
0169-023X/

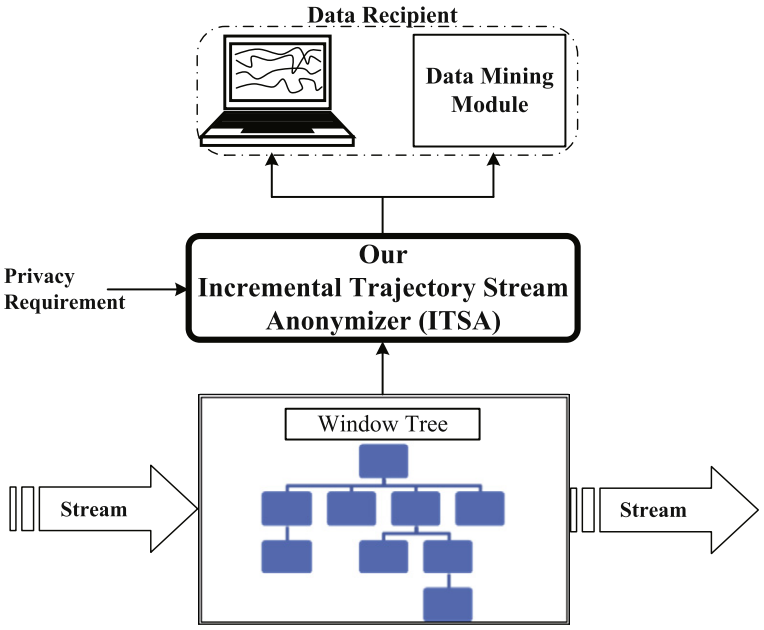

Fig. 1. Mining trajectory stream over a sliding window.

defined two types of sliding windows: *count-based* and *time-based* [6,18]. The former type defines a window that includes the N most recent data elements while the latter type defines a window that includes all elements belonging to the most recent N timestamps. We adopt a time-based sliding window because it is more general than count-based. However, using a count-based window has no impact on our approach. Hence, our approach models data stream as a sequence of sliding windows in which the most recent window includes the triples having the most recent N timestamps.

The Copenhagen International Airport is testing a mechanism for monitoring travelers' movements in real-time by following their Wi-Fi trails with the goals of improving airport design and security, directing the flow of travelers, and providing customized services to travelers [30]. Yet, disclosing the raw trajectory stream to some third-party service provider, such as an airline company or an outsourced security firm, may compromise the travelers' privacy. The following example illustrates two types of privacy attacks that an adversary can carry out by having access to the data stream.

Example 1.1. Table 1 shows the trajectories of eight travelers sorted by their *ids*. For simplicity, this example considers timestamps 1–4; however, in reality, timestamps continue indefinitely. Let us assume that sensitive information is being collected from travelers along with trajectories. The sensitive information is displayed in the sensitive attribute *sen _ att*. A potential sensitive attribute could be *Disability* where travelers with *Epilepsy*, for instance, may require special attention to facilitate their journey. The data holder (the airport) can specify a set of sensitive values from the sensitive attributes. Upon publishing the anonymized data, sensitive values should not be associated with their respected travelers. Suppose s_1 is the only sensitive value in this example.

Let the size of the sliding window be $N = 3$. The first window $W_{1\rightarrow 3}$ includes doublets with timestamps 1–3, as indicated by the dashed box in Table 1. As the window slides with step size $= 1$, the second window $W_{2\rightarrow 4}$ now includes doublets with timestamps 2–4 with no traces of doublets having timestamp 1. We note that the absence of doublets within a given window (the empty spots in Tables 1 and 2) indicates no change in a traveler's location.

Suppose an adversary has access to the trajectory stream in the form a sliding window, as in Table 1. It is possible to identify a target victim's trajectory and/or sensitive value by performing the following privacy attacks.

Identity linkage takes place when the collected trajectories contain a sequence of doublets with a rare appearance. This allows an adversary to uniquely identify a target victim. For example, suppose that the current window is $W_{2\rightarrow 4}$, and that an adversary knows that a target victim has visited location e at timestamp 4. $W_{2\rightarrow 4}$ contains only one trajectory ($id = 8$) with doublet $e4$. Hence, the adversary is able to learn the victim's other visited locations and sensitive value.

Attribute linkage takes place if there is a group of records, sharing the same sequence of doublets, that contains infrequent sensitive values. These values can be associated with their pertinent individuals with high confidence. This type of privacy attacks is also known as homogeneity attack [24,25]. Suppose that an adversary knows that a target victim has visited locations b and d at timestamps 2 and 4, respectively. $W_{2\rightarrow 4}$ shows that one of two records that contain $\langle b2 \rightarrow d4\rangle$ has the sensitive value s_1. Hence, the adversary is able to infer that the target victim has s_1 with 50% confidence. □

Table 1
Raw sliding windows, $W_{1\rightarrow 3}$ and $W_{2\rightarrow 4}$, on trajectory stream.

id	Timestamps				sen_att
	1	2	3	4	
1		*b*2	*c*3	*d*4	s_1
2	*a*1	*f*2	*c*3	*d*4	s_2
3		*b*2	*c*3	*d*4	s_3
4	*a*1	*f*2	*c*3		s_4
5		*b*2	*c*3		s_5
6			*c*3		s_2
7		*f*2		*d*4	s_3
8			*c*3	*e*4	s_1

$W_{1\rightarrow 3}$
$W_{2\rightarrow 4}$

1.1. Challenges

Data streams are characterized by being time-variant and potentially infinite. Therefore, it is infeasible to first store the data and then anonymize. Rather, data have to be anonymized on the fly. Moreover, every possible combination of location and timestamp in trajectory data forms a distinct dimension [42]. This characteristic is referred to as *the curse of high dimensionality* [2]. For example, if the airport in Example 1.1 contains 200 hotspots (access points), then monitoring travelers' movements over the period of 60 min will result in 12,000 dimensions. Consequently, applying anonymization methods based on quasi-identifier attributes [33] will impose too much data loss, rendering the anonymous data useless for any data analysis.

1.2. Contributions

To the best of our knowledge, this is the first work to anonymize high-dimensional trajectory stream. We summarize our contributions as follows. First, to address the transient nature of trajectory streams, we propose an anonymization method based on a *dynamically* updated sliding window, where trajectories are modeled as a prefix tree to ensure compactness and efficient data retrieval. Second, a naïve solution to anonymizing a trajectory stream is by simply anonymizing every single window independently. To avoid this redundancy, we identify some important properties in trajectory streams, and utilize these properties to efficiently anonymize trajectories in a sliding window by *incrementally* updating the prefix tree. Third, our proposed method guarantees that the output anonymous trajectory stream satisfies *LKC*-privacy [27,28]. *LKC*-privacy is a flexible privacy model that has proven efficient in handling high-dimensional data [28]. Fourth, our experimental evaluation on simulated and real-life data sets demonstrates that our proposed algorithm is capable of handling large volume of trajectory streams without compromising their utility.

The rest of the paper is organized as follows. Section 2 provides a literature review. Section 3 formally defines the privacy attacks and *LKC*-privacy model. Section 4 discusses our proposed algorithm and data structure. Section 5 presents experimental evaluations. Section 6 concludes the paper.

2. Related work

In the area of privacy-preserving data publishing [15], researchers proposed various privacy models to thwart identity and attribute linkages. Different anonymization and sanitization methods were developed to transform raw data to an anonymous version.

Table 2
Anonymous sliding window $W^{\prime}_{2\rightarrow 4}$ for $L = 2, K = 2, C = 40$ %.

id	Timestamps			sen_att
	2	3	4	
1		*c*3	*d*4	s_1
2	*f*2	*c*3	*d*4	s_2
3		*c*3	*d*4	s_3
4	*f*2	*c*3		s_4
5		*c*3		s_5
6		*c*3		s_2
7	*f*2		*d*4	s_3
8		*c*3		s_1

In the context of publishing relational data, some attributes known as *quasi-identifiers* (*QI*) can be used to re-identify individuals. The classical privacy model of *k*-anonymity [33] requires that every record in the table must be indistinguishable from at least $k - 1$ other records over the QI attributes. Due to the curse of high dimensionality [2], applying anonymization methods based on QI attributes [24,25] to trajectory data imposes significant data loss because every doublet of location and timestamp in a trajectory is considered to be a distinct QI attribute. In other words, the notion of a fixed set of QI attributes in trajectory data does not exist anymore because of the continuously changing reported locations and timestamps [43].

In the context of continuous data release, updated versions of a data table are published on a regular basis, e.g., every 1 week [9,40,37]. Wang et al. [37] proposed a method for anonymizing temporal data in relational format. Their method is based on temporal record relocation within a window of multiple releases. Xiao and Tao [40] considered the problem of re-publishing updated relational data tables. Their method, called *m-invariance*, is the first to address the insertion and deletion of records in the updated data table. If a record does not meet the imposed privacy requirement, counterfeit records are created in order to achieve an *m*-invariant updated table. Applying *m*-invariance to trajectory streams is not suitable due to the following reasons. First, *m*-invariance anonymizes relational data. Trajectories are high dimensional by nature; thus, applying methods based on QID attributes incurs significant data loss as explained in the above paragraph. Second, *m*-invariance does not assume the existence of data streams, which gives the advantage of not having a sharp time constraint for publishing the updated data table. In contrast, our method maintains the transient nature of streams by anonymizing and publishing newly-arrived data on the fly. Third, *m*-invariance is achieved by adding counterfeit records to the data table. On the other hand, our method maintains truthfulness because all published records belong to real-life moving individuals. This property gives more credible results when analyzing the anonymized data. In summary, continuous data release does not require publishing the data table at the time of data collection since the table does not contain "live" data. Algorithms for anonymizing continuous data are not suitable for potentially infinite streams of transient and time-critical data because these properties require dynamic and scalable processing with little time delay.

Recently, several methods have been proposed targeting anonymizing *static* trajectory (or moving objects) data [7,17,31,1,34,29,43,21]. Gidofalviet al. [17] presented the first work to perform knowledge discovery on anonymized trajectory data. They proposed a probabilistic method by which trajectories are hidden behind a series of *rectangles*. Nergiz et al. [31] presented the first work to apply *k*-anonymity on static trajectory data by means of generalization. Their enforced privacy model stipulates that every trajectory, in its *entirety*, must be indistinguishable from at least $k - 1$ other trajectories. They further proposed a method for publishing randomly reconstructed versions of anonymized trajectories. Abul et al. [1] proposed (k, δ)-*anonymity* whereby *space translation* is used to make every trajectory coexists with a minimum of $k - 1$ other trajectories within a proximity of δ. Monreale et al. [29] used *spatial generalization* by replacing each specific location point in a trajectory with the centroid of a more general geographical area covering that point. The novelty of their method lies in generating geographical areas in a dynamic fashion based on the input data set, as opposed to generating a fixed grid [43].

Pensa et al. [32] studied the problem of anonymizing *sequential data* by preserving frequent sequential patterns. The authors consider temporal sequentiality, which can be considered a simpler form of trajectory data. To account for high dimensionality in sequential data, the authors use a prefix tree to structure sequences of temporal items. The proposed method is based on *k*-anonymity, thus, thwarts only identity linkage attacks. Whereas, our method extends *k*-anonymity to thwart *both* identity and attribute linkage attacks, covering broader attack scenarios against trajectory streams. Similar to space translation [1], Pensa et al. [32] use insertion, deletion, or substitution of items to anonymize a sequence.

