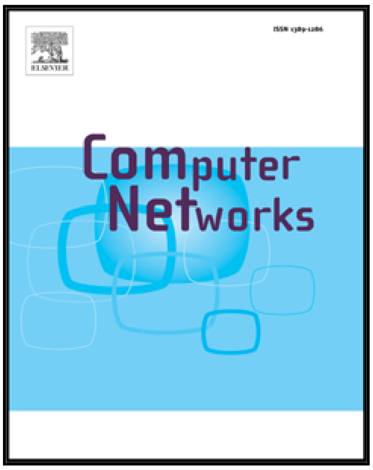


SafePath: Differentially-Private Publishing of Passenger Trajectories in Transportation Systems

Khalil Al-Hussaeni, Benjamin C.M. Fung, Farkhund Iqbal,
Gaby G. Dagher, Eun G. Park

PII:	S1389-1286(18)30488-2
DOI:	10.1016/j.comnet.2018.07.007
Reference:	COMPNW 6537
To appear in:	*Computer Networks*
Received date:	11 December 2017
Revised date:	19 May 2018
Accepted date:	2 July 2018

SafePath: Differentially-Private Publishing of Passenger Trajectories in Transportation Systems

Khalil Al-Hussaeni

CIISE, Concordia University, Montreal, Canada, H3G 1M8

Benjamin C. M. Fung

School of Information Studies, McGill University, Montreal, Canada, H3A 1X1

Farkhund Iqbal

College of Technological Innovation, Zayed University, Abu Dhabi, UAE, P.O. Box 144534

Gaby G. Dagher

Department of Computer Science, Boise State University, Boise, ID 83725-2055, USA

Eun G. Park

School of Information Studies, McGill University, Montreal, Canada, H3A 1X1

Abstract

In recent years, the collection of spatio-temporal data that captures human movements has increased tremendously due to the advancements in hardware and software systems capable of collecting person-specific data. The bulk of the data collected by these systems has numerous applications, or it can simply be used for general data analysis. Therefore, publishing such big data is greatly beneficial for data recipients. However, in its raw form, the collected data contains sensitive information pertaining to the individuals from which it was collected and must be anonymized before publication. In this paper, we study the problem of privacy-preserving passenger trajectories publishing and propose a solution under the rigorous differential privacy model. Unlike sequential data, which describes sequentiality between data items, handling spatio-temporal data is a challenging task due to the fact that introducing a temporal dimension results in extreme sparseness. Our proposed solution introduces an efficient algorithm, called *SafePath*, that models trajectories as a noisy prefix tree and publishes ϵ-differentially-private trajectories while minimizing the impact on data utility. Experimental evaluation on real-life transit data in Montreal suggests that *SafePath* significantly improves efficiency and scalability with respect to large and sparse datasets, while achieving comparable results to existing solutions in

Email addresses: `k_alhus@ciise.concordia.ca` (Khalil Al-Hussaeni), `ben.fung@mcgill.ca` (Benjamin C. M. Fung), `farkhund.iqbal@zu.ac.ae` (Farkhund Iqbal), `gabydagher@boisestate.edu` (Gaby G. Dagher), `eun.park@mcgill.ca` (Eun G. Park)

terms of the utility of the sanitized data.

Keywords: Differential privacy, trajectory data, sparse data, smart city, transportation.

1. Introduction

Location-aware devices and systems capable of collecting user-specific data have been implemented in many smart city infrastructures over the last several years. For example, *Société de transport de Montréal* (STM), the public transport agency responsible for the bus and metro systems in Montreal, has employed a *smart card automated fare collection (SCAFC)* system to collect passenger transit data. The data collected by these systems has numerous uses for data analysis and is crucial to enactment of administrative regulations and generation of metrics to weigh business practices [1]. However, the collected data in its raw form contains sensitive information that is specific to individuals and must be anonymized to protect their privacy before the data is shared by a certain third party for data analysis.

This paper is motivated by the rising concern of passengers' privacy, especially when passenger flow is under constant surveillance [12]. We study the problem of publishing trajectories with spatio-temporal information and propose a privacy-preserving trajectory publishing solution under the differential privacy model [14].

Let us consider the spatio-temporal transit data. For every passenger, the collected data includes the passenger's smart card number, the visited station ID, and a timestamp. Data is collected as passengers present their unique smart cards to an automated reader for fare collection when boarding a bus or entering a metro station. Once collected, the passenger's data is stored in a central database whereby the sequence of visited stations is arranged in a timely order. Table 1 provides an abstract representation of the transit data in its raw form. Passenger-specific information such as names and card numbers are omitted; instead, a trajectory ID has been added for ease of referencing. The sequence of $\langle location, timestamp \rangle$ pairs forms a passenger's path or *trajectory*. For example, trajectory tr_1 in Table 1 contains two pairs, $a.1$ and $c.2$, indicating that the passenger has visited locations a and c at timestamps 1 and 2, respectively.

Trajectories collected by transit companies are periodically shared with internal and external organizations for the purpose of trajectory mining [44][53] and traffic management [52][6][33]. Publishing or sharing raw trajectories raises privacy concerns because the data is susceptible to attacks that rely on attacker's background knowledge about some target victims whose trajectories are included in the published data. We illustrate these attacks in the following example.

Example 1.1. Suppose that Alice is a passenger whose trajectory is in Table 1, which is being released to the public. Possessing auxiliary knowledge about Alice, an attacker can uniquely identify Alice's full trajectory. For example, knowing that Alice was at location b at timestamp 2, an attacker can use this knowledge to associate Alice with trajectory tr_5 with 100% certainty and determine that she has traveled through location a at timestamp 1 and exited the transit system at location c at timestamp 3. Further, suppose that an

Table 1: Raw trajectories of 7 passengers

TID	Trajectory
tr_1	a.1 → c.2
tr_2	c.2 → b.4
tr_3	a.2 → b.3 → c.4
tr_4	c.3 → a.4
tr_5	a.1 → b.2 → c.3
tr_6	a.3 → c.4
tr_7	a.3 → b.4

attacker knows that Bob was at location b at timestamp 4. Hence, Bob's trajectory is either tr_2 or tr_7. In other words, the attacker is able to deduce that Bob was at $c.3$ with 50% confidence or at $a.3$ with 50% confidence, as well. The former attack is called *record linkage* and the latter is called *attribute linkage*. ■

Previously, publishing trajectories has been explored through syntactic and semantic privacy models. Syntactic models such as k-anonymity [43] and ℓ-diversity [34] have been combined with *spatial generalization*, *space translation*, and *suppression-based* techniques to anonymize trajectories. However, while retaining data utility, these models are prone to a number of privacy attacks [47][22][31].

Differential privacy [14] is a semantic privacy model that provides provable privacy guarantees. In this paper, we utilize differential privacy to anonymize a set of trajectories and output a sanitized version with an effective level of utility. There have been several semantic models [27][28][9][26][19] developed by researchers for achieving privacy-preserving trajectory publishing. Unlike existing approaches, however, our approach preserves the time dimension in the sanitized trajectories, allowing for more precise analysis, as opposed to solely preserving the *sequentiality* of locations.

Differential privacy works in two environments: the *interactive setting* [17] and the *non-interactive setting* [5][49]. The former setting allows the data holder (e.g., the transit company) to receive a limited number of requests pertaining to the underlying raw data, and sanitized answers are released. In the latter setting the data holder sanitizes then releases the entire data. In both settings, the differentially-private mechanism consumes at most ϵ budget, where ϵ is a pre-defined *privacy budget*. We propose a solution that works under the non-interactive setting as it gives extra flexibility in terms of analysis power, especially when there is no specific data recipient for the released data. Subsequently, the target data analysis task is unknown and can vary considerably depending on the desired analysis task. Moreover, many specific data mining techniques rely on count queries to accomplish their objective. Hence, we focus on preserving *count queries* in the sanitized data to provide various data recipients with the freedom to perform a wide variety of analysis tasks. It is worth noting that in our problem definition we assume the presence of the entire trajectory of a moving individual before sanitizing the complete set of trajectories. Section 2 presents a detailed discussion of related problems and existing solutions.

Contributions. In this paper, we investigate the problem of privacy-preserving trajectory publishing. Unlike sequential data, trajectories are spatio-temporal data that contains a time dimension. The existence of a temporal dimension renders the problem challenging because extra care is required in the sanitization process to handle the sparseness of data. We first formalize the problem of publishing trajectories under the rigorous privacy model of differential privacy in the non-interactive setting. We then propose an efficient and scalable sanitization algorithm, called *SafePath*, that models trajectories in a prefix tree structure, which significantly contributes to the ability of handling sparse and high-dimensional data. The closest work to ours is by Chen et al. [9]. They proposed to sanitize *sequential data* (e.g., sequences of locations) under differential privacy and further suggested that their method could be extended to spatio-temporal data, i.e., sequences of locations *coupled* with timestamps. We implemented Chen et al.'s extension and call it *SeqPT*. Through theoretical analysis and experiment on real-life trajectory datasets obtained in Montreal, we demonstrate that *SafePath* provides several fold improvement over *SeqPT* with respect to efficiency in terms of runtime and scalability in terms of both number of records and dimensionality of the dataset. Moreover, we experimentally compare with another sequential data sanitization method, called*N-gram* [8], and showcase the superiority of our proposed method in terms of runtime.

The rest of the paper is organized as follows. Section 2 discusses related work in the literature. Section 3 introduces some preliminary concepts and formalizes the problem. Section 4 presents the details of our method and ends with a theoretical analysis of the proposed solution. Section 5 provides extensive experimental evaluations on real-life trajectory datasets. Finally, Section 6 concludes the paper.

2. Related Work

Due to the significance of mining trajectories or moving object databases (MOD), there has been extensive work on privacy-preserving trajectory publishing under different assumptions and privacy models. Next, we categorize those works based on their privacy models, namely, *syntactic* and *semantic*.

Syntactic privacy models, such as k-anonymity [43] and ℓ-diversity [34], stipulate that the output dataset of an anonymization algorithm must adhere to some syntactic conditions in order to protect data records and sensitive items. Nergiz et al. [38] were the first to apply k-anonymity to a trajectory dataset, whereby every trajectory in its *entirety* must be indistinguishable from at least $k-1$ other trajectories. Abul et al. [2] proposed (k, δ)-*anonymity* that enforces *space translation*, resulting in having every trajectory coexisting with a minimum of $k-1$ other trajectories within a proximity of δ. Monreale et al. [37] achieved k-anonymity by using *spatial generalization*. The novelty of their method lies in dynamically generating geographical areas based on the input dataset, as opposed to generating a fixed grid [51]. Hu et al. [29] applied k-anonymity to a trajectory dataset with respect to a reference dataset containing sensitive events. Particularly, they developed *local enlargement* that transforms the trajectory dataset such that every sensitive event is shared by at least k users.

In addition to generalization [38][37][51][42] and space translation [2][39], suppression-based techniques [45][10][11] have been proposed to achieve k-anonymity-based privacy models. Terrovitis and Mamoulis [45] developed a privacy model that assumes different adversaries possess different background knowledge, and consequently they modeled such knowledge as a set of projections over a sequential (trajectory) dataset. Their anonymization method limits the inference confidence of locations to a pre-defined threshold. Similarly, Cicek et al. [11] ensured location diversity by proposing p-confidentiality, which limits the probability of visiting a sensitive location to p. *Local suppression* has been used in [10][23] to boost data utility. Under local suppression, only some instances of an item will be removed from the dataset - as opposed to *global suppression*, which removes all instances in the underlying dataset. The authors in [23] proposed to preserve flow analysis in published trajectories under the LK-anonymity model. We argue that it is possible to achieve comparable analysis results without employing syntactic privacy models, which have been proven to be prone to privacy attacks such as *minimality attack* [47], *composition attack* [22], and *deFinetti attack* [31]. Moreover, our proposed algorithm preserves count queries, which is the basis for many data analysis tasks, including flow analysis. Section 5 conducts experimental comparisons when anonymizing trajectories under two privacy models, i.e., LK-anonymity and *differential privacy* [15], in terms of data utility and efficiency.

In the past few years, the body of research in the area of privacy-preserving data publishing has shifted towards adopting *differential privacy* [15], a semantic privacy model that provides provable privacy guarantees. The literature has defined two settings under which differential privacy can be achieved: *interactive* and *non-interactive*. For more information on the interactive setting, non-interactive setting, and recent works on differentially-private data publishing, we refer the reader to [16][32][46], respectively. In the following, we review recent works in the non-interactive setting that are relevant to ours.

Protecting trajectories under differential privacy has been gaining increasing attention in the past few years. Some of these works focus on publishing data mining results, e.g., mining trajectories for frequent location patterns [27][28], whereas other works aim at publishing differentially-private trajectories. We focus on the latter approach as it provides more analytical power to data recipients.

Chen et al. [9] introduced the first differentially-private work to publish a large-volume of *sequential locations*. Although their sanitization algorithm preserves count queries and frequent sequential pattern mining [3] only, data recipients can perform several other data mining tasks on the sanitized output dataset. The work in [8] also targets sanitizing sequential locations, however, by proposing a $variable-lengthn-grammodel$ and constructing a synthetic dataset based on the Markov assumption. In a more recent study, He et al. [26] proposed to synthesize trajectories from a probabilistic model based on the *hierarchical reference system* of the input dataset. Xiao and Xiong. [50] considered temporal correlation to protect true locations *within a single trajectory*, as opposed to *user-level* privacy (adopted in our work), which protects the presence of an entire trajectory *within a dataset*. This is achieved by hiding the true location within a set of probable locations, called δ*-location set*. Works similar to [9][9][26][50] define trajectories as *sequential locations*. We argue that in real-life trajectories every location is paired with a timestamp that should also be accounted for by the trajectory publishing mechanism. For example, it is important to know busy streets when performing traffic analysis, but it is equally important to also know the time period during which traffic jams peak.

Trajectories have also been modeled as *time-series* [19][20] [41][7], where the objective of the differentially-private mechanism is to publish summary statistics at every increasing timestamp in a continuous fashion. Fan et al. [19] were among the first to study the problem of publishing time-series data for traffic monitoring. Their proposed method partitions a given geographical area into a grid of equally-sized cells; then, for every timestamp, a Laplace noise is added to every cell's true count in the 2D space. Finally, the perturbed 2D snapshot is published. Two estimation techniques, temporal and spacial, were proposed to mitigate the effect of the added noise. To achieve ϵ-differential privacy, the privacy budget ϵ is distributed equally among all timestamps. To address the problem of data sparseness, Qardaji et al. [41] studied enhancing partitioning spacial domains. They first proposed a method that uniformly partitions a spacial grid into equally-sized cells where the size of the cell is calibrated based on the input privacy budget and the characteristics of the input dataset. Furthermore, they proposed an adaptive-grid method that imposes finer or coarser partitioning based on cell density. Recently, Cao and Yoshikawa [7] studied publishing sanitized statistics of ℓ-length trajectories in a stream of time-series data. Their proposed framework comprises three steps: dynamic budget allocation among timestamps, private decision making for approximately re-publishing "close" noisy data, and private data release.

Unlike our problem, which assumes the availability of the full trajectory, time-series solutions focus on periodically publishing snapshots of summary statistics because the full trace of a moving individual is not yet available (e.g., location-based services [18]). Furthermore, published summaries statistics under time-series solutions present a sanitized overview of the entire data as a whole. Our proposed solution, on the other hand, is fine-tuned towards preserving every trajectory, i.e., the path of every moving individual in the sanitized data is traceable on the location-timestamp level. For example, there are two passengers at $c.4$ in Table 1; our trajectory publishing method allows us to know which passenger came from $b.3$ and which came from $a.3$. This is considerably advantageous for data analysts.

Jiang et al. [30] sampled distance and angle between true locations within a trajectory in order to publish an ϵ-differentially private version of that trajectory. However, their method publishes a single trajectory only, i.e., the entire privacy budget ϵ is spent on sanitizing a single trajectory. Primault et al. [40] proposed to hide moving individuals' points of interest [21], such as home or work. While their method protects against inference attacks, we argue that hiding points of interest is harmful for applications that rely on such information, e.g., traffic analysis and probabilistic flowgraph analysis. Our sanitization approach, however, aims at maintaining the spatio-temporal characteristics of the raw trajectories in order to support a wide range of data analysis tasks. Assam et al. [4] presented a method whereby both spacial and temporal domains are sanitized and published under differential privacy. In [4], trajectories are represented by a series of GPS-like data points (x, y, t). Their method first creates temporal blocks (called *Running*

Windows) that average all the data points that fall in them. Every *Running Window* is then represented by its average location and timestamp values, which are further perturbed under the Laplace mechanism. The sequence of noisy averages constitutes the sanitized trajectory. Assam et al.'s method is robust enough to output a *single* trajectory with fairly good utility. However, it is unclear how their method can handle multiple moving individuals since the output of their proposed algorithm is always a single sequence of noisy averaged data points. In contrast, our proposed algorithm outputs a *multiset* of trajectories, each belongs to a unique moving individual.

3. Preliminaries

In this section, we introduce *differential privacy*, followed by an introduction to *prefix tree*, and, lastly, we present the problem statement.

3.1. Differential Privacy

Differential privacy [14] is a probabilistic privacy model that bounds the probability of obtaining the same answer from two different input datasets, D and D', that differ by *only* one record. Any privacy leak on the differentially-private dataset, symbolized as $\hat{D}$, will not be conclusive, as $\hat{D}$ could have been obtained from sanitizing either D or D'. This gives incentive for individuals to participate in the dataset because a differentially-private mechanism is impartial to the input raw dataset. Below is a formal definition:

Definition 3.1 (*ϵ-Differential Privacy*)**.** *A randomized algorithm Ag gives ϵ-differential privacy if for any neighboring datasets D and D' differing by at most one record, and for any possible output dataset $\hat{D}$,*

$$Pr[Ag(D) = \hat{D}] \leq e^{\epsilon} \times Pr[Ag(D') = \hat{D}], \tag{1}$$

where the probability is taken over the randomness of the Ag. ■

ϵ is a privacy parameter, called the *privacy budget*, that calibrates the utility of the sanitized data. Typically ranging $0 < \epsilon \leq 1$, lower values of ϵ cause more noise to be added to the true answer, and vice versa.

Suppose there exists a function f that maps a dataset D to real values. The *sensitivity* of f is the maximum change in the true answer due to adding or removing a single record in D. For example, suppose f answers to count queries over D. The maximum change of a true query answer due to adding/removing one record in D is 1. Therefore, the sensitivity of f in this case is 1. The sensitivity of f, symbolized as Δf, is defined as follows:

Definition 3.2 (*Sensitivity*)**.** *For any function $f : D \to \mathbb{R}^d$, the sensitivity of f is*

$$\Delta f = max_{D,D'} ||f(D) - f(D')||_1 \tag{2}$$

for all D, D' differing by one and only one record. ■

The literature has defined two techniques to aid in realizing differential privacy: the *Laplace mechanism* [17] and the *exponential mechanism* [36].

The **Laplace mechanism** first computes the true answer of a function f over a dataset D, $f(D)$, and then adds to $f(D)$ a noise drawn from the Laplace distribution. More formally, the Laplacian noisy answer given by the Laplace mechanism is $f(\hat{D}) = f(D) + \texttt{Lap}(\lambda)$, where $\texttt{Lap}(\lambda)$ is a noise drawn from the Laplace distribution with probability density function $\Pr(x|\lambda) = \frac{1}{2\lambda}\texttt{exp}(-|x|/\lambda)$ of variance $2\lambda^2$ and mean 0.

Theorem 3.1. *[17] Given any function $f : D \to \mathbb{R}^d$ over an arbitrary domain of database D with d attributes, an algorithm Ag that adds independently generated noise with distribution $\texttt{Lap}(\Delta f/\epsilon)$ to each of the d outputs satisfies ϵ-differential privacy.* ■

For example, suppose f answers to count queries over D. Given a privacy budget ϵ and the sensitivity of f, Δf, then according to Theorem 3.1, $f(\hat{D}) = f(D) + \texttt{Lap}(1/\epsilon)$ satisfies ϵ-differential privacy.

As for the **exponential mechanism**, it is used in situations where the true answer is not a real value. In this case, the exponential mechanism assigns a probability to every candidate output o in the output domain $\mathcal{O}$. The assigned probability is based on a utility function u that gives real-valued scores to every candidate output $o \in \mathcal{O}$. Outputs with higher scores are exponentially more likely to be selected by the exponential mechanism. This ensures that the selected output is close to the true output.

Theorem 3.2. *[36] Given any utility function $u : (D \times \mathcal{P}) \to \mathbb{R}$ with sensitivity $\Delta u = max_{\forall p,D,D'} \; |u(D,p) - u(D',p)|$, an algorithm Ag that chooses an output p with probability proportional to $\texttt{exp}(\frac{\epsilon u(D,p)}{2\Delta u})$ satisfies ϵ-differential privacy.* ■

Differential privacy enjoys two **composition** properties: *sequential composition*, and *parallel composition*. Sequential composition stipulates that if a sequence of differentially-private computations takes place on *the same* set of data, then the entire sequence guarantees the collective privacy guarantee of every computation in the sequence. Whereas, parallel composition applies to situations where a sequence of differentially-private computations is performed on *disjoint* sets of data. In this case, the entire sequence gives the worst privacy guarantee, i.e., the highest privacy budget among the parallel computations.

Lemma 3.1 (*Sequential Composition* [35])**.** Let each computation Ag_i provide ϵ_i-differential privacy. A sequence of $Ag_i(D)$ over the dataset D provides $(\sum_i \epsilon_i)$-differential privacy. ■

Lemma 3.2 (*Parallel Composition* [35])**.** Let each computation Ag_i provide ϵ-differential privacy. A sequence of $Ag_i(D_i)$ over a set of disjoint datasets D_i provides ϵ-differential privacy. ■

3.2. Trajectories as Prefix Tree

In this paper we assume that individuals are traveling from one location to another on a geographical map. The map is discretized into unique spacial areas that collectively form a location universe. A single trajectory is the trace left by a single individual, where every visited location is coupled with a timestamp. Timestamps within a single trajectory are non-decreasing and are drawn from a timestamp universe, whereas a location may appear multiple times and/or consecutively. Formally speaking,

Definition 3.3 (*Trajectory*)**.** *A trajectory*

$$tr = L_1.T_1 \rightarrow L_2.T_2 \rightarrow \ldots L_i.T_j \tag{3}$$

is a finite sequence of pairs consisting of a location $L_i \in \mathcal{L}$ and a monotonically-increasing timestamp $T_j \in \mathcal{T}$, where $1 \leq i \leq |\mathcal{L}|$, $1 \leq j \leq |\mathcal{T}|$, and $T_j \leq T_{j+1}$. ■

Definition 3.3 describes timestamps in a trajectory as monotonically increasing. That is, given a passenger's trajectory, two consecutive location-timestamp pairs may have the same timestamp, i.e., $T_j = T_{j+1}$. We define a trajectory in such a way because of two reasons. First, the sequentiality of timestamps depends on the granularity at which data were collected. For example, a transportation agency may choose to collect passengers data every hour, whereas another agency may choose to collect such data every 10 minutes. Second, a sequence of monotonically-increasing timestamps ($T_j = T_{j+1}$) is a general case of strictly-increasing ($T_j < T_{j+1}$) timestamps. Our proposed method can be applied to both types of increasing timestamps. Without loss of generality, we present the examples throughout the paper following the strictly-increasing assumption purely to deliver a clear understanding to the reader.