Methods based on probabilistic approaches [17,21], space translation [1,32], or generalization [31,29,43] degrade the utility of the published data, rendering analysis less accurate. In this paper we perform a series of *suppressions* (informally defined as the removal of specific doublets) on raw data. Suppression guarantees data truthfulness because any anonymous trajectory is a subset of its raw version.

Suppression-based approaches have been used to anonymize trajectory data before [34,12]. The privacy model in [34] assumes that different adversaries possess different background knowledge, requiring data holders to obtain all such background knowledge. We argue that collecting this information is non-trivial and impractical under normal circumstances. Therefore, our proposed method limits any adversary's knowledge about any target victim to a maximum of *L* doublets. This realistic assumption is also adopted in other works involving high-dimensional data [35,42]. More discussion on the adversary's background knowledge *L* will be provided in Section 3.1. The authors in [12] use *local suppression* to anonymize a set of trajectories *T*. While our method removes a doublet by removing *all* its instances from *T* (global suppression), local suppression removes only *some* instances from *T*. Naturally, local suppression causes less information loss by preserving more instances; however, in Section 3.3 we explain why local suppression is not feasible for our target problem.

Next, we review some of the relevant work in the context of privacy preservation in *data streams*.

Li et al. [22] target preserving individuals' privacy in numerical data streams. Dwork et al. [13] proposed a set of algorithms based on differential privacy to address some specific counting tasks. Both [22] and [13] involve noise addition, while our method preserves data truthfulness, allowing for various data mining tasks. Recently, Chan et al. [11] addressed the problem of privacy-preserving aggregation on sensitive streams. In this setting, an untrusted aggregator should not learn individuals' exact data points, rather, only estimated aggregate statistics. This goal is different from ours, as follows. The work in [11] achieves statistic aggregations while our work preserves the exact whereabouts of each moving individual. The latter approach allows for providing customized services, e.g., recommendations, tailored for each individual. Hence, our method supports more operations on the output anonymous data.

Zhou et al. [45] proposed a framework for k-anonymizing a stream of relational data. Clusters are created and filled with arriving data elements. Once a cluster contains data belonging to at least k moving individuals, the data are published in the same generalization level. To limit information loss due to generalization, both [45] and [44] incorporate a prediction mechanism of future data elements. Another approach targeting the same type of streaming data is *CASTLE* [10], a cluster-based approach. It incorporates cluster merging and splitting mechanisms based on a maximum allowable delay parameter. In *SANATOMY*, Wang et al. [38] employed anatomy to publish an ℓ-diverse data stream. They also assumed the arrival of a single data element in the stream at any given timestamp. Li et al. [23] proposed a method called *SKY* that allows a data owner to determine how much an anonymized stream deviates from its raw version. All [10,38,23] enforce a user-defined time constraint to limit the delay of the published data. We argue that imposing time constraint damages the anonymized data usefulness due to higher generalization levels incurred on nearly-expired data. We differentiate our work as follows. (a) All the above works target stream of *relational* data while we target *trajectory* data. (b) The privacy models in these works are heavily based on traditional k-anonymity while our model is more relaxed to accommodate both identity and attribute linkage attacks with a wide spectrum of adversary's knowledge. (c) The anonymous data in these works are achieved mostly by means of generalization while we perform suppression. These points have been explained earlier in this section.

3. Problem definition

A data holder is constantly collecting trajectories of moving individuals. A trajectory tr is a sequence of *triples*. A triple $\langle id_p, loc, t\rangle$ that belongs to individual $p \in P$ reports p's location loc at timestamp $t \in \mathbb{N}$, where P is the universe of moving individuals generating trajectories. We define trajectory stream as follows.

Definition 3.1. Trajectory stream

A trajectory stream $S = \left\{\langle id_1, loc_{id_1}, t_{id_1}\rangle, \langle id_2, loc_{id_2}, t_{id_2}\rangle, \ldots, \left\langle id_p, loc_{id_p}, t_{id_p}\right\rangle, \ldots\right\}$ is a continuous sequence of triples generated by every moving individual $p \in P$. ■

We assume that triples are being generated continuously; therefore, it may not be feasible to store all the data in a conventional database. Moreover, we assume that the data recipient is more concerned with recent data rather than "outdated" data. In Example 1.1, the airport manager may want to monitor the data in real-time by checking for congestions at any gate for the past 60 min and react by opening new gates or allocating additional staff. If the airport, however, is to store trajectories for later analysis, say on a weekly or monthly basis, several anonymization solutions exist for this particular problem [7].

For the aforementioned reasons, we use a time-based sliding window W to represent the most recent data in a trajectory stream S. A data holder specifies the size of W in terms of timestamps.

Definition 3.2. Sliding window

Let N be the size of a sliding window W, x be the starting timestamp, and $y = x + N - 1$ be the ending timestamp. $W_{x \to y} = \{triple \in S | x \le triple.t \le y\}$. ■

Following Definition 3.2, Table 1 shows $W_{1 \to 3}$, which starts at timestamp 1 and has a size of $N = 3$.

When a window W contains all proper triples, the next step would be to anonymize this window then publish it (Section 4) — a data recipient has only a view over stream S through a sequence of anonymized windows published one at a time, and any mining operation is performed exclusively on the most recent window. After that, the window *slides*, a process by which *outdated triples* are dropped out and *new triples* are added. When a window slides, it shifts by a certain number of timestamps determined by *step _ size*.

Definition 3.3. Outdated and new triples

From Definition 3.2, given a window size N, $x^+ = x + step\_size$, and $y^+ = y + step\_size$, we define *outdated triples* $O = \{triple \in W_{x \to y} | x \le triple.t < x^+\}$ and *new triples* $E = \{triple \in S | y < triple.t \le y^+\}$, where $E \cup W_{x \to y} = \varnothing$. ■

Definition 3.4. A slide

From Definition 3.3, when window $W_{x \to y}$ experiences a single slide, it becomes $W_{x^+ \to y^+} = (W_{x \to y} - O) \cup E$. ■

Example 3.1. Consider Table 1. Suppose that the first window is $W_{1 \to 3}$ (dashed box), $N = 3$, $step\_size = 1$, and that timestamp 4 has not appeared yet. For $W_{1 \to 3}$, $x = 1$ and $y = 1 + 3 - 1 = 3$. $W_{1 \to 3}$ is then anonymized an published. At this point, timestamp 4 appears, and since $step\_size = 1$, $O = \{\langle 2, a, 1\rangle, \langle 4, a, 1\rangle\}$ and $E = \{\langle 1, d, 4\rangle, \langle 2, d, 4\rangle, \langle 3, d, 4\rangle, \langle 7, d, 4\rangle, \langle 8, e, 4\rangle\}$. The new window is now $W_{2 \to 4}$. E is now a subset of $W_{2 \to 4}$. However, a set union between E and $W_{1 \to 3}$ remains an empty set because window $W_{1 \to 3}$ is unaware of any future data, therefore, E does not exist in this particular window. ■

We assume that the data holder publishes the moving individuals' sensitive information along with their trajectories. Therefore, we define an *object table* in which each record corresponds to a unique moving individual p and contains p's trajectory and sensitive information. More formally,

$$\left\langle id_p, tr_{x\to y}, s_1, \ldots, s_m \right\rangle,$$

where id is a record identifier, $tr_{x \to y}$ is p's trajectory of doublets corresponding to $W_{x \to y}$, and $s_i \in SA_i$ are values from sensitive attributes $SA_1, \ldots, SA_m$ where $m \geq 1$. For the sake of simplicity, we consider $m = 1$ throughout our examples; i.e., object tables contain only one sensitive attribute. Without loss of generality, our method can handle multiple sensitive attributes. Recall that a *doublet* is nothing but *loc* and t from the triple to which it corresponds. We use this term whenever the focus is rather on trajectories themselves, regardless to whom they belong.

3.1. Privacy threats

A data recipient has access to the most recently updated sliding window W. The published window includes recent moving individuals' trajectories along with their sensitive information. We mentioned in Section 1 that adversaries are data recipients who attempt to identify a target victim's trajectory tr and/or sensitive value s. We assume that an adversary possesses a *subsequence* of the victim's trajectory. We denote this subsequence by κ, and we call it the *adversary's background knowledge*. We also assume that κ has a maximum size of L doublets, that is,

$$\kappa = \langle (loc_1 t_1) \to \ldots \to (loc_z t_z) \rangle,$$

where $z \leq L$. We note that given a sliding window with size N, $L \leq N$. κ is a subsequence of a victim's trajectory tr if each and every doublet in κ also exists in tr following the same order.

Obtaining the background knowledge κ from real-time trajectories is feasible due to the following two reasons. First, a relatively long window allows a stalking adversary to gather a considerable amount of data about a target victim. Second, an adversary may learn a victim's trends and habits (e.g., route from home to office), which are highly likely to appear in several windows to come.

As we mentioned in Section 2, the concept of estimating the maximum length of adversary's background knowledge, has been previously proposed in the literature [42,35]. Those works proposed privacy models that take into consideration the attacker's "power". The "power" of any attacker is the maximum number of items known by the attacker about any transaction, and is denoted by L. We use a privacy model (Section 3.2) that shares this same concept.

It is possible for the data holder to estimate how much background knowledge the attacker can acquire based on the effort needed to obtain such knowledge. If acquiring background knowledge about target victims is deemed relatively easy, then the data holder can increase L to its maximum value. We note that the worst case about setting L is not the entire trajectory size, rather, the window size N because the size of real-time trajectories is unknown to our anonymization algorithm.

Given an object table T that contains the trajectory of a target victim, κ could be found in a group of trajectories in T. We denote the group of records containing κ by $G(\kappa)$, and the group size, i.e., number of records in $G(\kappa)$, by $|G(\kappa)|$. κ may match to only a few records in T. That is, if $|G(\kappa)|$ is very small, the adversary might be able to uniquely identify the victim's record, thus, learning his/her other visited locations and sensitive value.

Example 1.1 demonstrates that given $W_{2\to 4}$ in Table 1, and given that $\kappa = \langle e4 \rangle$, an adversary is able to uniquely identify the victim's record ($id = 8$) since $|G(\langle e4 \rangle)| = 1$. We refer to this type of attack as *identity linkage*.

If the sensitive values in $G(\kappa)$ are not diverse enough, an adversary might be able to infer the victim's sensitive value s with high *confidence*. We denote the probability of inferring the victim's sensitive value s from $G(\kappa)$ as follows:

$$Conf(s|G(\kappa)) = \frac{|G(\kappa \cup s)|}{|G(\kappa)|},$$

where $G(\kappa \cup s)$ is the group of records within $G(\kappa)$ containing both the subsequence κ and the sensitive value s. In Example 1.1, if $\kappa = \langle b2 \to d4 \rangle$, then $Conf(s_1|G(\langle b2 \to d4 \rangle)) = 1/2 = 50\,\%$.

3.2. Privacy model

We use *LKC*-privacy model [27,28,14] to transform raw window $W_{x \to y}$ to an anonymous version $W'_{x \to y}$ such that the published object table T thwarts identity and attribute linkage attacks. The reason for choosing *LKC*-privacy is its flexibility, demonstrated as follows. (a) By changing the input parameters, *LKC*-privacy can metamorphose into k-anonymity or an instance of ℓ-diversity (Section 3.3). This property also implies that if no sensitive information is involved in the process of data publishing, *LKC*-privacy can still function properly. (b) A larger L provides more protection against adversaries with longer background knowledge.