We use $|tr|$ to denote the trajectory *length*, which is the number of location and timestamp pairs in tr. For example, the first trajectory tr_1 in Table 1 is of length $|tr_1| = 2$.

As opposed to sequential data [9], trajectory data contain a time dimension that renders the data high-dimensional and, in most cases, extremely sparse. Handling sparse data is a challenging problem because processing time is an important aspect of sanitization and should be performed in a timely manner. For this reason, we structure a set of trajectories D as a *prefix tree*, which provides the desired compactness for achieving efficient processing. A prefix tree creates a node for every unique pair of location and timestamp, such that the node is a prefix to all its descendants. Moreover, a $root-to-leaf$ path in a prefix tree represents one unique trajectory $tr_n \in D$. Every node along a $root-to-leaf$ path contains all the trajectories in D to which the $root-to-node$ path is considered a *prefix*. A trajectory $tr_m = L_1^m.T_1^m \rightarrow L_2^m.T_2^m \rightarrow \ldots L_i^m.T_j^m$ is a prefix to another trajectory $tr_n = L_1^n.T_1^n \rightarrow L_2^n.T_2^n \rightarrow \ldots L_i^n.T_j^n$, denoted by $tr_m \preceq tr_n$, if and only if: (1) $|tr_m| \leq |tr_n|$, and (2) $\forall L_i^m.T_j^m \in tr_m$, $L_i^m = L_i^n$ and $T_j^m = T_j^n$, where $1 \leq i, j \leq |tr_m|$. For instance, let $tr_1 = L_1.T_1 \rightarrow L_2.T_2 \rightarrow L_3.T_3$, $tr_2 = L_1.T_1 \rightarrow L_2.T_2$, $tr_3 = L_1.T_1 \rightarrow L_3.T_3$, and $tr_4 = L_1.T_1 \rightarrow L_1.T_2$. $tr_2 \preceq tr_1$, but $tr_3, tr_4 \npreceq tr_1$.

Definition 3.4 (*Prefix Tree*)**.** *A prefix tree $R = (Nodes,\ Edges, Root)$ of a trajectory dataset D is a collection of $Nodes$ connected by $Edges$ and rooted at the $Root$ node. $\forall v \in Nodes$, $v(L.T) = \langle tr(v), c(v)\rangle$, where $L.T \in tr \in D$ is a pair of location and timestamp representing v, $tr(v) = \{tr \in D \mid prefix(v) \preceq tr\}$ where $prefix(v)$ is a unique prefix preresenting the $Root - to - v$ path, and $c(v) = |tr(v)| + \texttt{Lap}(\lambda)$ is a noisy count. The $Root$ node is represented by the pair 0.0 and $tr(Root) = D$.* ■

From Definition 3.4, given a node v in a prefix tree R, $tr(v) \subseteq D$ is the set of trajectories containing the prefix defined by the $Root-to-v$ path, symbolized as $prefix(v)$. Going from a parent node $v_p(L_i.T_j)$ at level ℓ in R to a child node $v_c(L_{i+1}.T_{j+1})$ at level $\ell+1$ implies the transition $L_i.T_j \rightarrow L_{i+1}.T_{j+1}$. We refer to the set of nodes at level ℓ by $level(\ell)$. The *Root* node belongs to $level(0)$, and the set of *Root*'s child nodes belongs to $level(1)$, etc.

3.3. Problem Statement

A trajectory dataset $D = \{tr_1, tr_2, \ldots, tr_{|D|}\}$ is a multiset of trajectories where every record in D is a trajectory that belongs to a single and unique record owner. $|D|$ denotes the size of the dataset, i.e., the number of individuals in D. A data holder has access to D and wishes to publish a differentially-private version of D, denoted by $\hat{D}$, that can be used for various data analysis tasks.

Moreover, we assume the existence of two *taxonomy trees*, one for locations and one for timestamps. A taxonomy tree defines a generalization hierarchy across the entire domain values. For example, given the set of all metro stations $\mathcal{L}$ in a trajectory dataset provided by a transportation agency, a location taxonomy tree A_{loc} defines non-overlapping subsets of metro stations as the child nodes of their respective metro lines. The same generalization concept extends to timestamp domain values $\mathcal{T}$. We note that such taxonomy trees are either publicly available (e.g., a map of metro stations and bus routes) or are practically straightforward to create.

In Section 4 we propose a differentially-private algorithm under the objective of maintaining *count queries* over the sanitized trajectories $\hat{D}$ with respect to the raw trajectories D. Many data mining techniques rely on count queries to accomplish data analysis. Stemming from the goal of providing data recipients with the freedom to perform a wide variety of data analysis tasks on the sanitized trajectories, and given that the target data analysis task is unknown at the sanitization stage, we aim at sanitizing a set of raw trajectories without compromising its utility in terms of count queries.

A count query over a trajectory dataset returns the number of trajectories of which the issued query is a *subtrajectory*. Query q is a subtrajectory of trajectory tr, denoted by $q \subseteq tr$, if and only if: (1) $|q| \leq |tr|$, and (2) $\forall L_i.T_j \in q$, $L_i.T_j$ is also $\in tr$, where $|q|$ denotes the length of the query, i.e., the number of location and timestamp pairs in q. A count query $q \subseteq tr$ contains a subset of pairs from tr while maintaining sequentiality. For example, suppose query $q = L_1.T_1 \rightarrow L_5.T_{10}$ is issued over the dataset $D = \{L_1.T_1 \rightarrow L_5.T_{10}, L_1.T_1 \rightarrow L_2.T_5 \rightarrow L_5.T_{10}\}$. Then, the returned answer $q(D) = 2$.

Definition 3.5 (*Count Query*)**.** *Let q be a count query that contains a sequence of location and timestamp pairs in accordance with Definition 3.3. A count query q issued over a trajectory dataset D is defined as* $q(D) = |\{tr \in D \mid q \subseteq tr\}|$. ■

The utility of a count query q over a sanitized data $\hat{D}$ is computed by its *relative error* [48][49][9], which measures how far the noisy answer $q(\hat{D})$ is from the true answer $q(D)$. That is, $relative\_error(q(\hat{D})) = \frac{|q(\hat{D}) - q(D)|}{q(D)}$. However, when $q(D)$ returns a very small value, the computed error becomes very large. In order to limit the impact of

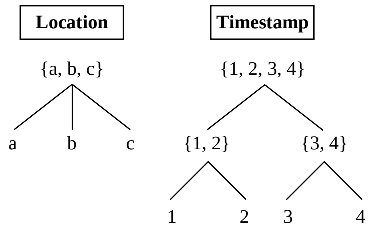


Figure 1: Taxonomy trees

extremely small count queries, it is common to use a *sanity bound* [48][49]. This prevents the relative error from being excessively dominated by extremely small fractions of the data [24]. Therefore, the relative error is computed as follows:

$$relative\_error(q(\hat{D})) = \frac{|q(\hat{D}) - q(D)|}{max\{q(D), s\}},$$

where s is a sanity bound. In Section 5, we choose $s = 0.1\%$ of the raw dataset as in [9] and [49].

The following is a definition of the problem that we tackle in this work:

Definition 3.6 (*Privacy-Aware Trajectory Publishing*)**.** *Given a raw trajectory dataset D, a privacy budget ϵ, and a set of taxonomy trees, we wish to publicly publish $\hat{D}$, a differentially-private version of D, such that: (1) $\hat{D}$ minimizes the distortion inflicted on count queries due to sanitization, and (2) the sanitization process is efficient and scalable to handle extremely sparse trajectories.* ■

4. Proposed Algorithm

In this section we present *SafePath*, our proposed algorithm for publishing differentially-private trajectories. Section 4.1 gives an overview of the entire algorithm. Sections 4.2 and 4.3 describe the algorithm in detail. We end with a theoretical analysis of the proposed solution in Section 4.4.

4.1. Overview

Summary. We propose a sanitization algorithm primarily comprised of two phases: (1) building the noisy prefix tree, and (2) constructing the sanitized dataset. Given a raw trajectory dataset D, a privacy budget ϵ, location and timestamp taxonomy trees A_{loc} and A_{time}, respectively, and the height, h, of the noisy prefix tree, our proposed sanitization algorithm (presented in Algorithm 1) returns a differentially-private trajectory dataset $\hat{D}$. Algorithm 1 first builds the noisy prefix tree R, and then feeds R to *EnhanceConsistency* that constructs $\hat{D}$ by systematically traversing R to ensure data consistency along trajectory paths. Lastly, Algorithm 1 releases the sanitized trajectory dataset $\hat{D}$.

Taxonomy Trees. A taxonomy tree consists of a generalization hierarchy in which every node contains a unique range interval that is a subset of the domain values. The range intervals of all the nodes that belong to the *same* generalization level collectively constitute the entire domain. Given a taxonomy tree, we call a leaf node a *non-general node* and any other node a *general node*. A taxonomy tree is defined by two parameters on which we assume no restrictions: the *taxonomy tree height* and the number of *children*. The former parameter defines the height of the *generalization hierarchy*, i.e., the number of levels containing the general nodes. The latter taxonomy tree parameter, number of children, defines the maximum number of child nodes that belong to the same parent node in the taxonomy tree. An example of user-defined taxonomy trees is given in Figure 1. The height of the timestamp taxonomy tree is 1 because there exists only one level that contains general nodes, and the number of children is 2; similarly, the height of the location taxonomy tree is 0 because the tree contains no levels with general nodes. In Section 5, we experimentally evaluate the impact of different taxonomy trees on the performance of our method by varying the taxonomy tree height and the number of children.

Privacy Budget Allocation. In order to satisfy ϵ-differential privacy, Algorithm 1 effectively distributes the input parameter ϵ among its differentially-private operations. ϵ is *uniformly* distributed among each level in the noisy prefix tree, i.e., every level ℓ is allocated $\epsilon_\ell = \frac{\epsilon}{h}$. Upon construction, a level consists of two sublevels: location and timestamp. Each sublevel is assigned an equal portion of the level's privacy budget, i.e., $\epsilon_s = \frac{\epsilon_\ell}{2}$. Furthermore, each sublevel consists of a hierarchy that generalizes location/timestamp domain values to multiple generalization levels. Here, we employ *non-uniform* budget distribution, as follows. A general node receives a portion of ϵ_s proportional to the generalization hierarchy level to which the general node belongs, and a non-general node receives the remaining portion of ϵ_s. More specifically, a general node (i.e., a non-leaf node) at generalization level d in taxonomy tree A is allocated a privacy budget $\epsilon_g = d \times \epsilon_u$, where $\epsilon_u = \frac{2\epsilon_s}{|\mathcal{U}|}$ is a *unit budget* defined as a function of the underlying universe size $|\mathcal{U}|$, where $\mathcal{U} =$ the location universe $\mathcal{L}$ for a location subselvel and $\mathcal{U} =$ the timestamp universe $\mathcal{T}$ for a timestamp sublevel. A non-general node (i.e., a leaf node) receives a privacy budget $\epsilon_{ng} = \frac{(|\mathcal{U}|-2\sum_{x=1}^{d} x)\epsilon_s}{|\mathcal{U}|}$. We choose such an allocation scheme because less general nodes (at the lower levels of a given hierarchy) contain smaller trajectory counts, thus it is fair to increase the allocated budget portion as the nodes go deeper in the hierarchy. In Section 5, we examine the effect of ϵ on data utility.