We note that unlike the work in [20], *LKC*-privacy does not require the data holder to specify a set of predefined subsequences in S. Rather, we explore the entire domain of *loc* and t with no restrictions on the number of triples in any window.

In a given window, *LKC*-privacy ensures that any subsequence of size up to L appears at least K times and the probability of inferring victims' sensitive values is at most C. We formalize this model as follows.

Definition 3.5. *LKC*-privacy

Given a set of sensitive values *Sen*, a positive integer L, an anonymity threshold $K \geq 1$, and a confidence threshold $0 \leq C \leq 1$, a window $W_{x \to y}$ satisfies *LKC*-privacy iff for any subsequence q with $|q| \leq L$, $|G(q)| \geq K$ and $Conf(s|G(q)) \leq C$ for any $s \in Sen$. ■

Sen is defined by the data holder. In Example 1.1, s_1 is the only sensitive value in the sensitive attribute. If a stream does not contain any sensitive values, then $Sen = \emptyset$. If a data holder wants to ignore all sensitive values completely, then assigning $C = 100\,\%$ would let any subsequence q satisfy the condition $Conf(s|G(q)) \leq C$. Furthermore, should certain locations be deemed sensitive, the data holder can include such locations in *Sen*. This hallmark in our privacy model gives the data holder further flexibility in terms of privacy requirements.

Given an anonymous window, denoted by $W_{x \to y}'$, that satisfies *LKC*-privacy, the probabilities of successful identity and attribute linkage attacks are $\leq 1/K$ and $\leq C$, respectively. We note that the same degree of data utility is achieved from both $W_{x \to y}'$ and its static version, i.e., static trajectories.

3.3. Problem statement

Our proposed method achieves anonymity through *suppression* by efficiently removing selected doublets from raw window $W_{x \to y}$ with the goal of preserving its utility. We perform *global suppression*: all instances of the selected doublet will be removed from $W_{x \to y}$. For example, the anonymous window $W_{2 \to 4}'$ depicted in Table 2 is the result of suppressing all instances of doublets $b2$ and $e4$ from the raw window $W_{2 \to 4}$ in Table 1.

Suppression does not require a predefined taxonomy tree, which is essential for performing *generalization* and may not be conveniently available in real-life scenarios. Moreover, suppression targets specific doublets, resulting in more useful data for better analysis. Generalization, on the other hand, is imposed on all the sibling nodes in any subtree in the taxonomy tree, causing unnecessary information loss.

Although applying *local/multidimensional generalization* achieves less information loss than suppression, the former methods suffer from significant limitations. Existing statistical tools, such as SAS and SPSS, are unable to handle data anonymized by local/multidimensional generalization due to the complexity of performing analysis on overlapping sub-domains [41]. Furthermore, even though sibling nodes are not affected by locally generalizing value v to u, most standard data mining methods treat v and u as two independent values, which is not the case [15]. For instance, mining classification rules may create fuzzy rules; the following two rules make it ambiguous to classify a new v: $v \to class1$ and $u \to class2$.

Applying techniques based on local suppression is not feasible even though such techniques may cause less information loss than global suppression. We use global suppression because it takes advantage of the monotonicity property of Apriori. In contrast, this property does not hold for local suppression because the number of violations does not decrease monotonically with respect to local suppressions. For example, suppose a trajectory table contains the sequence $a1 \to b2$ with support $= 2$. Let $K = 2$ and $L = 2$. $a1 \to b2$ is not a violation. If $b2$ was locally suppressed from one record only, then the resulting sequence $a1 \to b2$ becomes a new violation. As a result, applying local suppression requires an extra step to check for newly generated violations. The authors in [12] showed that such extra step is computationally costly. Therefore, to accommodate the transient, real-time trajectories, we cannot afford local suppression for trajectory streams.

Definition 3.6. Anonymizing trajectory stream

Given a trajectory stream S, a sliding window $W_{x \to y}$, and an *LKC*-privacy requirement, the problem is to efficiently publish a sequence of anonymized sliding windows over S such that suppressions are minimized. ■

k-Anonymity and confidence bounding are special cases of *LKC*-privacy [14]. According to [26] and [36], achieving optimal k-anonymity and optimal confidence bounding is NP-hard. It follows that achieving optimal *LKC*-privacy, i.e., performing the least number of suppressions in any window, is also NP-hard. As a result, anonymizing a sequence of windows over S with minimum number of suppressions is NP-hard. In the next section we propose a greedy algorithm that obtains a sub-optimal solution.

4. Anonymization algorithm

In this section, we present Incremental Trajectory Stream Anonymizer (ITSA), our algorithm for incrementally anonymizing every window $W_{x \to y}$ on S by means of suppression. We identify all subsequences in $W_{x \to y}$ that *violate* a given *LKC*-privacy requirement. A window is anonymous when it contains no violations. We also present the dynamic tree structure of window for efficient updates and data retrieval.

4.1. Incremental identification of violations

In order to publish an anonymous window, we need to make sure it does not contain any violation. We formally define a violation as follows.

Definition 4.1. Violation

Assume a given *LKC*-privacy requirement and a sliding window $W_{x \to y}$ over S. If any subsequence q in $W_{x \to y}$, with $1 \le |q| \le L$, has $1 \le |G(q)| < K$ and/or $Conf(s|G(q)) > C$, then q is a violation. ■

A violation can be any possible combination of doublets in $W_{x \to y}$ that does not adhere to *LKC*-privacy. For example, $\langle b2 \to d4 \rangle$ in $W_{2 \to 4}$ (Table 1) is a violation, as Example 1.1 demonstrates. Eliminating all violations transforms $W_{x \to y}$ to an anonymous version $W'_{x \to y}$ that protects against privacy threats. Section 4.3 discusses how violations are efficiently suppressed.

If a window contains z distinct doublets, the total number of possible sequences to be checked is $2^z - 1$. Due to this exponential growth of candidate subsequences, we adopt the monotonicity property of Apriori [3] and only identify *critical violations* instead of exhaustively finding all violations in a window. A critical violation is defined below.

Definition 4.2. Critical violation

A sequence v is a critical violation iff v is a violation and none of its subsequences is a violation. ■

If v has at least one subsequence v' that violates a given *LKC*-privacy requirement, then v is a violation but not a critical violation.

Example 4.1. Given window $W_{2 \to 4}$ in Table 1, let $L = 2$, $K = 2$, $C = 40\,\%$, and s_1 be a sensitive value. $q_1 = \langle b2 \to c3 \rangle$ is not a violation because $|G(q_1)| = 3 \ge 2$ and $Conf(s_1|G(q_1)) = 33\,\% \le 40\,\%$. $q_2 = \langle b2 \to c3 \to d4 \rangle$ is a violation because, although $|G(q_2)| = 2 \ge 2$, $Conf(s_1|G(q_2)) = 50\,\% > 40\,\%$. On the other hand, q_2 is not a critical violation because one of its subsequences, $q'_2 = \langle b2 \to d4 \rangle$, is a violation. q'_2 itself, however, is a critical violation because neither $b2$ nor $d4$ is a violation. ■

From Definition 4.2, we make the following observation about anonymizing a raw window.

Observation 4.1. Removing all critical violations from a raw window $W_{x \to y}$ transforms it to an anonymous version $W'_{x \to y}$, w.r.t. a given *LKC*-privacy requirement, that contains no violations. ■

Proof 4.1. Let v_1 be a critical violation due to $|G(v_1)| < K$. Then any supersequence v''_1 of v_1 is a violation because $|G(v''_1)| \le |G(v_1)| < K$. However, any subsequence v'_1 of v_1 is not a violation because $|G(v'_1)| \ge q|G(v_1)|$. Therefore, if v_1 satisfies *LKC*-privacy then v_1 also satisfies *L'KC*-privacy, for $L' < L$.

Let v_2 be a critical violation due to $Conf(s|G(v_2)) > C$. A supersequence v''_2 of v_2 may or may not be a violation because there exists no relation between $Conf(s|G(v_2))$ and $Conf(s|G(v''_2))$. Therefore, according to Definitions 3.5 and 4.2, v''_2 will not be in any $Cand_i$. ■

We iteratively generate the set of all i-size candidate subsequences, $Cand_i$, by self-joining non-violating subsequences in $Cand_{i-1}$. Every subsequence $q \in Cand_i$ is checked against the given privacy requirement. If q is a (critical) violation, it is removed from $Cand_i$. To mitigate information loss due to suppression, Section 4.3 shows that we do not actually need to remove all occurrences of a critical violation v from $W_{x \to y}$; rather, removing specific doublets in v is sufficient.

We identify below certain properties in a trajectory stream S and integrate them in building an efficient algorithm for incrementally anonymizing a sliding window over S. Recall that the subsequent window of $W_{x \to y}$ is denoted by $W_{x^+ \to y^+}$ (Definition 3.4).

Property 4.1. When anonymous window $W'_{x \to y}$ slides, $W'_{x \to y} - O$ incurs no violations. ■

Removing the set of outdated doublets O from anonymous window $W'_{x \to y}$ does not create violations. This is because: (a) All doublets in O are globally suppressed from $W'_{x \to y}$. (b) According to Definition 3.5, all subsequences of size up to L satisfy the privacy requirement.

Property 4.2. If subsequence q in anonymous window $W'_{x \to y}$ satisfies *LKC*-privacy, then q also satisfies *LKC*-privacy in any subsequent window that contains q. ■

Let q be a non-violation in anonymous window $W'_{x \to y}$. If subsequent raw window $W_{x^+ \to y^+}$ contains q, then q is still a non-violation.

Property 4.3. When anonymous window $W'_{x \to y}$ slides, $\cup E$ may create new violations. ■

Let q be a non-violation in anonymous window $W'_{x \to y}$. Adding the set of new doublets E to the subsequent raw window $W_{x^+ \to y^+}$ creates new combinations of doublets. Consequently, $q \cup e$, where e is a subsequence from E, may or may not be a violations.

Algorithm 1 (ITSA): This algorithm runs every time the window slides over a trajectory stream S. Suppose that anonymous window $W'_{x \to y}$ of size N has just been published. Line 2 slides the window by *step _ size* timestamps. Outdated doublets O drop out and

new doublets E arrive. The updated window, $W_{x^+ \to y^+}$, contains raw data. Phase 1 anonymizes E, Phase 2 obtains all critical violations V, and Phase 3 removes V from the raw window. Finally, anonymous $W'_{x^+ \to y^+}$ is published.

Algorithm 1. Incremental Trajectory Stream Anonymizer (ITSA)

Input: Anonymous window $W'_{x \to y}$. Trajectory stream S.
Input: Thresholds L, K, C, and sensitive values Sen.
Input: Window size N and $step\_size$.
Output: New anonymous window $W'_{x^+ \to y^+}$.