Budget distribution among tree levels does not follow a scientific procedure. We justify our uniform distribution strategy as per the following two reasons. First, we follow the tradition in the literature, such as the work in [9], by equally distributing the budget among tree levels. More importantly, the second reason for choosing a uniform budget distribution strategy is as follows. A noisy prefix tree level i represents a location and timestamp pair $L_i.T_i$ along trajectory tr. Level $i+1$ represents pair $L_{i+1}.T_{i+1}$ in the same tr. Following a uniform allocation strategy, both pairs $L_i.T_i$ and $L_{i+1}.T_{i+1}$ in tr have an equal chance of receiving the same noise. Whereas, if level i has, for example, a higher budget than level $i+1$, then $L_i.T_i$ has a higher chance of receiving less noise, thus being more accurate, than $L_{i+1}.T_{i+1}$. Assuming that all trajectory pairs are of the same significance, we consider uniform distribution to be a "fair" allocation strategy.

We note that only non-general nodes will be added to the prefix tree. The sublevels

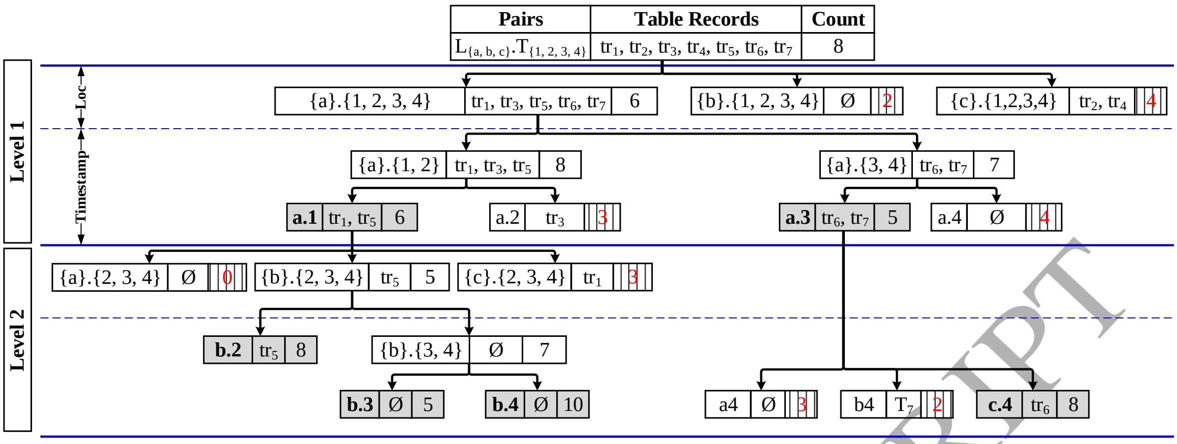


Figure 2: Noisy prefix tree of the trajectories in Table 1

in the noisy prefix tree, including general nodes, are only part of the building process.

4.2. Building the Noisy Prefix Tree

Our proposed algorithm builds a noisy prefix tree whereby trajectories are distributed among tree nodes based on noisy counts and prefixes. The idea is to construct tree level i by extending every node at level $i-1$. In order to satisfy differential privacy, a differentially-private operation should not depend on the underlying dataset. Recall a node is represented by a unique pair of location and timestamp; we extend every node by considering *all* possible combinations of location and timestamp given their respective universes. In other words, any location and timestamp pair that does not appear in the raw trajectories has a *non-zero probability* of appearing in the sanitized trajectories.

Example 4.1. Figure 2 presents a running example of a noisy prefix tree R, where the input dataset is the raw trajectories in Table 1. The *Root* of R is the first node at the top of the tree. Any node in R consists of three pieces of information: location and timestamp pair(s), a portion of the dataset trajectories, and a noisy count (except the *Root*). The height of the tree is 2, and each level consists of two sublevels separated by a dashed line. ■

A node is added to the noisy prefix tree, and thus qualifies for extension, if it is considered to be *non-general* and *non-empty*. A non-general node is represented by a specific location and a specific timestamp (as opposed to a general node as described in Section 4.1). A decision whether a node is non-empty is rendered based on the node's noisy count. Recall a node v's noisy count is stored in its own $c(v)$, a non-empty node is a node with $c(v) \geq \theta$, where θ is a pre-defined threshold computed as follows. Let node $v \in level(i)$, we define the noisy count threshold θ as a function of level i's allocated budget portion, ϵ_ℓ. More specifically, we define θ to be two times the standard deviation of the level's noise. Recall from Section 3.1 that the Laplace mechanism has variance $2\lambda^2$, where $\lambda = \frac{1}{\epsilon}$ for count queries as a utility function. Therefore, $\theta_{ng} = 2\frac{\sqrt{2}}{\epsilon_\ell}$, where θ_{ng} is defined for non-general nodes. The same concept is applied to general nodes when

Algorithm 1 *SafePath*

```
Input: Raw trajectory dataset $D$, privacy budget $\epsilon$
Input: Taxonomy trees: $A_{loc}$ and $A_{time}$
Input: Height of the noisy prefix tree $h$
Output: Differentially-private trajectory dataset $\hat{D}$
 1: Create a prefix tree $R$ with node $Root$;
 2: $tr(Root) \leftarrow$ all trajectories in $D$;
 3: $\epsilon_\ell = \frac{\epsilon}{h}$;
 4: Compute $\epsilon_g$ and $\epsilon_{ng}$ for loc. and timestamp sublevels;
 5: Compute $\theta_g$ and $\theta_{ng}$;
 6: $i = 1$;
 7: while $i \leq h$ do
 8:     for each non-general node in $level(i-1)$ do
 9:         $\mathcal{W} \leftarrow BuildSubLevel(A_{loc}, \epsilon_g, \epsilon_{ng}, \theta_g, \theta_{ng})$;
10:         for each non-general location node in $\mathcal{W}$ do
11:             $level(i) \leftarrow BuildSubLevel(A_{time}, \epsilon_g, \epsilon_{ng}, \theta_g, \theta_{ng})$;
12:     $i++$;
13: $\hat{D} \leftarrow EnhanceConsistency(R)$;
14: return $\hat{D}$;
```

building taxonomy trees. That is, a general node is deemed non-empty, and thus can be extended within its taxonomy tree, if its noisy count is not less than θ_g, where $\theta_g = 4\frac{\sqrt{2}}{\epsilon_\ell}$.

Example 4.2. For the sake of simplicity, let $\theta_{ng} = \theta_g = 5$ for the example in Figure 2. Any node with noisy count < 5 (barred) is considered empty and is not going to be extended. Grey nodes indicate the nodes that are added to the noisy prefix tree. ■

The above filtering technique prunes the nodes with small true counts, albeit with reasonable impact on data utility as suggested by the experiments performed on real-life datasets in Section 5. On the other hand, nodes that contain no trajectories (i.e., $tr(v) = 0$) get filtered out in the early stages of building the noisy prefix tree. This significantly improves utility and efficiency by preventing building and processing false trajectories. We next describe our entire approach, as summarized in Algorithm 1.

Algorithm 1 starts at Line 1 by creating a *Root* node under which the noisy prefix tree R will be built. Line 2 assigns all the trajectories of the input dataset D to $tr(Root)$, and Line 3 computes the privacy budget portion dedicated for each level. Lines 4 computes the privacy budget portions ϵ_g and ϵ_{ng} for the general and non-general nodes in both the location and timestamp sublevels. We note that the location and timestamp sublevels have the same pair of budget portions ϵ_g and ϵ_{ng} if taxonomy trees A_{loc} and A_{time} are identical (see Section 4.1). Line 5 computes the noisy count thresholds θ_g and θ_{ng}. Lines 7-12 build the noisy prefix tree by iteratively constructing every level $i \leq h$ in a breadth-first manner. Under every non-general node $v \in level(i-1)$, the algorithm attempts to extend v by first building the location sublevel (Line 9), and then under every non-empty *non-general location node*, the algorithm builds the timestamp sublevel (Line 10).

Procedure 1 *BuildSubLevel*

```
Input: Taxonomy tree $A$
Input: Privacy budgets $\epsilon_g$ and $\epsilon_{ng}$
Input: Thresholds $\theta_g$, and $\theta_{ng}$
Output: Set of non-general loc. or time. nodes $\mathcal{W}$
1: $\mathcal{W} = \emptyset$;
2: $\mathcal{C} \leftarrow BuildTaxonomy(A, \epsilon_g, \theta_g)$;
3: for each general node $C \in \mathcal{C}$ do
4:     for each non-general node $n$ in $C$'s children do
5:         $tr(n) = \{tr \in C \mid prefix(n) \preceq tr\}$;
6:         $c(n) = |tr(n)| + \texttt{Lap}(1/\epsilon_{ng})$;
7:         if $c(n) \geq \theta_{ng}$ then
8:             $\mathcal{W} \leftarrow n$;
9: return $\mathcal{W}$;
```

A non-general location node is a node that contains a specific location value but a general timestamp value (range interval). In Line 9, *BuildSubLevel* builds the location sublevel and returns the set of all the non-empty, non-general location nodes, $\mathcal{W}$. In Line 10, *BuildSubLevel* builds the timestamp sublevel and returns the set of all the non-empty, non-general nodes. Note that only the nodes returned by *BuildSubLevel* in Line 10 are added to the noisy prefix tree R.

Procedure 1 describes *BuildSubLevel*. Line 2 calls $BuildTaxonomy$ that builds a generalization hierarchy according to a user-input taxonomy tree A, where A is rooted at a non-general node from the previous prefix tree level. $BuildTaxonomy$ (presented in Procedure 2) returns the set of *non-empty leaf general nodes*, $\mathcal{C}$. Lines 3-8 of *BuildSubLevel* iterate through all the non-general nodes in $\mathcal{C}$ to determine if they are non-empty. We note that if *BuildSubLevel* is building a *location sublevel*, then $prefix(n) \preceq tr$ in Line 5 ignores the general timestamp information in the non-general location node n because the timestamp at this point is a range interval that is by default a prefix to any specific timestamp value that exists within the range.

4.3. Constructing the Sanitized Trajectories

In Section 4.2 we detailed the steps for building a noisy prefix tree R that contains the differentially-private trajectories, as summarized in Lines 7-12 of Algorithm 1. At this point, trajectories are sanitized and ready for release. Constructing the sanitized trajectories from R is performed as follows. Starting from the *Root* node, we traverse R such that each and every node v is visited exactly once. Upon visiting node v, where $v \neq Root$, we construct $c(v)$ copies of $prefix(v)$ (see Definition 3.4) and append them to the output dataset $\hat{D}$.

Since it is likely for a set of trajectories to terminate at a parent node, a parent node's true trajectory count is never less than the sum of its children's true counts. Considering that noise is added to true trajectory counts, the above rule may not hold for noisy

Procedure 2 *BuildTaxonomy*

```
Input: Taxonomy tree $A$
Input: Privacy budget $\epsilon_g$
Input: Threshold $\theta_g$
Output: Set of general loc. or time. nodes $\mathcal{C}$
j = 1;
$\mathcal{G} = \emptyset$;
while $j \leq$ height of $A$ do
    $\mathcal{G} \leftarrow$ set of general nodes in $A$ at level $j$;
    for each general node $G \in \mathcal{G}$ do
        $tr(G) = \{tr \in parent \mid prefix(G) \preceq tr\}$;
        $c(G) = |tr(G)| + \texttt{Lap}(1/\epsilon_g)$;
        if $c(G) < \theta_g$ then
            Remove $G$ and its descendants from $A$;
    $j++$;
$\mathcal{C} = \mathcal{G}$; // Non-empty leaf general nodes
return $\mathcal{C}$;
```

counts. Hence, we need to make sure that $c(v)$ is no less than the sum of its children's noisy counts. We define a *consistent noisy prefix tree* as follows:

Definition 4.1 (*Consistent Noisy Prefix Tree*)**.** *Given a noisy prefix tree R, $\forall v$ in R, where $v \neq$ Root, it holds that $c(v) \geq \sum_{u \in children(v)} c(u)$.* ■

The objective of *EnhanceConsistency* in Line 13 of Algorithm 1 is to enforce the above condition on the noisy prefix tree R before releasing the sanitized trajectories. We note that the *Root* node is excluded from the rule in Definition 4.1 because the first location-timestamp pair in any trajectory in the sanitized (output) dataset of Algorithm 1 belongs to the first level in R, and not the root level.

We introduce a post-processing step that modifies the noisy counts of the nodes in R such that Definition 4.1 is satisfied. Post-processing noisy counts has been applied in similar privacy problems whereby *consistent estimates* are computed to achieve more accurate results [25][9]. Consequently, the objective of *EnhanceConsistency* is to find the consistent estimate for each node in the noisy prefix tree R, except the *Root* node.

The consistent estimate, denoted by $\bar{c}(v)$, is described in terms of the noisy count $c(v)$ for each v in R. The consistent estimate is computed by traversing R starting from the first level and going towards the leaves. Let w be the parent of v and u be a sibling of v; the idea is to lower each noisy count $c(u) \in children(w)$ with respect to $\bar{c}(w)$ such that the sum of all the children's noisy counts does not exceed the parent's noisy count. The rationale behind decreasing the children's noisy counts (as opposed to increasing the parent's noisy count) stems from the fact that the sum of the children's noisy counts results in a numeric value that has $|children(w)|$ times the Laplacian noise of the parent's count. Intuitively, suppressing the extra noise helps in achieving a more accurate noisy version of the raw trajectories. The consistent estimate $\bar{c}(v)$ of node v is computed as follows:

$$\bar{c}(v) = \begin{cases} c(v), & v \in level(1) \\ c(v) + min(0, \dfrac{\bar{c}(w) - \sum_{u \in children(w)} \bar{c}(u)}{|children(w)|}), & \text{o.w.} \end{cases}$$

Lastly, the differentially-private trajectories are ready to be released. Line 14 of Algorithm 1 traverses the consistent noisy prefix tree in a top-down fashion. Each node v is visited exactly once, and $\bar{c}(v)$ copies of $prefix(v)$ are appended to the output dataset $\hat{D}$.

4.4. Theoretical Analysis

Performance Improvement. Employing a *simple* prefix tree structure does achieve a differentially-private trajectory dataset. Chen et al. [9] have suggested a similar approach towards publishing sanitized trajectories; we call it *SeqPT*. In this approach, no hierarchies are imposed; rather, each node is extended by considering every possible combination of location and timestamp. In other words, if an empty node passes threshold θ, then the subtree rooted at that node will consist solely of empty nodes. On the other hand, our proposed solution (as described in Algorithm 1) filters out empty nodes as early as possible, preventing false trajectories from being constructed. Subsequently, runtime is significantly reduced while data utility is greatly boosted. We present herein a formal analysis that estimates the factor by which the number of empty nodes of a simple prefix tree solution is reduced when applying our proposed method.

Lemma 4.1. *Let* $p(x) = \frac{1}{2\lambda}exp(\frac{-x}{\lambda})$ *be probability density function of the Laplace distribution. Given sensitivity* $\Delta f = 1$ *for a count queries-based function* f*, and privacy budget portion* ϵ_ℓ*, then* $\lambda = \frac{\Delta f}{\epsilon_\ell} = \frac{1}{\epsilon_\ell}$*. Hence,* $p(x) = \frac{\epsilon_\ell}{2}exp(-x\epsilon_\ell)$*. Given threshold* θ*, the probability of an empty node having noisy count* $x \geq \theta$ *is*

$$Pr_\theta = Pr[x \geq \theta] = \int_\theta^\infty \frac{\epsilon_\ell}{2}exp(-x\epsilon_\ell)dx = \frac{1}{2}exp(-\epsilon_\ell\theta). \blacksquare \qquad (4)$$

Recall from Section 4.2 that $\theta_{ng} = 2\frac{\sqrt{2}}{\epsilon_\ell}$ and $\theta_g = 4\frac{\sqrt{2}}{\epsilon_\ell}$. From Equation 4, the probabilities of an empty general node and an empty non-general node passing their respective thresholds are $Pr_{\theta_g} = \frac{exp(-4\sqrt{2})}{2}$ and $Pr_{\theta_{ng}} = \frac{exp(-2\sqrt{2})}{2}$.

Theorem 4.1. *Given location and timestamp taxonomy trees with heights* h_{t_1} *and* h_{t_2}*, respectively, and fan-out* F*, a noisy prefix tree (as described by Algorithm 1) reduces the number of empty nodes generated under empty node* $v \in level(i)$ *due to* v *having adequately large noisy count by a factor of* $(\frac{2^{(h_{t_1}+h_{t_2})+1}}{F^{(h_{t_1}+h_{t_2})}})^{h-i}exp(2\sqrt{2}(2(h_{t_1}+h_{t_2})+1)(h-i))$*, where* h *is the prefix tree height.* $\blacksquare$

Proof. Let empty node v be at level i in a given prefix tree. In a simple prefix tree, the estimated number of empty nodes descending from v is $\mathbb{E}_1 = (|\mathcal{L}||\mathcal{T}|Pr_{\theta_{ng}})^{h-i}$, where $\mathcal{L}$ is the location universe, $\mathcal{T}$ is the timestamp universe, and h is the height of the prefix tree. In a noisy prefix tree, the estimated number of empty nodes descending from

v, $\mathbb{E}_2$, is evaluated as follows. Let the heights of the location and timestamp taxonomy trees be h_{t_1} and h_{t_2}, respectively. Further, let F denote the maximum number of general child nodes given any taxonomy tree; thus, F^{h_t} is the number of leaf general nodes in a taxonomy tree with height h_t. This estimates the number of empty location nodes in one location sublevel to $F^{h_{t_1}}(Pr_{\theta_g}^{h_{t_1}} \cdot \frac{|\mathcal{L}|}{F^{h_{t_1}}} Pr_{\theta_{ng}}) = F^{h_{t_1}} Pr_{\theta_g}^{h_{t_1}} \cdot |\mathcal{L}| Pr_{\theta_{ng}}$ and the number of empty timestamp nodes in one timestamp sublevel *under one empty location node* to $F^{h_{t_2}}(Pr_{\theta_g}^{h_{t_2}} \cdot \frac{|\mathcal{T}|}{F^{h_{t_2}}} Pr_{\theta_{ng}}) = F^{h_{t_2}} Pr_{\theta_g}^{h_{t_2}} \cdot |\mathcal{T}| Pr_{\theta_{ng}}$. Consequently, the estimated number of empty non-general nodes in one level in the noisy prefix tree is $(F^{h_{t_1}} Pr_{\theta_g}^{h_{t_1}} \cdot |\mathcal{L}| Pr_{\theta_{ng}}) \cdot (F^{h_{t_2}} Pr_{\theta_g}^{h_{t_2}} \cdot |\mathcal{T}| Pr_{\theta_{ng}}) = F^{(h_{t_1}+h_{t_2})} |\mathcal{L}||\mathcal{T}| Pr_{\theta_g}^{(h_{t_1}+h_{t_2})} Pr_{\theta_{ng}}^2$. Hence, $\mathbb{E}_2 = (F^{(h_{t_1}+h_{t_2})} |\mathcal{L}||\mathcal{T}| Pr_{\theta_g}^{(h_{t_1}+h_{t_2})} Pr_{\theta_{ng}}^2)^{h-i}$. Given $\mathbb{E}_1$ and $\mathbb{E}_2$, the reduction factor is estimated by the following equation:

$$\begin{aligned}
\frac{\mathbb{E}_1}{\mathbb{E}_2} &= \frac{(|\mathcal{L}||\mathcal{T}| Pr_{\theta_{ng}})^{h-i}}{((F^{(h_{t_1}+h_{t_2})} |\mathcal{L}||\mathcal{T}| Pr_{\theta_g}^{(h_{t_1}+h_{t_2})} Pr_{\theta_{ng}}^2)^{h-i})^{h-i}} \\
&= \frac{1}{(F^{(h_{t_1}+h_{t_2})} Pr_{\theta_g}^{(h_{t_1}+h_{t_2})} Pr_{\theta_{ng}})^{h-i}} \\
&= \frac{1}{(F^{(h_{t_1}+h_{t_2})} \cdot \frac{1}{2^{(h_{t_1}+h_{t_2})}} exp(-4(h_{t_1}+h_{t_2})\sqrt{2}) \cdot \frac{1}{2} exp(-2\sqrt{2}))^{h-i}} \qquad (5) \\
&= \frac{1}{(\frac{F^{(h_{t_1}+h_{t_2})}}{2^{(h_{t_1}+h_{t_2})+1}})^{h-i} (exp(-2\sqrt{2}(2(h_{t_1}+h_{t_2})+1)))^{h-i}} \\
&= (\frac{2^{(h_{t_1}+h_{t_2})+1}}{F^{(h_{t_1}+h_{t_2})}})^{h-i} exp(2\sqrt{2}(2(h_{t_1}+h_{t_2})+1)(h-i)).
\end{aligned}$$

□

Privacy Analysis. Algorithm 1 first builds a noisy prefix tree by accessing the raw trajectories, and then invokes *EnhanceConsistency* to achieve a consistent noisy prefix tree. Building a noisy prefix tree is performed by iteratively constructing one level at a time. Each level is dedicated privacy budget portion $\epsilon_\ell = \frac{\epsilon}{h}$, where h is the noisy prefix tree height. One prefix tree level consists of two sublevels: location and timestamp, each is dedicated $\epsilon_s = \frac{\epsilon_\ell}{2}$. Within a sublevel, the collective budget portions $\epsilon_{ng} + \sum \epsilon_g$ dedicated for non-general and general nodes add to ϵ_s; i.e., $\frac{(|\mathcal{U}|-2\sum_{x=1}^{d} x)\epsilon_s}{|\mathcal{U}|} + \frac{2\epsilon_s \sum_{x=1}^{d} x}{|\mathcal{U}|} = \epsilon_s$, where d is a taxonomy tree level. We note that sequential composition (Lemma 3.1) is used for designing ϵ_g and ϵ_{ng} because of the following reason. Given a taxonomy tree in a sublevel of the noisy prefix tree, a path that starts from the root node to the leaf general node consumes a budget equal to the summation of all the budgets computed at every general level of the taxonomy tree.

The algorithm consumes a privacy budget that amounts to $(2 \times \epsilon_s) \times h = \epsilon$. Such a budget allocation scheme leverages sequential and parallel compositions (Lemmas 3.1 and 3.2). That is, the total budget consumed along a $Root - to - leaf$ path amounts to ϵ, whereas the total budget needed for any group of sibling nodes is equal to the same budget portion dedicated for a single node within the group.

EnhanceConsistency post-processes the differentially-private data (the noisy counts in the noisy prefix tree) without accessing the underlying raw trajectories. Consequently, *EnhanceConsistency* maintains the same differential privacy guarantees because post-processing differentially-private data has no impact on the privacy guarantees. We refer the reader to [25] for a proof.