1: **while** S exists and $L \leq N$ **do**
2: slide $W'_{x \to y}$ over S to get raw $W_{x^+ \to y^+}$.
3: obtain E;
/*Phase 1*/
4: $E' = \text{AnonymizeNew}(L, K, C, step\_size, E)$;
5: $Cand_1 =$ all unique doublets in $W_{x^+ \to y^+}$. $Cand_1 \cap E' = \emptyset$;
6: $Cand_2 = Cand_1 \bowtie E'$;
7: let $V = \emptyset$;
8: let $i = 2$;
/*Phase 2*/
9: **repeat**
10: **for all** $q \in Cand_i$ **do**
11: **if** $\exists v \in V$ s.t. $v \subseteq q$ **then**
12: remove q from $Cand_i$ and add q to V;
13: **else**
14: **if** $|G(q)| < K$ or $Conf(s|G(q)) > C, \forall s \in Sen$ **then**
15: remove q from $Cand_i$ and add q to V;
16: **end if**
17: **end if**
18: **end for**
19: **if** $++i \leq L$ **then**
20: $Cand_i = Cand_{i-1} \bowtie Cand_{i-1}$.
21: **end if**
22: **until** $i > L$ or $Cand_i = \emptyset$
/*Phase 3*/
23: $Win = \text{findWinners}(V)$;
24: **for all** $w \in Win$ **do**
25: remove all instances of winner doublet w from $W_{x^+ \to y^+}$;
26: **end for**
27: **Publish** $W'_{x^+ \to y^+}$;
28: **end while**

Phase 1. As a preprocessing step, we first anonymize E. Any subsequence e from E is a special case to which Property 4.3 applies. Consequently, any subsequence e with $|e| \leq L$ is checked. The maximum length of any e is equal to $step\_size$, which is relatively small compared to window size N. Procedure 2 applies LKC-privacy on raw E. Critical violations found in E are removed from $W_{x^+ \to y^+}$. Line 6 creates the 2-size candidate set $Cand_2$ by self-joining $Cand_1$ and E'. This process is demonstrated below.
Phase 2. We identify all critical violations V in $W_{x^+ \to y^+}$. Thanks to Properties 4.2 and 4.3, we check only subsequences that contain at least one doublet from E. This phase iteratively generates $Cand_i$ (Line 20) to check for critical violations. The iteration is terminated when i exceeds L or $Cand_i$ cannot be generated. Two subsequences, $q_y = \langle (loc_1^y t_1^y) \to \ldots \to (loc_{i-1}^y t_{i-1}^y) \rangle$ and $q_z = \langle (loc_1^z t_1^z) \to \ldots \to (loc_{i-1}^z t_{i-1}^z) \rangle$, can be self-joined only if all doublets except the last (the one having t_{i-1}) are identical in both subsequences, and $t_{i-1}^y < t_{i-1}^z$. The resulting sequence is $l\langle (loc_1^y t_1^y) \to \ldots \to (loc_{i-1}^y t_{i-1}^y) \to (loc_{i-1}^z t_{i-1}^z) \rangle$. Lines 10–12 ensure that a candidate subsequence q is not a super sequence of a violation in V.
Phase 3. We remove all critical violations from $W_{x^+ \to y^+}$. Line 23 calls Procedure 3 in order to suppress only selected doublets in V from $W_{x^+ \to y^+}$. Finding these doublets, referred to as *winner doublets Win*, for suppression is motivated by the goal of incurring less impact on the data utility. This process is detailed in Section 4.3. Line 27 publishes the anonymous window $W'_{x^+ \to y^+}$ after all winner doublets have been removed.

Example 4.2. We continue from Example 4.1. As $W'_{1 \to 3}$ slides, $E = \{\langle 2, d, 4 \rangle, \langle 3, d, 4 \rangle, \langle 7, d, 4 \rangle, \langle 8, e, 4 \rangle\}$. First, Procedure 2 in Phase 1 determines that $e4$ is a violation and, thus, suppresses $e4$ from E resulting in E'. Second, Phase 2 generates and tests every candidate

subsequence q containing existing and new doublets in $W_{2\to 4}$. $Cand_2 = \{\langle b2 \to d4\rangle, \langle c3 \to d4\rangle, \langle f2 \to d4\rangle\}$. $q_1 = \langle b2 \to d4\rangle$ is a critical violation because C on $f(s_1|G(q_1)) = 50\,\% > 40\,\%$. To demonstrate self-joining, let $L = 3$. Then, $Cand_3 = \{\langle f2 \to c3 \to d4\rangle\}$. Note that $Cand_3$ does not include $\langle b2 \to c3 \to d4\rangle$ because it is a supersequence of q_1. ■

Procedure 2. AnonymizeNew()

Input: New doublets E, and $step\_size$.
Input: Thresholds L, K, C, and Sensitive values Sen.
Output: Anonymous E'.

```
1: let V = ∅;
2: for (i = 1, i ≤ min(L, step_size), ++i) do
3:    generate every possible i-size sequence e from E;
4:    scan E once to find |G(e)| and Conf(s|G(e)) for any s ∈ Sen;
5:    if |G(e)| < K or Conf(s|G(e)) > C for any s ∈ Sen then
6:       add e to V;
7:    end if
8: end for
9: if V = ∅ then
10:   return E as anonymous E';
11: else
12:   Win = findWinners(V);
13:   for all w ∈ Win do
14:      suppress w from E;
15:   end for
16:   return E' as anonymous version of E;
17: end if
```

4.2. Sliding window as a tree

When the sliding window moves, a relatively small fraction of the data is added/dropped while the remaining larger portion does not change (overlapping data). Our proposed method efficiently handles this transition. Moreover, our method allows the sliding window to add/remove trajectories of joining/leaving individuals. ITSA entails adding, removing, and searching sequences of doublets. We explain below how our method facilitates these operations on the window.

We use a *trie* structure to implementing sliding window. A trie is a type of trees where each node is a prefix to all its descendants. The trie is created once (first window), then is dynamically updated upon every window slide. Our tree structure is reminiscent to the FP-tree structure introduced in FP-growth, a method for mining frequent patterns [19].

Definition 4.3. Trie

A trie, $R = (Nodes, Edges, root)$, of trajectories in $W_{x\to y}$ consists of a collection of nodes, edges, and a root node. Every node $n \in Nodes$ contains a doublet b from $W_{x\to y}$ and a prefix count, *count*. The *count* of node *n* stores the number of distinct trajectories containing the prefix subsequence in the unique *root-to-node* path. The sequence of doublets on a *root-to-leaf* path constitutes a full trajectory. The root node contains only a *count*, which is the total number of trajectories. ■

Procedure 3. findWinners()

Input: Critical violations V.
Output: Winner doublets Win.

```
1: initialize score table;
2: let Win = ∅;
3: repeat
4:    choose winner doublet w with the highest Score;
5:    add w to Win;
6:    remove w from score table and update it accordingly;
7: until score table is empty
8: return Win;
```

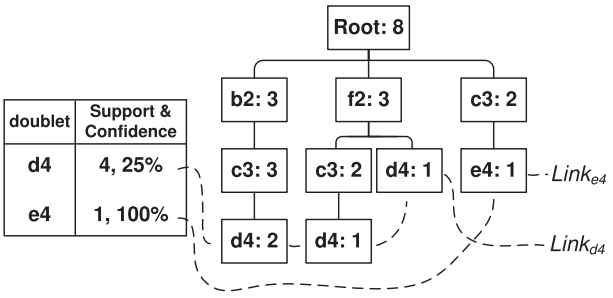

Fig. 2. Raw window $W_{2\rightarrow 4}$ in Table 1 structured as a trie.

The trie structure has a significant impact on storage space by allowing a concise view over trajectories. For example, raw window $W_{2\rightarrow 4}$ in Table 1 contains three instances of the sequence $\langle b2 \rightarrow c3\rangle$. Fig. 2 is a trie of window $W_{2\rightarrow 4}$. Only two nodes are used to represent all the three instances of this sequence.

Nodes containing new doublets from E' are added as leaves, and nodes on the higher levels are removed when removing outdated doublets O. Fig. 3 shows anonymous window $W'_{2\rightarrow 4}$, where nodes containing $a1 \in O$ and $b2, e4 \in Win$ are removed.

When generating candidate subsequences from trie R, a subsequence $q \in Cand_i$ may appear in several branches in R. This is challenging when counting the total number of trajectories containing q (i.e., the *support* of q). A naïve way is exhaustively search R. Instead, we use a virtual line, called *Link*, to connect all nodes containing the same doublet in R. Figs. 2 and 3 show $Link_{e4}$ and $Link_{d4}$.

Definition 4.4. Link

Given a trie R, $Link_b$ is a sequence of positions that belong to all the nodes containing doublet b in R. ■

Finding the support and confidence of a doublet b is achieved by traversing $Link_b$. Simply adding up the *counts* of all the nodes on $Link_b$ yields the support of b, i.e., $|G(b)|$. To calculate $Conf(s|G(b))$, we use a separate *sensitive count* for every unique sensitive value in *Sen*. Summing up the sensitive counts pertaining to sensitive value s yields $|G(b \cup s)|$, which is used along with the previously found support to calculate the confidence.

Example 4.3. Fig. 2 shows the trie R of the raw window $W_{2\rightarrow 4}$. New nodes of $d4, e4 \in E$ are added as leaves. The node of $d4$ in the sequence $q_1 = \langle b2 \rightarrow c3 \rightarrow d4\rangle$ has $count = 2$. This indicates that two instances of $d4$ preceded by $\langle b2 \rightarrow c3\rangle$ appear in two trajectories in $W_{2\rightarrow 4}$. These two trajectories, as shown in Table 1, are associated with $s_1 \in Sen$ and s_3. Therefore, the node of $d4$ has sensitive count $= 1$. Hence, $Conf(s_1|G(q_1)) = 1/2 = 50\,\%$. ■

From the above discussion, we can find the support of a given sequence $q = \langle b_1 \rightarrow b_2 \rightarrow b_3 \rightarrow \ldots \rightarrow b_{|q|}\rangle$ in a given trie R as follows. Since every node in R is a prefix to all its descendants, it follows that $count_{b_i} \leq count_{b_j}$, where $j < i$. This finding implies that $count_q = count_{b_{|q|}}$. To find the support of q, we need to trace $Link_{b_{|q|}}$ to check for other instance of q in other branches of R. The support of $q = |G(q)| = \sum count_{b_{|q|}}$, given q exists in these branches. Calculating the confidence follows the same reasoning.

Example 4.4. Let us review q_1 in Example 4.3. The leaf node of $d4$ has $count = 2$. Consequently, any subsequence of q_1 containing $d4$ will have its $count = 2$. Let $q_1' = \langle c3 \rightarrow d4\rangle \subseteq q_1$. q_1' has two instances in two separate branches in R. The support of $q_1' = \sum count_{d4} = 2 + 1 = 3$. ■

As we pointed out earlier, the trie is created once (first window). Only at this one point does Procedure 2 run on an entire window since all doublets in the first window are new, where $step\_size = N$. Afterwards, the trie is adjusted dynamically as new doublets arrive and outdated doublets drop out. Hence, only new doublets E are read from S.

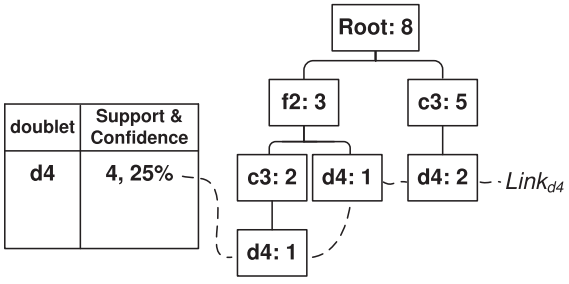

Fig. 3. Anonymous window $W_{2\rightarrow 4}'$ in Table 2 structured as a trie.

4.3. Suppression

After identifying all critical violations V in raw window $W_{x \to y}$ (Phase 2 in Algorithm 1), we need to remove them from $W_{x \to y}$. A naïve approach is to remove every instance of a critical violation $v \in V$. Although this approach will result in an anonymous window that satisfies *LKC*-privacy, the incurred data loss (referred to it as *distortion*) would be significantly high. Instead, we propose a greedy procedure that selects certain *winner* doublets from V for suppression.

When a doublet w is chosen for suppression, w is globally suppressed. That is, all instances of w are removed from $W_{x \to y}$. Global suppression does not increase the maximum probability of a successful attack (identity and attribute linkages) because any proper subsequence in $W'_{x \to y}$ still satisfies the imposed *LKC*-privacy requirement.

$$Score(d) = \frac{PrivGain(d)}{InfoLoss(d)} \tag{1}$$

Removing a violating sequence v implies that: (a) we are gaining privacy by removing an attack channel, and (b) we are inflicting data distortion due to permanently removing doublets from the original trajectories in $W_{x \to y}$. Hence, we adopt a greedy function that attempts at removing all violating sequences in V by suppressing selected winner doublets, *Win*. We apply Eq. (1) to every distinct doublet b in V. $PrivGain(b)$ is the number of critical violations containing b, and $InfoLoss(b)$ is the support of b in the current raw window. The doublet with the highest *Score* is labeled a winner doublet w and is added to *Win*.

Procedure 3 finds the winner doublets *Win* from the set of all critical violations V produced by Phase 2 in Algorithm 1. Line 1 creates the score table and fills in the values of $PrivGain(b)$ and $InfoLoss(b)$ pertaining to every unique doublet b in V. Lines 3–7 iterate the score table to find a winner doublet. The purpose is to find a doublet that its removal would result in the removal of a maximum number of critical violations yet causing least distortion. Therefore, Procedure 3 selects the doublet w that has the highest *Score*. Then, w is added to the set of winner doublets *Win*. After that, Procedure 3 updates the score table by removing w and adjusting the value of $PrivGain(b)$ of every doublet that coexists with w in the same critical violation. If b no longer exists in V, b's entry is removed from the score table. Line 8 returns *Win* containing all winner doublets to be suppressed from raw $W_{x \to y}$.

4.4. Complexity analysis of ITSA

The algorithm starts by sliding the window over stream S (Line 2). This process requires reading the set of new doublets E from S and inserting the new subsequences into the window. The former step is proportional to the total number of individuals, $|P|$. The latter step costs $O(N.|\mathcal{U}|.|e|)$, where $\mathcal{U}$ is the universe of locations, and e is a new subsequence from E and $|e| \leq step\_size$.

We explain the complexity of generating $Cand_i$ (Phase 2). Let D be the set of all distinct doublets in a given window. $Cand_i$, where $2 \leq i \leq L-1$ (Property 4.3), is a set of i-combinations from D. We can approximate the size of a candidate set $|Cand_i| \approx |D|^i$ due to the following two reasons. First, $Cand_i$ is achieved by self-joining the sequences in $Cand_{i-1}$ having the same prefix. Second, a sequence q is removed from $Cand_{i-1}$ if q is a supersequence of a previous critical violation. Therefore, increasing i does not have a significant effect on the size of $Cand_i$, because it is less likely to find long sequences sharing the same prefix. However, the worst-case scenario would be $|Cand_i| \approx |D|^{L-1}$, knowing that $L \leq N$. Hence, Phase 2 is bounded by $O(|D|^L.|Link_e|)$, where $|Link_e|$ is the number of branches containing the full sequence q in the current trie. Likewise, Procedure 2 in Phase 1 generates $Cand_i$. However, for any sequence q in this case, $|q| \leq min(L, step\_size) \leq N$.

To search for a single winner doublet w (Phase 3) in a trie R, we only need to follow $Link_w$. When updating R after deleting a single w node, the child nodes of w go up by one level. Nodes containing the same doublet are merged. Any level contains at most $= (N - Level + 1).|\mathcal{U}|$ child nodes.

In summary, the most costly operation in ITSA is candidate sets generation, described in Phase 2. The complexity of ITSA is dominated by attacker's knowledge L: $O(|D|^L)$, where $L \leq N$. Practically, Phase 2 is most likely to terminate in early iterations due the aforementioned reasons.

4.5. Discussion

We discuss two types of privacy attacks on the published data. These attacks differ from the privacy attacks we introduced in Section 1. Furthermore, we discuss the situation where adversary's knowledge covers more than one published window.

4.5.1. Attacks

The first attack is called the *minimality attack* [39]. It stems from knowing that a certain anonymization method does not anonymize data beyond a *minimum* point at which the data satisfy a given privacy requirement. Minimality provides less data distortion, allowing for better data utility. This attack is based on the ability of an adversary to reconstruct raw data from its anonymized version. Data reconstruction is made possible when there exists a public data set with matching quasi-identifier attributes. From the reconstructed data, the adversary would be able to narrow down to the portions of the data that contain potential violations. However, our method produces anonymized trajectories that are subsets of their pertinent raw versions by a sequence of suppressions. By looking at an anonymized trajectory, it is impossible to conclude if suppression took place and on which doublets. Also, we assume raw data only show doublets when a traveler changes locations. For example, Record # 8 in Table 2 contains only $c3$. An adversary would not be able to tell if two doublets at timestamps 2 and 4 were suppressed or the traveler stayed at location c at timestamp 4.

The second attack is *correlation attack*. It takes place when a doublet is widely known by data recipients to be highly associated with a sequence of other doublets. As an example, most drivers in area a taking highway h cross over bridge r to get to mall m. If the sequence $\langle a1 \rightarrow h2 \rightarrow m4 \rangle$ exists in the published data, then an adversary will be able to deduce the missing doublet of the target victim, $r3$. We point out that if such association exists (i.e., high confidence), it is unlikely that ITSA would suppress the recurring doublet since it would have a high support. Moreover, in this paper, we assume that locations are not considered to be sensitive information. If locations are sensitive, some methods already exist that incorporate this assumption in their solution [34].

4.5.2. Adversary's knowledge

The sliding window contains recent whereabouts of the moving individuals, since recent movements are most useful for data mining tasks. However, if adversary's knowledge spans multiple windows, the data holder needs to adjust the window size so that a wider range of trajectories is anonymized.

This solution, though simple and practical, suffers from two limitations. First, it is not always trivial to deduce the length of the adversary's knowledge, L. Second, from Section 4.4, we can see that larger L incurs higher complexity. To circumvent these drawbacks, in a future work we will consider anonymizing sequential windows so that recent windows are not published independently. In other words, ITSA will still publish one anonymized window at a time, but recent windows will cover wider adversary's knowledge. One way to accomplish this goal is to maintain a buffer containing count statistics of the most recent n windows. The algorithm would then anonymize the current window based on the information available about recently published anonymized windows.

5. Performance analysis

We implemented our method in C++. All experiments run on an Intel Core i5 CPU with 2.4 GHz and 4 GB of RAM. We evaluate the performance of our method in terms of data utility, efficiency, and scalability.

We carry out performance evaluation using three data sets. The first one is a simulated data set called *MetroData*[1]. The second one is a real-life data set called *MSNBC*, which is publicly available at the UCI machine learning repository [4]. The third data set is generated using Brinkhoff's network-based synthetic data generator [8], and is called *Oldenburg*. *MetroData* is a simulation of the traffic routes of a large group of passengers using the public transit metro system in Montreal, Canada. Routes are generated based on the information and statistics provided in an annual report published by www.metrodemontreal.com. The generator that we built takes into consideration the actual metro map and the passenger flow rate of each station. According to the published statistics, a passenger passes through 8 stations on average.

MetroData contains trajectories generated for 100,000 passengers who use any of the metro's 65 stations over a 60-minute time period. Therefore, *MetroData* includes 100,000 records; each record belongs to a unique passenger. The total dimensionality of the data set is $65 \times 60 = 3900$ dimensions. We also assume the existence of a sensitive attribute, namely *social_status*. The sensitive attribute has five domain values, we choose *On-Welfare* to be sensitive. In this case, the set of sensitive values $Sen = \{On\text{–}Welfare\}$. All records are evenly assigned one value from the sensitive attribute.

The second data set, *MSNBC*, is a real-life web log containing visited webpages by nearly 1 million users, where every record belongs to a unique user. The data set contains 17 categories of webpages: News, Tech, Health, etc. Despite the fact that this data set contains no physical locations, it shares the same property of high dimensionality with a typical trajectory data set. Therefore, categories of webpages are treated as unique locations visited by users at non-decreasing timestamps. We also impose a sensitive attribute on the original data. The sensitive attribute contains 10 domain values, each record is randomly assigned one value. We choose two domain values to be sensitive.

The last data set, *Oldenburg*, is generated using Brinkhoff's network-based traffic generator. *Oldenburg* contains 100,000 trajectories of objects moving throughout the city of Oldenburg (Germany) over the course of 24 h. The representation of the generated trajectories has been modified to adhere to Definition 3.1. Hence, as a preprocessing step, the road-network map of the city of Oldenburg was discretized into $10 \times 10 = 100$ regions and all X–Y coordinates of the generated trajectories were replaced with their pertinent regions. In addition, a sensitive value is randomly assigned to every trajectory in a similar setting as in *MetroData*.

To simulate a streaming environment, the window slides over trajectories reading only doublets that fit within the window scope. Algorithm 1 runs only on the data available in a given window. That is, any future or outdated doublets are unknown to the algorithm. Without loss of generality, we set $step\_size = 1$ in all experiments.

We compare our method ITSA with two other methods, namely RFIDAnonymizer [14] and Never Walk Alone (NWA) [1]. RFIDAnonymizer was previously proposed for anonymizing a static RFID data set. To apply this static anonymization method on trajectory stream, we apply RFIDAnonymizer on every window independently. We compare our method with RFIDAnonymizer only in terms of efficiency and scalability, not in terms of data utility, because both methods apply exactly the same sequence of suppression operations, yielding the same result. The contribution of ITSA over RFIDAnonymizer is on efficiency and scalability. We also compare our method with another method called NWA that was proposed by Abul et al. [1] to anonymize a static trajectory data set. We carry out performance analysis with NWA in terms of data utility and

[1] http://dmas.lab.mcgill.ca/fung/pub/MetroDataSet.txt.

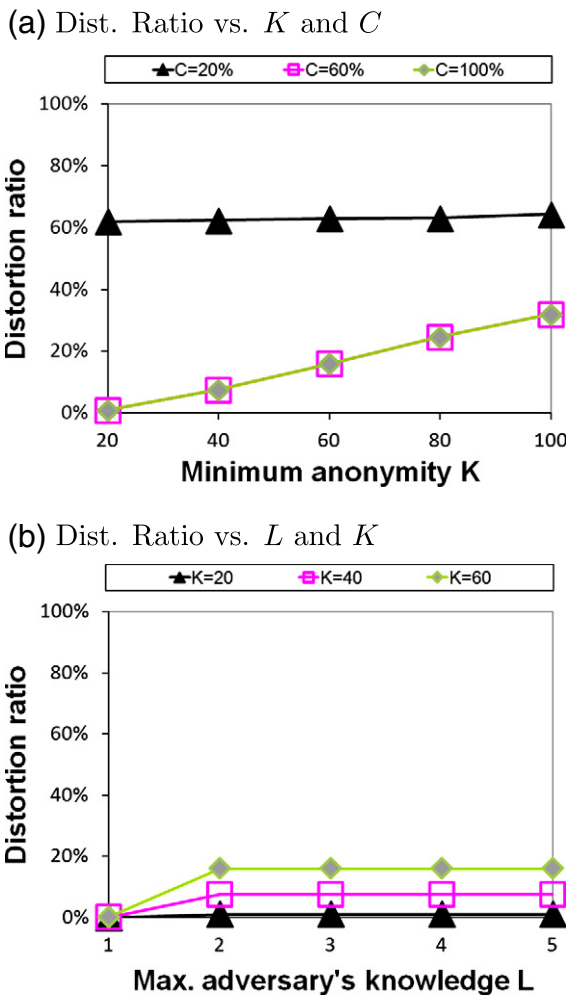

Fig. 4. *MetroData*: the impact of L, K, C ($N = 7$).

efficiency. In addition, we compare the impact of applying different privacy models, namely *LKC*-privacy and *k*-anonymity, on the data utility.