Theorem 4.2. *Given ϵ as a user-input privacy budget, Algorithm 1 is ϵ-differentially private.* ■

Complexity Analysis. We analyze the runtime of Algorithm 1 with respect to its input parameters by first examining Lines 7-12, which build the noisy prefix tree, followed by *EnhanceConsistency*. Building the noisy prefix tree starts by building a taxonomy tree, requiring distributing trajectories from the parent node to its children. At any level in the taxonomy tree[1] there exists exactly $|D|$ trajectories in total. Therefore, a single level in the noisy prefix tree requires scanning the input dataset $O((h_{t_1} + h_{t_2})|D|) = O(h_t|D|)$ times, where h_{t_1} and h_{t_2} are the heights of the location and timestamp taxonomies, respectively, and $h_t = max\{h_{t_1}, h_{t_2}\}$.

Generating nodes is a costly operation. Generating general nodes does not depend on the underlying data and is done in linear time with respect to the user-defined taxonomy. We now estimate the number of non-general nodes in the noisy prefix tree as follows. At worst, Algorithm 1 fails to filter out empty nodes through general nodes; i.e., all general nodes are constructed according to their taxonomies. Recall the timestamp sublevel extends the non-general location nodes in the location sublevel. Intuitively, the number of non-general nodes in the timestamp sublevel is much greater (no less at best) than that at the location sublevel. Therefore, we estimate the number of generated nodes at the timestamp sublevel to be $|D| + k$, where k represents the number of empty nodes from $\mathcal{T}$ that pass the noisy count threshold θ_{ng}. k is a random variable that follows a *binomial distribution* and can be estimated according to Theorem 4.3 [13].

Theorem 4.3. *Let n denote the total number of empty nodes, each having a success probability Pr_θ. The number of successes, k, in a series of independent pass/fail experiments on each node follows a binomial distribution $Bin(n, Pr_\theta)$, denoted by $k \sim Bin(n, Pr_\theta)$.* ■

Let $k'_l \sim Bin(|\mathcal{L}| - |D|, Pr_\theta)$, $k''_l \sim Bin(|\mathcal{L}|, Pr_\theta)$, $k'_t \sim Bin(|\mathcal{T}| - |D|, Pr_\theta)$, and $k''_t \sim Bin(|\mathcal{T}|, Pr_\theta)$. The number of generated nodes at the timestamp sublevel at level $i \leq h$ is estimated to be $|D|(1 + k'_t + \mathbb{Z}) + (k'_l k''_t \cdot (k''_l k''_t)^{i-1})$, where

$$\mathbb{Z} = \begin{cases} 0, & i = 1 \\ \displaystyle\sum_{j=0}^{i-2}((k''_l k''_t)^j [k'_l k''_t + k'_t k''_l k''_t]), & i > 1. \end{cases}$$

Therefore, generating non-general nodes in a prefix tree of height h runs in $O(|D|(k''_l k''_t)^h)$, where $k''_l \ll \mathcal{L}$ and $k''_t \ll \mathcal{T}$. Subsequently, building the noisy prefix tree runs in $O(hh_t|D| + |D|(k''_l k''_t)^h) = O(|D|(hh_t + (k''_l k''_t)^h))$, where $k''_l \ll \mathcal{L}$ and $k''_t \ll \mathcal{T}$.

[1] Assuming the taxonomy tree is a perfect F-ary tree.

Table 2: Summary statistics of the transit datasets

Dataset	$\|\mathcal{D}\|$	$\|\mathcal{L}\|$	$\|\mathcal{T}\|$	$max\|tr\|$	$avg\|tr\|$
Bus	773,296	893	168	121	4.69
Metro	847,668	68	168	90	3.22

EnhanceConsistency does not require scanning the input dataset; rather, this procedure performs one operation, which is computing the consistent estimate for each node. This is achieved by scanning the noisy prefix tree once in a top-down fashion whereby each node is visited twice. Therefore, *EnhanceConsistency* is performed in linear time. Lastly, Algorithm 1 scans the consistent noisy prefix tree once starting from *Root* to release the sanitized trajectories.

In closing, Algorithm 1 runs in $O(|D|(hh_t + (k_l''k_t'')^h))$ at worst. However, this is a theoretical conclusion of the worst-case scenario. In real-life situations, h is a fairly small integer and the number of generated empty nodes at level i is much smaller than $O(|D|(k_l''k_t'')^i)$ because general nodes attempt to filter out as many empty nodes as possible. In the next section, experiments on real-life datasets successfully sanitize extremely sparse trajectories within seconds.

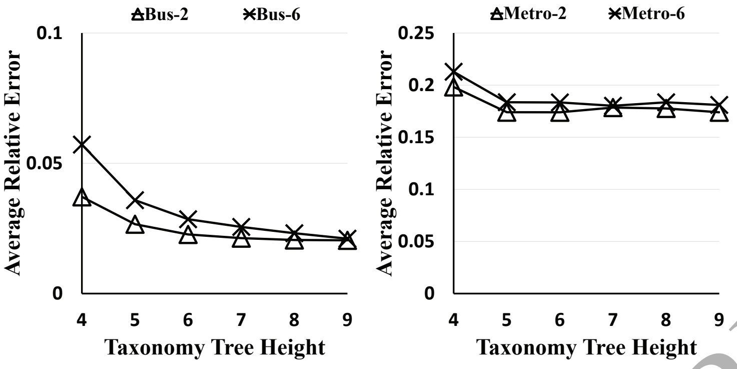
5. Experimental Evaluation

In this section, we thoroughly evaluate our proposed differentially-private trajectory sanitization method, *SafePath*, using real-life trajectories. Evaluation encompasses three criteria: utility of the sanitized data, efficiency of *SafePath* in terms of runtime, and scalability for handling large datasets. Two real-life trajectory datasets are used for conducting experiments: *Bus* and *Metro*, each containing the paths of passengers traveling through the Montreal bus and metro transit networks, respectively. Table 2 contains summary statistics of each dataset, where $|\mathcal{D}|$ is the number of records in D where each record represents a trajectory of a unique passenger, $|\mathcal{L}|$ is the size of the location universe, $|\mathcal{T}|$ is the size of the time universe, $max|tr|$ is the maximum trajectory length, and $avg|tr|$ is the average trajectory length. Throughout the experiments conducted in this section, we note the following about the choice of the value of the noisy prefix tree height h. In order to capture enough information from the underlying dataset, h should be large enough. While this is true, our *Metro* and *Bus* datasets have average trajectory lengths of 4.69 and 3.22, respectively, as described in Table 2. Therefore, we choose h to be around the average in order to avoid outliers.

SafePath is implemented in C++, and all the experiments in this section are performed on a 64-bit Windows 7 running on an Intel Core 2 Due 2.13GHz PC with 8GB RAM.

5.1. Utility and Efficiency

We examine the impact of the different parameters of *SafePath* on utility and runtime. Similar to the evaluation methodology of [9] and [49], we measure the utility of a sanitized dataset by issuing a set of randomly generated count queries. Specifically, we issue 40,000 count queries that randomly draw values from the location and timestamp universes



(a) *Bus* (b) *Metro*

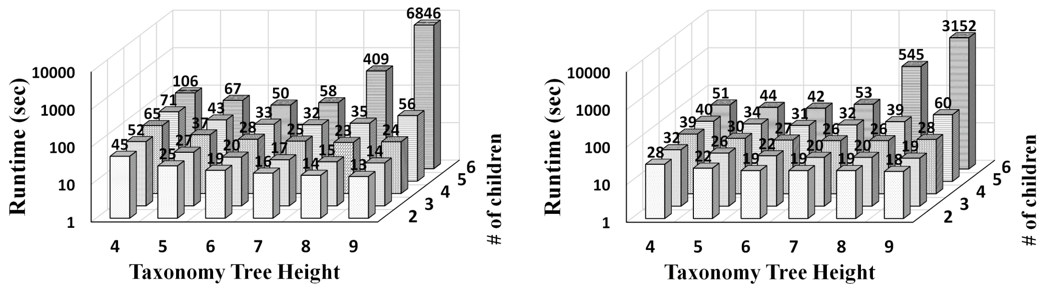
Figure 3: Average relative error vs. taxonomy tree height

following an even distribution. For every query issued over the sanitized dataset, we compute its relative error, then we average all the errors and report the latter value in the graphs herein. The *length* of a count query, denoted by $|q|$, refers to the number of location and timestamp pairs in the query. For example, $q = L_1.T_1 \rightarrow L_2.T_2$ has length = 2. Finally, we set the sanity bound to 0.1% of the underlying dataset.

Taxonomy Trees. Filtering out empty nodes in the early stages of building the prefix tree is especially important for high-dimensional and sparse data, such as trajectory data. Therefore, in the next set of experiments we examine the impact of location and timestamp taxonomy trees on utility and runtime. A taxonomy tree is defined by two parameters: *taxonomy tree height*, and number of *child nodes* (or simply *children*). The former parameter defines the number of levels in the tree, whereas the latter parameter refers to the maximum number of child nodes that belong to the same parent node. Both parameters can either be public knowledge (e.g., a metro map) or user-defined. Without loss of generality, one taxonomy tree setting will be applied to both location and timestamp hierarchies.

To demonstrate the impact of taxonomy trees on relative error, Figure 3 reports the average relative errors on *Bus* and *Metro* with respect to the two taxonomy tree parameters: *height* and *number of child nodes*. Let the location and timestamp taxonomy trees have the same values on the same parameter. We vary the taxonomy tree height (the x-axis in Figure 3). We set the number of child nodes to be 2 and 6, denoted in the graphs by *Bus-2*/*Metro-2* and *Bus-6*/*Metro-6*, respectively. Finally, We set the query length $|q| = h = 2$. Figure 3 suggests that the relative error decreases as taxonomy tree height increases and the number of child nodes decreases. This is because fewer empty non-general nodes are generated throughout the process of building the prefix tree, where empty nodes get filtered out by the taxonomy trees. For taxonomy tree height ≥ 6 the relative error does not exhibit significant change. Moreover, we observe that on both datasets no significant drop in error results from decreasing the number of child nodes.

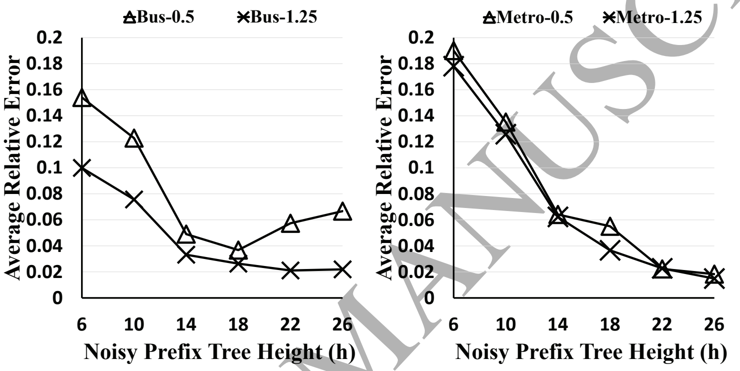
Figure 4 shows how runtime varies under different values of taxonomy tree height and



(a) Bus

(b) Metro

Figure 4: Runtime vs. taxonomy tree



(a) *Bus*

(b) *Metro*

Figure 5: Average relative error vs. noisy prefix tree height

the number of child nodes. Figure 4(a) and Figure 4(b) study *Bus* and *Metro*, respectively, where the noisy prefix tree height $h = 2$, the privacy budget $\epsilon = 1$, and the runtime axis is set on a logarithmic scale. Figure 4 suggests that runtime increases as the number of children increases for all values of taxonomy tree height. This is because larger values of children allow for more empty general nodes to be mistakenly considered non-empty due to the added noise, resulting in a much larger number of empty non-general nodes that require extra processing. Moreover, we observe that runtime decreases non-monotonically as the taxonomy tree height increases. More specifically, runtime decreases for a smaller number of children, less than 4, after which it starts increasing up to a significant value at taxonomy tree height = 9 and number of children = 6. The increase in runtime for a larger number of children can be noticed at taxonomy tree height ≥ 6. These observations are shared by both *Bus* and *Metro*.