Unless otherwise specified, a value on the distortion ratio and runtime reported in any of the following figures reflects the average value of all the windows sliding over the same data set in one run.

5.1. Metro data set

First, we study the impact of the different parameters on data utility. Then, we generate much larger data sets to test the scalability of our method.

Fig. 4a: We fix N to 7 and the adversary's background knowledge L to 4. We vary the minimum anonymity threshold K from 20 to 100 and the maximum inference confidence C from 20% to 100%. We see that increasing K implies higher distortion ratio. This is because more trajectories must share any subsequence q with $|q| \leq L$, and it is unlikely that a large number of passengers share longer journey routes. Therefore, more suppressions are needed to achieve anonymity. We also notice that the distortion ratio is highest at $C = 20\,\%$. We attribute this odd jump to the random distribution of the sensitive values in each record, 1 out of $5 = 20\,\% = C$. As a result, more doublets are suppressed to satisfy the inference confidence requirement.

Fig. 4b: We fix N to 7 and C to 60%. We vary L from 1 to 5 and K from 20 to 60. For $L \geq 2$, the distortion ratio experiences a steady behavior because the values of K are relatively small compared to the total number of records in the data set. We note that in real-life data sets, the distortion ratio is expected to be proportional to L. This is because in order to satisfy LKC-privacy, more trajectories must share longer subsequences (which is unlikely by itself), thus, requiring further suppressions. Fig. 4b also affirms that distortion ratio increases when K increases.

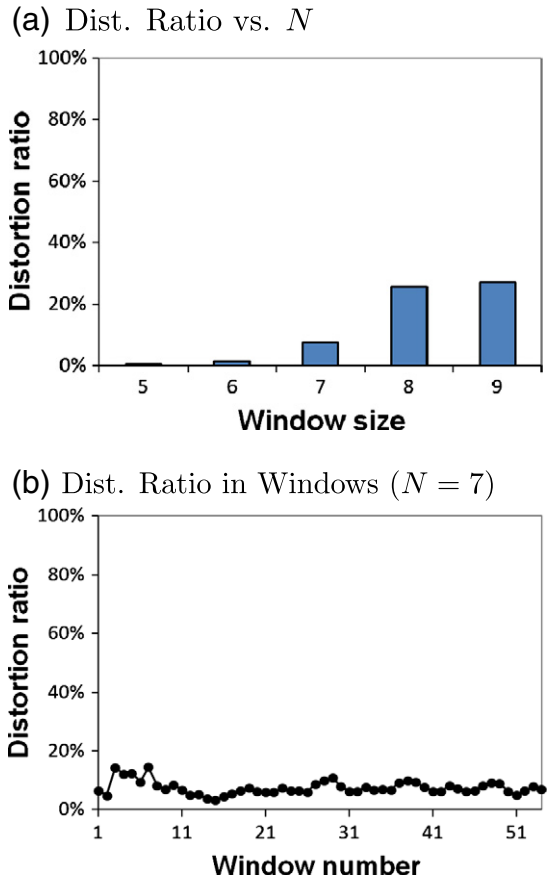

Fig. 5. *MetroData*: sliding window ($L = 4$, $K = 40$, $C = 60$ %).

Fig. 5a: We evaluate the impact of N. We fix L to 4, K to 40, and C to 60%. We vary N from 5 to 9. Fig. 5a insinuates that larger window sizes are likely to cause higher distortions. This is because a larger window size produces a much larger $Cand_i$, resulting in increasing the number of potential critical violations. Generally speaking, as the window slides, new data may or may not introduce new critical violations (Property 4.3), and thus, there is no formal approach for systematically predicting an association between window size and distortion. This reasoning is reflected in Fig. 5a when $5 \leq qN \leq 7$ where the distortion ratio increases slowly. However, the distortion ratio experiences a sudden jump at $N = 8$, but has a marginal increase at $N = 9$.

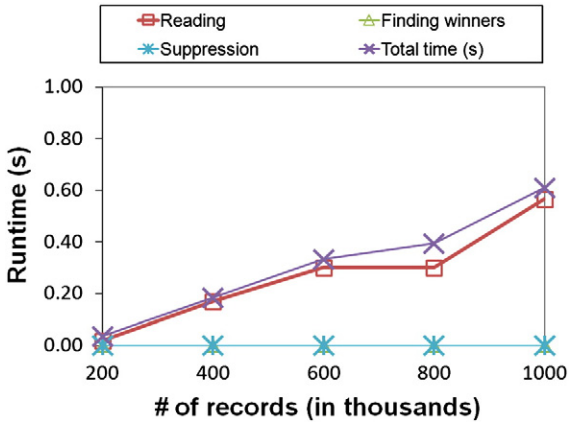

Fig. 6. *MetroData*: scalability ($L = 4$, $K = 120$, $C = 60$ %, $N = 7$).

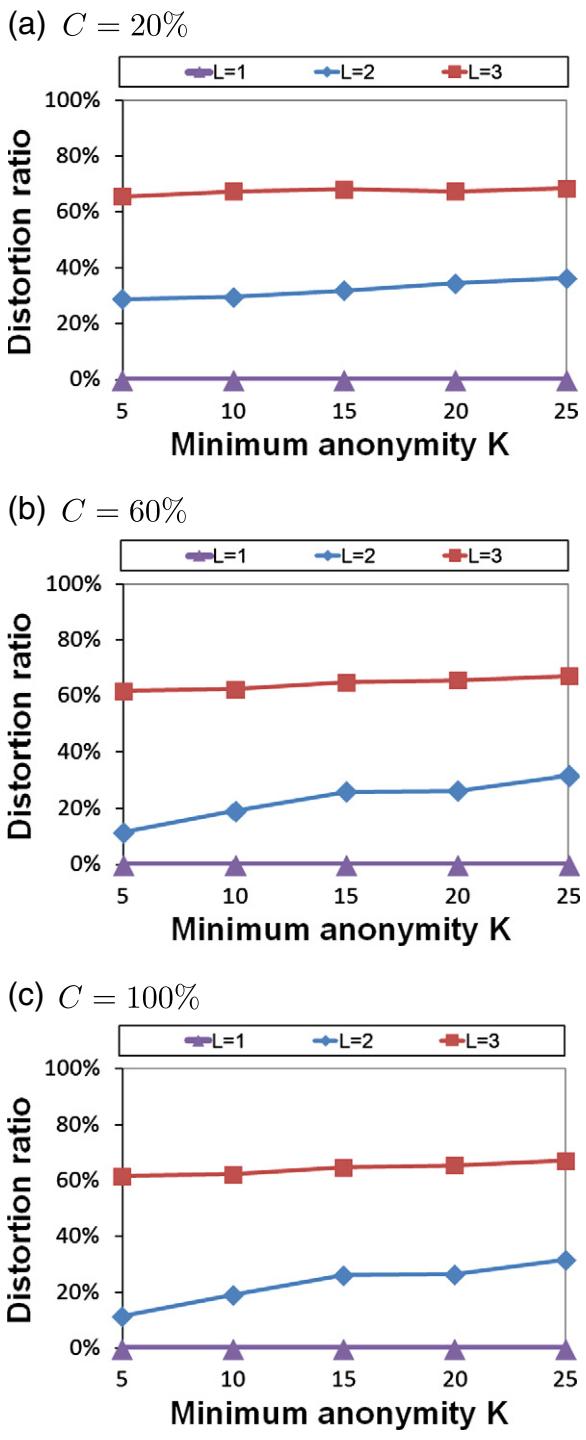

Fig. 7. *MSNBC*: distortion ratio vs. *L*, *K*, *C*.

Fig. 5b: All previous figures were depicting the distortion ratio on average for all windows. Fig. 5b presents the distortion ratio of every window on *MetroData*. We set $L = 4$, $K = 40$, $C = 60\,\%$, and $N = 7$. We see that the distortion ratio exhibits mild fluctuation as the window slides.

Efficiency and scalability: In all the experiments thus far, the runtime of anonymizing a single window before it slides is less than 1 s. We now evaluate the efficiency and scalability of our method in terms of handling data sets with a huge number of records. Our

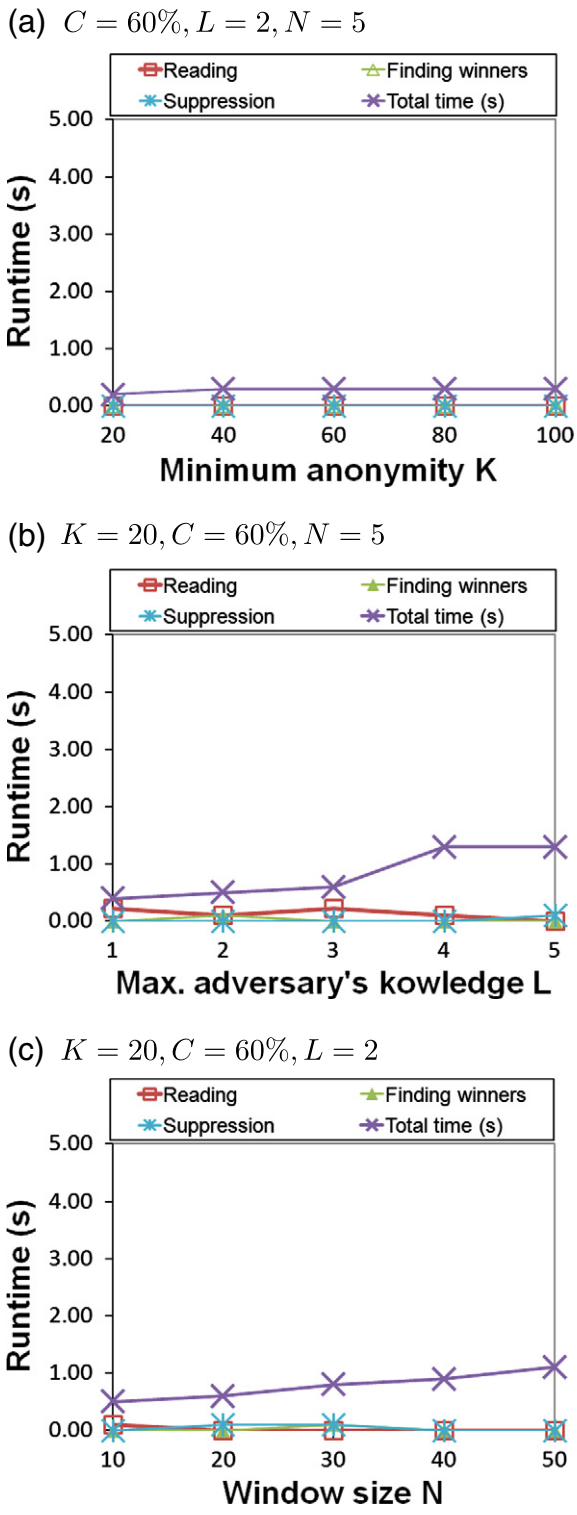

Fig. 8. *MSNBC*: runtime vs. *K*, *L*, *N*.

method puts no restrictions on the amount of data to be processed in a sliding window. The maximum number of doublet instances in any window is equal to window size $N \times$ the number of records [23,38,45]. We use our generator to produce data sets with size ranging from 200 thousand to 1 million records. We set $L = 4$, $K = 120$, $C = 60$ %, and $N = 7$. Fig. 6 shows that a window sliding over 1 million records finishes anonymization in less than 1 s.

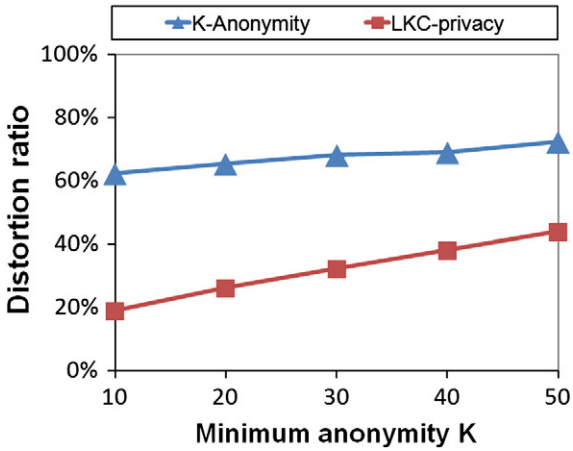

Fig. 9. *k*-Anonymity vs. *LKC*-privacy ($C = 60\,\%, L = 2, N = 5$).

5.2. MSNBC data set

For our second data set, we perform more in-depth analysis and observe the impact of the different parameters on distortion ratio and runtime. All experiments consider the first 10 windows sliding over the entire data set.

Fig. 7: We evaluate the impact of L, K, and C on distortion ratio. Experiments are carried out for window size $N = 5$, anonymity threshold $5 \leq K \leq 25$, maximum adversary's background knowledge $1 \leq L \leq 3$, and confidence threshold $C = 20\,\%, 60\,\%, 100\,\%$. Overall, distortion ratio increases as K increases. At $L = 1$ (sequences of one doublet), the distortion ratio stays below 1% mainly because MSNBC contains a very large number of records, thus, unique doublets are bound to appear more frequently.

Fig. 7 also shows that distortion ratio is proportional to L. Higher values of L imply that longer sequences must exist more frequently. Since it is unlikely that too many users would visit the same sequence of webpages, many doublets have to be suppressed to satisfy the given privacy requirement. Setting $L = 3$ produces a high distortion of $\geq 60\,\%$ in Fig. 7a–c.

Lastly, Fig. 7a ($C = 20\,\%$) reports a higher distortion ratio at $L = 2$ (30% to 40%) than Fig. 7b and c (10% to 30%). This is justified by how sensitive values are assigned to each record. Since the sensitive attribute contains 10 domain values, 2 of which are sensitive, each record has a probability of 20% to be assigned a sensitive value, which is equal to the confidence threshold. Therefore, more suppressions are performed. This observation is reflected in Fig. 7b and c that report lower distortions.

Fig. 8: We study the effect of K, L, and N on the runtime of our method. For every graph, we vary one parameter and fix the others (values are in caption). In Fig. 8a, for $20 \leq K \leq 100$, runtime stays below 0.5 s. Moreover, we note that runtime is insensitive to the change of K. This is because the number of critical violations does not grow significantly as K increases, hence the time to find winner doublets is not generally affected. This observation is depicted in Fig. 7b at $L = 2$ where the distortion ratio increases slowly as K goes higher. The same reasoning can be applied to the inference confidence, C.

Fig. 8b depicts the impact of L on runtime. The maximum value that can be assigned to L is the window size N. Therefore, we vary L between 1 and N. In general, the runtime increases as L increases. For $1 \leq L \leq 5$, $0.4s \leq runtime \leq 1.3s$. As L increases, more sequences

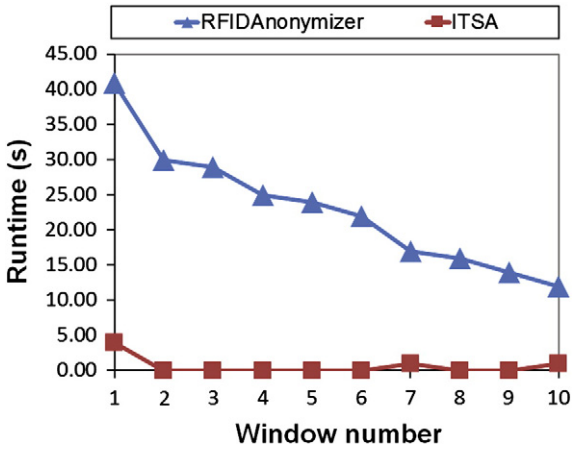

Fig. 10. RFIDAnonymizer vs. ITSA ($K = 20, C = 60\,\%, L = 2, N = 10$).

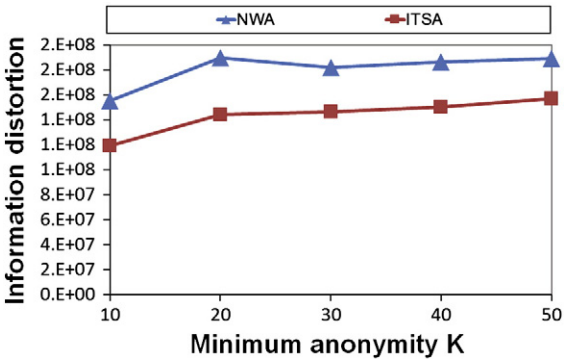

Fig. 11. *Oldenburg*: information distortion.

are generated to be checked for potential critical violations. This fact is also reflected in Fig. 8c where $10 \leq N \leq 50$. Larger values of N imply more unique sequences in any candidate set $Cand_i$, hence more processing time.

Fig. 9: We compare the distortion ratios caused by applying the *LKC*-privacy model and the traditional k-anonymity model, respectively, in our ITSA method. By setting $L = N$ and $C = 100$ %, *LKC*-privacy turns into k-anonymity. For *LKC*-privacy, we set $C = 60$ %, $L = 2$, $N = 5$ and $10 \leq K \leq 50$. Results in Fig. 9 show that k-anonymity maintains a minimum distortion ratio of 60%. Applying *LKC*-privacy significantly lowers the distortion ratio over the different values of K. This can be explained by the fact that k-anonymity requires every record to be shared in its entirety by at least $k - 1$ other records. This strict requirement is addressed in *LKC*-privacy by manipulating the parameter L.

Fig. 10: We measure the efficiency and scalability of our method in terms of runtime (in seconds), and compare the results with those achieved by using RFIDAnonymizer [14]. For this experiment, we measure the runtime of every individual window when ITSA and RFIDAnonymizer are applied, independently. Fig. 10 depicts the runtimes of 10 windows. It is evident that ITSA performs significantly better than RFIDAnonymizer by maintaining an overall runtime of ≤ 1 s, thanks to the properties that we identified in trajectory streams (Section 4.1) and the compact and dynamic tree structure for representing trajectories (Section 4.2). Moreover, when the window slides, ITSA reads only the new data in the stream, while RFIDAnonymizer reads every window in its entirety.

5.3. Oldenburg data set

In this set of experiments, we compare the performance of our method ITSA with another method called NWA that was proposed by Abul et al. [1] to anonymize static trajectory data. We carry out performance analysis of both methods in terms of data utility and efficiency using the *Oldenburg* data set.

NWA applies anonymization through *space translation*, by which trajectory points are either suppressed or dragged in space until every trajectory coexists with at least $k - 1$ other trajectories. A translated point is assigned a penalty equal to the translation distance. If a point is suppressed, it is assigned a penalty equal to the *max point translation* (a constant value corresponding to the maximal translation distance in the entire experiment). This penalty metric is called *Information Distortion*, and is used in our experiments to evaluate the utility of the output anonymized data set. Higher *Information Distortion* implies less data utility.

Our method ITSA applies anonymization through a sequence of suppressions. To compute *Information Distortion*, every suppressed point is assigned a penalty equal to the *max point translation* obtained from applying NWA on the same data set.

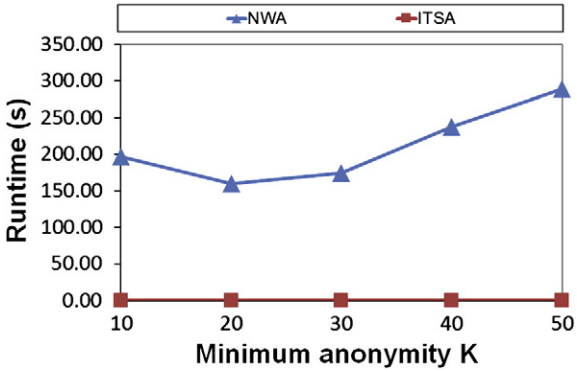

Fig. 12. *Oldenburg*: runtime (s).

Fig. 11: We compare ITSA with NWA in terms of data utility by measuring *Information Distortion*. For ITSA, we set $C = 60\,\%$, $L = 4$, and $N = 5$. As for NWA, we mostly use the default parameters values, specifically, we set $\delta = 200$, $\pi = 5$, $\delta_{max} = 0.01$, and $trash_{max} = 10$. For both methods, we set $10 \leq K \leq 50$. As depicted in Fig. 11, our method constantly achieves 20% less distortion than NWA.

Fig. 12: We compare ITSA with NWA in terms of efficiency by measuring the average runtime (in seconds) of anonymizing a single window. The parameters of both methods are the same as those used in Fig. 11, except that in this experiment we set L to be equal to the window size, $L = 5$. Increasing L requires processing of longer sequences, causing our algorithm to run longer. However, Fig. 12 shows the following two observations. First, ITSA runs significantly faster than NWA, which takes several minutes to anonymize a single window. Second, the runtime of our method is insensitive to the minimum anonymity threshold K (as also observed in Fig. 8a), steadily reporting a runtime of almost 1 s. These two observations suggest that our proposed method is suitable for anonymizing a "live" stream of trajectories.

In summary, the distortion ratio caused by ITSA is dominated by the maximum adversary's background knowledge L and the window size N. This is because both of these parameters incur larger candidate sets containing potential critical violations. This finding validates our analysis of ITSA in Section 4.4. Changing the minimum anonymity threshold K and the inference confidence C does not cause a significant impact on distortion ratio.

6. Conclusion

Due to recent advancement in mobile technology, spatio-temporal data are being continuously generated. The data can be automatically collected by some data holder. In this paper, we propose a novel approach for anonymizing a stream of trajectories generated by moving individuals. The anonymized trajectories are published on the fly to guarantee freshness. We illustrate and formalize two types of privacy threats. We also propose an algorithm for incrementally anonymizing a sequence of dynamically-updated sliding windows on the stream. We structure the window in a way to accommodate massive volumes of data. We evaluate the performance of our method on simulated and real-life data sets, and compare with other methods. Empirical evaluation demonstrates that our method is suitable for anonymizing real-life high-volume trajectory streams, and that it outperforms existing methods.