Figures 3 and 4 suggest that a certain range of taxonomy trees allows for a significant improvement in terms of runtime without compromising utility. Unless otherwise mentioned, all our subsequent experiments will have taxonomy tree height = 6 and children = 2.

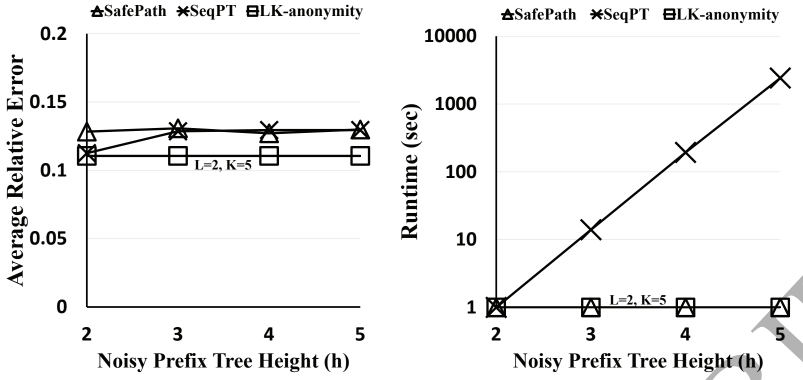
Noisy Prefix Tree Height and Privacy Budget. Figure 5 examines the effect of varying the noisy prefix tree height, h, and the privacy budget, ϵ, on the relative error on both *Bus* and *Metro*. The parameters are set as follows: $6 \leq h \leq 26$, $|q| = h/3$, and $\epsilon = 0.5$ and 1.25 denoted in the graph by *Bus-0.5/Metro-0.5* and *Bus-1.25/Metro-1.25*, respectively. Figure 5 reports a general decrease in the relative error when h increases. Increasing h allows us to retrieve and sanitize more data from the raw trajectories, however, up until a certain threshold, after which the added noise at each level of the noisy prefix tree becomes more dominant, causing the relative error to increase. This is more evident in sparse datasets with low privacy budget (*Bus-0.5* in Figure 5 (a)) because more empty nodes are generated upon building the noisy prefix tree to satisfy differential privacy. Lastly, increasing ϵ successfully manages to reduce the relative error for all values of h. It is interesting to see that our proposed method is capable of reducing the relative error below 1% on both datasets, suggesting that *SafePath* maintains high utility even for extremely sparse data.

5.2. Comparisons

SeqPT. In [9], Chen et al. proposed a prefix tree-based method for sanitizing sequential locations under differential privacy. Further, they suggested that their method can be extended to trajectories - series of locations *coupled with* timestamps. To do so, we implemented their suggested extension that exhaustively considers every possible combination of location and timestamp under every node throughout the process of building the prefix tree. We call their extension *SeqPT*. Moreover, we implemented LK-anonymity [23], a syntactic-based privacy approach that makes sure that every L-sized sequence of location and timestamp pairs from the input dataset is shared by at least K trajectories via means of suppression [45][11].

We carry out performance evaluation with respect to average relative error and runtime. *SeqPT* takes nearly 11 minutes to sanitize *Metro* when setting the prefix tree height $h = 2$, and fails to complete a single run on *Metro* when $h \geq 3$ and on *Bus* when $h \geq 2$. This is because the number of generated nodes increases exponentially as h increases. For this reason, we reduce the dimensionality of *Bus* to a maximum of 20 locations and 24 timestamps. A pair that does not fall within the shrunken universe will be excluded from the dataset. This results in a dataset with 200,000 records and a maximum trajectory length of 8. For this set of experiments we generate 10,000 non-empty random queries of size 2, set the privacy budget $\epsilon = 1$, and vary the prefix tree height $2 \leq h \leq 5$. We note that unlike the experiments in Figure 5, which allowed for the choice of larger values of h due to our proposed method's efficient design, the choice of the value range of h in this particular set of experiments is limited due to the exhaustive nature of *SeqPT*. Lastly, for LK-anonymity, we choose $L = 2$ (because the average trajectory length is 1.5 in the new version of *Bus*) and $K = 5$.

Figure 6 studies the performance of *SafePath*, *SeqPT*, and LK*-anonymity* in terms of utility in Figure 6 (a) and runtime in Figure 6 (b), where the runtime axis is set on a logarithmic scale. We notice that the utility achieved by *SafePath* is comparable to that achieved by *SeqPT* and LK*-anonymity* for all values of h. On the other hand, *SafePath* substantially lowers runtime from 2400 seconds to 1 second at $h = 5$.



(a) Average relative error vs. h (b) Runtime vs. h

Figure 6: *SafePath* vs. *SeqPT* vs. *LK-anonymity*

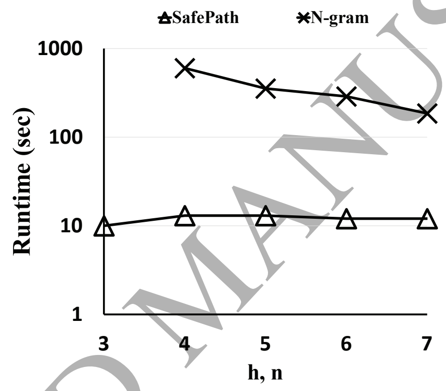


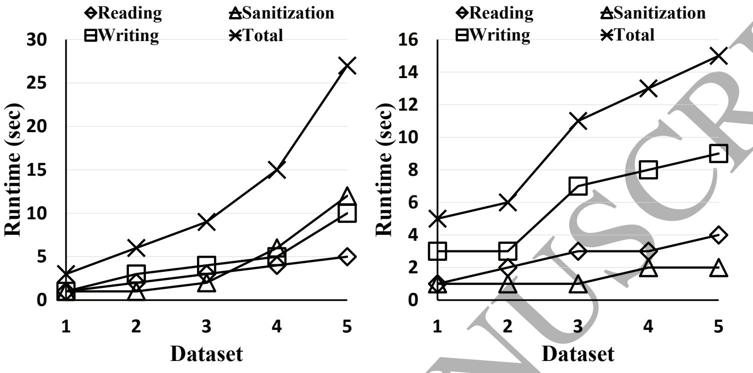
Figure 7: Comparing *SafePath* vs. *N-gram*

N-gram. The authors of [8] proposed *N-gram*: a sequential data sanitization method that integrates the Markov assumption to synthesize differentially-private sequential data. The n in $N-gram$ defines the maximum length of a sanitized sequence (before publishing the synthetic data).

In this set of experiments, we transform our *metro* dataset, which consists of trajectories containing a sequence of location and timestamp pairs, to another version that contains unique "locations". That is, every unique location and timestamp pair is mapped to a unique "location". The resulting dimensionality of the transformed dataset is $|\mathcal{L}| \times |\mathcal{T}|$, where $\mathcal{L}$ and $\mathcal{T}$ are the respective universes of locations and timestamps in the source dataset. For example, trajectory $L_1.T_1 \rightarrow L_1.T_2$ is transformed to sequence $Loc_1 \rightarrow Loc_2$. Both our noisy prefix tree height h and the maximum sequence length n of $N - gram$ describe the height of a prefix tree (Definition 3.4) in the underlying sanitization method. Figure 7 compares *SafePath* with *N-gram* in terms of runtime (in seconds) by varying $3 \leq h, n \leq 7$, where the x-axis is set on a logarithmic scale. We can see that our method performs significantly less than *N-gram*, which takes several minutes to produce

Table 3: Summary statistics of sub-datasets

	$\vert\mathcal{T}\vert$		$\vert\mathcal{D}\vert$		$max\vert tr\vert$		$avg\vert tr\vert$	
Dataset	*Bus*	*Metro*	*Bus*	*Metro*	*Bus*	*Metro*	*Bus*	*Metro*
1	35	35	304,238	263,303	21	16	1.96	1.58
2	70	70	508,283	485,049	44	35	3.12	2.28
3	105	105	618,238	606,917	70	51	3.98	2.81
4	140	140	722,786	766,419	106	73	4.46	3.11
5	168	168	773,296	847,668	121	90	4.69	3.22



(a) *Bus* (b) *Metro*

Figure 8: Runtime vs. $|\mathcal{T}|$ and $|D|$

a differentially-private synthetic dataset. At $n = 3$, *N-gram* failed to finish and produced an out-of-memory error. We note that for all the values of n, *N-gram* produced a dataset with a maximum sequence length of 2, whereas our method was able to retain more information by producing datasets with longer sequences. In all the experiments in Figure 7, both methods achieved similar average relative errors for query length $|q| = 2$.

5.3. Scalability

We study the scalability of our approach by varying the *size* of the input dataset. In this set of experiments, a dataset size is determined by *both* the universe size and the number of records. We limit the timestamp universe $\mathcal{T}$ of *Bus* and *Metro* to a threshold $\leq |\mathcal{T}|$, and only consider the portion of the trajectories that satisfy the timestamp limit. By varying the threshold, we achieve different datasets with smaller timestamp universes and number of records than D. We consider limiting only the timestamp universe because, unlike the location universe, the former defines the length of trajectories in the dataset. Table 3 contains summary statistics of 5 datasets generated from *Bus* and 5 datasets generated from *Metro*. All the scalability experiments are reported in Figure 8, where $h = 12$ and $\epsilon = 1$.

Figure 8 measures how runtime varies as we change the timestamp universe size $|\mathcal{T}|$ and number of records $|D|$ of the input raw dataset. Each value on the x-axis in Figures 8 (a) and (b) represents a sub-dataset extracted from *Bus* and *Metro*, respectively, as illustrated in Table 3. Runtime increases reasonably with the increase of the input dataset size. We observe that sanitization time increases faster on the *Bus* sub-datasets because *Bus* has a larger location universe size compared with *Metro*; hence, more processing of empty nodes is required. In total, our method takes 27 seconds to process the full *Bus* dataset and 15 seconds to process the full *Metro* dataset.

In summary, our proposed method achieves high data utility for count queries and is robust to handle large-scale and extremely sparse trajectory data without compromising runtime. Such robustness stems from the ability to fine-tune taxonomy trees according to the desired utility. In order to achieve a reasonable balance between utility and runtime, experiments on taxonomy trees suggest minimizing the number of children and choosing a moderate taxonomy tree height.

6. Conclusion

We study the problem of publishing sanitized trajectories under the rigorous model of differential privacy. Trajectories are spatio-temporal data characterized by being high-dimensional and sparse due to the existence of a time dimension; hence, extra care is required in the sanitization process in order to handle such data efficiently. We propose *SafePath*, an efficient and scalable sanitization method for publishing differentially-private trajectories. *SafePath* structures trajectories as a noisy prefix tree, then performs a post-processing step that enhances the utility of the sanitized data. The authors of [9] proposed sanitizing sequential data and suggested that their method can be extended to trajectories. We implement their method and provide theoretical and experimental analysis on real-life trajectory datasets. Evaluation suggests that applying the above extension to trajectories fails to provide a scalable solution, whereas our proposed method demonstrates significant improvement in terms of efficiency and scalability with comparable data utility.

Acknowledgments

This research is supported by the Discovery Grants (RGPIN-2018-03872) from the Natural Sciences and Engineering Research Council of Canada, Canada Research Chairs Program (950-230623), and the Research Cluster Award (R16083) from Zayed University.

References

References

[1] S. Abraham and P. S. Lal. Spatio-temporal similarity of network-constrained moving object trajectories using sequence alignment of travel locations. *Transportation Research Part C: Emerging Technologies*, 23:109 – 123, 2012. Data Management in Vehicular Networks.