Acknowledgment

The research is supported in part by the Discovery Grants (356065-2013) from the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

[1] O. Abul, F. Bonchi, M. Nanni, Never walk alone: uncertainty for anonymity in moving objects databases, Proc. of the 24th IEEE International Conference on Data Engineering (ICDE), April 2008, pp. 376–385.
[2] C.C. Aggarwal, On *k*-anonymity and the curse of dimensionality, Proc. of the 31st Very Large Data Bases (VLDB), 2005, pp. 901–909.
[3] R. Agrawal, R. Srikant, Fast algorithms for mining association rules in large databases, Proc. of the 20th Very Large Data Bases (VLDB), 1994, pp. 487–499.
[4] A. Asuncion, D. Newman, UCI Machine Learning Repository, 2007.
[5] B. Babcock, S. Babu, M. Datar, R. Motwani, J. Widom, Models and issues in data stream systems, Proc. of the 21st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS), ACM, New York, NY, USA, 2002, pp. 1–16.
[6] B. Babcock, M. Datar, R. Motwani, Sampling from a moving window over streaming data, Proc. of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '02, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2002, pp. 633–634.
[7] F. Bonchi, L.V. Lakshmanan, H.W. Wang, Trajectory anonymity in publishing personal mobility data, SIGKDD Explor. Newsl. 13 (Aug. 2011) 30–42.
[8] T. Brinkhoff, Generating traffic data, IEEE Data Eng. Bull. 26 (2) (2003) 19–25.
[9] J.-W. Byun, Y. Sohn, E. Bertino, N. Li, Secure anonymization for incremental datasets, Proc. of the VLDB Workshop on Secure Data Management (SDM), 2006.
[10] J. Cao, B. Carminati, E. Ferrari, K. Lee Tan, Castle: a delay-constrained scheme for ks-anonymizing data streams, Proc. of the 24th International Conference on Data Engineering, IEEE Computer Society, Washington, DC, USA, 2008, pp. 1376–1378.
[11] T.-H.H. Chan, E. Shi, D. Song, Privacy-preserving stream aggregation with fault tolerance, Proc. of the International Conference on Financial Cryptography, 2012.
[12] R. Chen, B.C.M. Fung, N. Mohammed, B.C. Desai, Privacy-preserving trajectory data publishing by local suppression, Inf. Sci. Spec. Issue Data Min. Inf. Secur. 231 (May 2013) 83–97.
[13] C. Dwork, M. Naor, T. Pitassi, G.N. Rothblum, S. Yekhanin, Pan-private streaming algorithms, Proc. of The 1st Symposium on Innovations in Computer Science (ICS), 2010.
[14] B.C.M. Fung, K. Al-Hussaeni, M. Cao, Preserving RFID data privacy, Proc. of the 2009 International Conference on RFID, IEEE Communications Society, Orlando, FL, April 2009, pp. 200–207.
[15] B.C.M. Fung, K. Wang, R. Chen, P.S. Yu, Privacy-preserving data publishing: a survey of recent developments, ACM Comput. Surv. 42 (4) (June 2010) 14:1–14:53.
[16] M.M. Gaber, A. Zaslavsky, S. Krishnaswamy, Mining data streams: a review, SIGMOD Rec. 34 (June 2005) 18–26.
[17] G. Gidofalvi, X. Huang, T. Pedersen, Privacy-preserving data mining on moving object trajectories, Proc. of the International Conference on Mobile Data Management, May 2007, pp. 60–68.
[18] L. Golab, M.T. Özsu, Issues in data stream management, SIGMOD Rec. 32 (June 2003) 5–14.
[19] J. Han, J. Pei, Y. Yin, Mining frequent patterns without candidate generation, SIGMOD Rec. 29 (2) (May 2000) 1–12.
[20] Y. He, S. Barman, D. Wang, J.F. Naughton, On the complexity of privacy-preserving complex event processing, Proc. of the 30th ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, PODS '11, ACM, New York, NY, USA, 2011, pp. 165–174.
[21] H. Hu, J. Xu, D.L. Lee, Pam: an efficient and privacy-aware monitoring framework for continuously moving objects, IEEE Trans. Knowl. Data Eng. 22 (3) (march 2010) 404–419.
[22] F. Li, J. Sun, S. Papadimitriou, G. Mihaila, I. Stanoi, Hiding in the crowd: privacy preservation on evolving streams through correlation tracking, Proc. of the 23rd International Conference on Data Engineering, April 2007, pp. 686–695.
[23] J. Li, B.C. Ooi, W. Wang, Anonymizing streaming data for privacy protection, Proc. of the 24th International Conference on Data Engineering, April 2008, pp. 1367–1369.
[24] N. Li, T. Li, S. Venkatasubramanian, t-Closeness: privacy beyond k-anonymity and l-diversity, Proc. of the International Conference on Data Engineering (ICDE), 2007, pp. 106–115.

[25] A. Machanavajjhala, J. Gehrke, D. Kifer, M. Venkitasubramaniam, ℓ-diversity: privacy beyond k-anonymity, Proc. of the 22nd IEEE International Conference on Data Engineering (ICDE), 2006.
[26] A. Meyerson, R. Williams, On the complexity of optimal k-anonymity, Proc. of the 23rd ACM PODS, ACM, Paris, France, 2004, pp. 223–228.
[27] N. Mohammed, B.C.M. Fung, M. Debbabi, Walking in the crowd: anonymizing trajectory data for pattern analysis, Proc. of the 18th ACM Conference on Information and Knowledge Management (CIKM), ACM Press, Hong Kong, November 2009, pp. 1441–1444.
[28] N. Mohammed, B.C.M. Fung, P.C.K. Hung, C. Lee, Anonymizing healthcare data: a case study on the blood transfusion service, Proc. of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD), ACM Press, Paris, France, June 2009, pp. 1285–1294.
[29] A. Monreale, G. Andrienko, N. Andrienko, F. Giannotti, D. Pedreschi, S. Rinzivillo, S. Wrobel, Movement data anonymity through generalization, Trans. Data Priv. 3 (August 2010) 91–121.
[30] C. Negroni, Tracking your wi-fi trail, New York TimesMarch 2011.
[31] M.E. Nergiz, M. Atzori, Y. Saygin, Towards trajectory anonymization: a generalization-based approach, Proc. of the SIGSPATIAL ACM GIS 2008 International Workshop on Security and Privacy in GIS and LBS, SPRINGL '08, ACM, New York, NY, USA, 2008, pp. 52–61.
[32] R.G. Pensa, A. Monreale, F. Pinelli, D. Pedreschi, Pattern-preserving k-anonymization of sequences and its application to mobility data mining, PiLBA, 2008.
[33] L. Sweeney, k-Anonymity: a model for protecting privacy, International Journal on Uncertainty, Fuzziness and Knowledge-based Systems, vol. 10, 2002, pp. 557–570.
[34] M. Terrovitis, N. Mamoulis, Privacy preservation in the publication of trajectories, Proc. of the 9th International Conference on Mobile Data Management (MDM), April 2008, pp. 65–72.
[35] M. Terrovitis, N. Mamoulis, P. Kalnis, Local and global recoding methods for anonymizing set-valued data, VLDB J. 20 (February 2011) 83–106.
[36] K. Wang, B.C.M. Fung, P.S. Yu, Handicapping attacker's confidence: an alternative to k-anonymization, Knowl. Inf. Syst. (KAIS) 11 (3) (April 2007) 345–368.
[37] K. Wang, Y. Xu, R.-W. Wong, A.-C. Fu, Anonymizing temporal data, Proc. of the 10th International Conference on Data Mining (ICDM), December 2010, pp. 1109–1114.
[38] P. Wang, L. Zhao, J. Lu, J. Yang, Sanatomy: privacy preserving publishing of data streams via anatomy, Proc. of the 3rd International Symposium on Information Processing, ISIP '10, IEEE Computer Society, Washington, DC, USA, 2010, pp. 54–57.
[39] R.C.-W. Wong, A.W.-C. Fu, K. Wang, J. Pei, Minimality attack in privacy preserving data publishing, Proc. of the 33rd international conference on Very large data bases, VLDB '07, VLDB Endowment, 2007, pp. 543–554.
[40] X. Xiao, Y. Tao, m-invariance: towards privacy preserving re-publication of dynamic datasets, Proc. of ACM SIGMOD, Beijing, China, June 2007.
[41] X. Xiao, K. Yi, Y. Tao, The hardness and approximation algorithms for l-diversity, Proc. of the 13th International Conference on Extending Database Technology, EDBT '10, ACM, New York, NY, USA, 2010, pp. 135–146.
[42] Y. Xu, B.C.M. Fung, K. Wang, A.W.C. Fu, J. Pei, Publishing sensitive transactions for itemset utility, Proc. of the 8th IEEE International Conference on Data Mining (ICDM), December 2008.
[43] R. Yarovoy, F. Bonchi, L.V.S. Lakshmanan, W.H. Wang, Anonymizing moving objects: how to hide a mob in a crowd? Proc. of the 12th International Conference on Extending Database Technology: Advances in Database Technology, EDBT '09, ACM, New York, NY, USA, 2009, pp. 72–83.
[44] J. Zhang, J. Yang, J. Zhang, Y. Yuan, Kids: k-anonymization data stream base on sliding window, Proc. of the 2nd International Conference on Future Computer and Communication (ICFCC), vol. 2, May 2010, (pages V2-311-V2-316).
[45] B. Zhou, Y. Han, J. Pei, B. Jiang, Y. Tao, Y. Jia, Continuous privacy preserving publishing of data streams, Proc. of the 12th International Conference on Extending Database Technology: Advances in Database Technology, EDBT '09, ACM, New York, NY, USA, 2009, pp. 648–659.

Khalil Al-Hussaeni received his Bachelor's degree in Computer Engineering from the University of Sharjah, U.A.E. He received his Master's degree in Information Systems Security in 2009 from the Concordia Institute for Information Systems Engineering, Concordia University, Canada. He is currently working toward the PhD degree in the Department of Electrical and Computer Engineering at Concordia University, Montreal, Canada. He is a research assistant in the Computer Security Laboratory. His research interests include privacy preservation in big data, data streams, data mining, privacy-preserving data publishing, and computer and network security.

Benjamin Fung is an Associate Professor of Information Studies (SIS) at McGill University and a Research Scientist in the National Cyber-Forensics and Training Alliance Canada (NCFTA Canada). He received a Ph.D. degree in computing science from Simon Fraser University in 2007. Dr. Fung has over 70 refereed publications that span the prestigious research forums of data mining, privacy protection, cyber forensics, services computing, and building engineering. His data mining works in crime investigation and authorship analysis have been reported by media worldwide. Dr. Fung is a licensed professional engineer in software engineering, and is currently affiliated with the Data Mining and Security Lab at SIS.

William K. Cheung is currently the Associate Head and an Associate Professor in the Department of Computer Science, Hong Kong Baptist University. He received the BSc and MPhil degrees in electronic engineering from the Chinese University of Hong Kong and the PhD degree in computer science in 1999 from the Hong Kong University of Science and Technology. He has served as the cochair and program committee members of a number of international conferences/workshops, as well as a guest editor of journals in areas including artificial intelligence, Web intelligence, data mining, Web services, and e-commerce technologies. Also, he is currently the Managing Editor of the IEEE Intelligent Informatics Bulletin. His research interests include artificial intelligence and machine learning, as well as their applications to collaborative filtering, Web mining, distributed and privacy-preserving data mining, planning under uncertainty, network data mining, and health informatics.