[2] O. Abul, F. Bonchi, and M. Nanni. Never walk alone: Uncertainty for anonymity in moving objects databases. In *Proceedings of the 24th IEEE International Conference on Data Engineering*, ICDE '08, pages 376–385, 2008.

[3] R. Agrawal and R. Srikant. Mining sequential patterns. In *Proceedings of the 11th International Conference on Data Engineering*, ICDE '95, pages 3–14, 1995.

[4] R. Assam, M. Hassani, and T. Seidl. Differential private trajectory protection of moving objects. In *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on GeoStreaming*, IWGS '12, pages 68–77, 2012.

[5] B. Barak, K. Chaudhuri, C. Dwork, S. Kale, F. McSherry, and K. Talwar. Privacy, accuracy, and consistency too: A holistic solution to contingency table release. In *Proceedings of the 26th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '07, 2007.

[6] M. Burger, M. van den Berg, A. Hegyi, B. D. Schutter, and J. Hellendoorn. Considerations for model-based traffic control. *Transportation Research Part C: Emerging Technologies*, 35:1 – 19, 2013.

[7] Y. Cao and M. Yoshikawa. Differentially private real-time data release over infinite trajectory streams. In *Proceedings of the 16th IEEE International Conference on Mobile Data Management - Volume 02*, MDM '15, pages 68–73, 2015.

[8] R. Chen, G. Acs, and C. Castelluccia. Differentially private sequential data publication via variable-length n-grams. In *Proceedings of the ACM Conference on Computer and Communications Security*, CCS '12, pages 638–649, 2012.

[9] R. Chen, B. C. Fung, B. C. Desai, and N. M. Sossou. Differentially private transit data publication: A case study on the montreal transportation system. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 213–221, 2012.

[10] R. Chen, B. C. M. Fung, N. Mohammed, and B. C. Desai. Privacy-preserving trajectory data publishing by local suppression. *Information Sciences: Special Issue on Data Mining for Information Security*, 231:83–97, 2013.

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

[12] B. Coifman, D. Beymer, P. McLauchlan, and J. Malik. A real-time computer vision system for vehicle tracking and traffic surveillance. *Transportation Research Part C: Emerging Technologies*, 6(4):271 – 288, 1998.

[13] G. Cormode, C. Procopiuc, D. Srivastava, and T. T. L. Tran. Differentially private summaries for sparse data. In *Proceedings of the 15th International Conference on Database Theory*, ICDT '12, pages 299–311, 2012.

[14] C. Dwork. Differential privacy. In *Proceedings of the 33rd International Conference on Automata, Languages and Programming - Volume Part II*, ICALP'06, pages 1–12, 2006.

[15] C. Dwork. Differential privacy: A survey of results. In *Proceedings of the 5th International Conference on Theory and Applications of Models of Computation*, TAMC'08, pages 1–19, 2008.

[16] C. Dwork. A firm foundation for private data analysis. *Communications of the ACM*, 54(1):86–95, 2011.

[17] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the 3rd Conference on Theory of Cryptography*, TCC'06, pages 265–284, 2006.

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

[19] L. Fan, L. Xiong, and V. Sunderam. Differentially private multi-dimensional time series release for traffic monitoring. In *Proceedings of the 27th Annual IFIP WG 11.3 Conference on Data and Applications Security and Privacy XXVII - Volume 7964*, DBSec 2013, pages 33–48, 2013.

[20] L. Fan, L. Xiong, and V. Sunderam. Fast: Differentially private real-time aggregate monitor with filtering and adaptive sampling. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, SIGMOD '13, pages 1065–1068, 2013.

[21] S. Gambs, M.-O. Killijian, and M. N. n. del Prado Cortez. Show me how you move and i will tell you who you are. In *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Security and Privacy in GIS and LBS*, SPRINGL '10, pages 34–41, 2010.

[22] S. R. Ganta, S. P. Kasiviswanathan, and A. Smith. Composition attacks and auxiliary information in data privacy. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 265–273, 2008.

[23] M. Ghasemzadeh, B. C. Fung, R. Chen, and A. Awasthi. Anonymizing trajectory data for passenger flow analysis. *Transportation Research Part C: Emerging Technologies*, 39:63 – 79, 2014.

[24] P. J. Haas and A. N. Swami. Sequential sampling procedures for query size estimation. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, SIGMOD '92, pages 341–350, 1992.

[25] M. Hay, V. Rastogi, G. Miklau, and D. Suciu. Boosting the accuracy of differentially private histograms through consistency. *Proceedings of the VLDB Endowment*, 3(1-2):1021–1032, 2010.

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

[27] S.-S. Ho. Preserving privacy for moving objects data mining. In *Proceedings of the IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages 135–137, 2012.

[28] S.-S. Ho and S. Ruan. Preserving privacy for interesting location pattern mining from trajectory data. *Transactions on Data Privacy*, 6(1):87–106, 2013.

[29] H. Hu, J. Xu, S. T. On, J. Du, and J. K.-Y. Ng. Privacy-aware location data publishing. *ACM Transactions on Database Systems (TODS)*, 35:18:1–18:42, 2010.

[30] K. Jiang, D. Shao, S. Bressan, T. Kister, and K.-L. Tan. Publishing trajectories with differential privacy guarantees. In *Proceedings of the 25th International Conference on Scientific and Statistical Database Management*, SSDBM, pages 12:1–12:12, 2013.

[31] D. Kifer. Attacks on privacy and definetti's theorem. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, SIGMOD '09, pages 127–138, 2009.

[32] D. Leoni. Non-interactive differential privacy: A survey. In *Proceedings of the 1st International Workshop on Open Data*, WOD '12, pages 40–52, 2012.

[33] X. Li, J. Han, J.-G. Lee, and H. Gonzalez. Traffic density-based discovery of hot routes in road networks. In *Proceedings of the 10th International Conference on Advances in Spatial and Temporal Databases*, SSTD'07, pages 441–459, 2007.

[34] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. L-diversity: Privacy beyond k-anonymity. volume 1, 2007.

[35] F. McSherry. Privacy integrated queries. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, SIGMOD '09, pages 19–30, 2009.

[36] F. McSherry and K. Talwar. Mechanism design via differential privacy. In *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '07, pages 94–103, 2007.

[37] A. Monreale, G. Andrienko, N. Andrienko, F. Giannotti, D. Pedreschi, S. Rinzivillo, and S. Wrobel. Movement data anonymity through generalization. *Transactions on Data Privacy*, 3(2):91–121, 2010.

[38] M. E. Nergiz, M. Atzori, and Y. Saygin. Towards trajectory anonymization: A generalization-based approach. In *Proceedings of the SIGSPATIAL ACM GIS International Workshop on Security and Privacy in GIS and LBS*, SPRINGL '08, pages 52–61, 2008.

[39] R. G. Pensa, A. Monreale, F. Pinelli, and D. Pedreschi. Pattern-preserving k-anonymization of sequences and its application to mobility data mining. In *Proceedings of the 1st International Workshop on Privacy in Location-Based Applications*, 2008.

[40] V. Primault, S. B. Mokhtar, C. Lauradoux, and L. Brunie. Time distortion anonymization for the publication of mobility data with high utility. In *Proceedings of the 14th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, pages 539–546, 2015.

[41] W. Qardaji, W. Yang, and N. Li. Understanding hierarchical methods for differentially private histograms. *Proceedings of the VLDB Endowment*, 6(14):1954–1965, 2013.

[42] R. Sherkat, J. Li, and N. Mamoulis. Efficient time-stamped event sequence anonymization. *ACM Transactions on the Web (TWEB)*, 8(1):4:1–4:53, 2013.

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

[44] L.-A. Tang, Y. Zheng, J. Yuan, J. Han, A. Leung, W.-C. Peng, and T. L. Porta. A framework of traveling companion discovery on trajectory data streams. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(1):3:1–3:34, 2014.

[45] M. Terrovitis and N. Mamoulis. Privacy preservation in the publication of trajectories. In *Proceedings of the The Ninth International Conference on Mobile Data Management*, MDM '08, pages 65–72, 2008.

[46] J. Wang, S. Liu, and Y. Li. A review of differential privacy in individual data release. *International Journal of Distributed Sensor Networks*, 2015:1:1–1:1, 2016.

[47] R. C.-W. Wong, A. W.-C. Fu, K. Wang, and J. Pei. Minimality attack in privacy preserving data publishing. In *Proceedings of the 33rd International Conference on Very Large Data Bases*, VLDB '07, pages 543–554, 2007.

[48] X. Xiao, G. Bender, M. Hay, and J. Gehrke. ireduct: Differential privacy with reduced relative errors. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, SIGMOD '11, pages 229–240, 2011.

[49] X. Xiao, G. Wang, and J. Gehrke. Differential privacy via wavelet transforms. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 23(8):1200–1214, 2011.

[50] Y. Xiao and L. Xiong. Protecting locations with differential privacy under temporal correlations. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, pages 1298–1309, 2015.

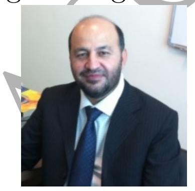
[51] R. Yarovoy, F. Bonchi, L. V. S. Lakshmanan, and W. H. Wang. Anonymizing moving objects: how to hide a mob in a crowd? In *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, EDBT '09, pages 72–83, 2009.

[52] D. Zekri, B. Defude, and T. Delot. Building, sharing and exploiting spatio- temporal aggregates in vehicular networks. *Mobile Information Systems*, 10(3):259–285, 2014.

[53] Y. Zheng, N. J. Yuan, K. Zheng, and S. Shang. On discovery of gathering patterns from trajectories. In *Proceedings of the IEEE International Conference on Data Engineering*, ICDE '13, pages 242–253, 2013.

Khalil Al-Hussaeni received his Ph.D. degree in 2017 from the Faculty of Engineering & Computer Science, Concordia University, Montreal, Canada. His doctoral thesis, ranked "Outstanding", proposed efficient and scalable techniques for anonymizing high-dimensional data. His research interest goes under the umbrella of privacy-preserving data publishing. Particularly, his research targets anonymizing relational data, trajectories, data streams, and Big Data for various data mining tasks. He was a research assistant in the Computer Security Laboratory at Concordia University, and is currently a member of the Data Mining and Security Laboratory research team at McGill University. Dr. Al-Hussaeni received his Bachelor's degree in Computer Engineering from University of Sharjah, U.A.E., and a Master's degree in Information Systems Security in 2009 from Concordia Institute for Information Systems Engineering, Concordia University, Canada.

Benjamin C. M. Fung is a Canada Research Chair in Data Mining for Cybersecurity, an Associate Professor of Information Studies at McGill University, a Co-curator of Cybersecurity in the World Economic Forum (WEF), and an Associate Editor of Elsevier Sustainable Cities and Society (SCS). He received a Ph.D. degree in computing science from Simon Fraser University in 2007. He has over 100 refereed publications that span the research forums of data mining, privacy protection, cyber forensics, 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.



Farkhund Iqbal holds the position of Associate Professor and Director Advanced Cyber Forensics Research Laboratory in the College of Technological Innovation, Zayed University, United Arab Emirates. He holds a Master (2005) and a Ph.D. degree (2011) from Concordia University, Canada. He is using machine learning and Big Data techniques for problem solving in healthcare, cybersecurity and cybercrime investigation in smart and safe city domain. He has published more than 80 papers in high ranked journals and conferences. He is an affiliate professor in school of information studies, McGill university, Canada and Adjunct professor in Adjunct Professor, Faculty of Business and IT, University of Ontario Institute of Technology, Canada. He is the recipient of several prestigious awards and research grants. He has served as a chair and TPC member of several IEEE/ACM conferences and is the reviewer of high rank journals. He is the member of several professional organization including ACM and IEEE Digital society.