Differentially Private Transit Data Publication: A Case Study on the Montreal Transportation System

Rui Chen
Concordia University
Montreal, QC, Canada
ru_che@encs.concordia.ca

Benjamin C. M. Fung
Concordia University
Montreal, QC, Canada
Benjamin.Fung@concordia.ca

Bipin C. Desai
Concordia University
Montreal, QC, Canada
BipinC.Desai@concordia.ca

Nériah M. Sossou
Société de transport de Montréal
Montreal, QC, Canada
Neriah.Sossou@stm.info

ABSTRACT

With the wide deployment of smart card automated fare collection (SCAFC) systems, public transit agencies have been benefiting from huge volume of transit data, a kind of sequential data, collected every day. Yet, improper publishing and use of transit data could jeopardize passengers' privacy. In this paper, we present our solution to transit data publication under the rigorous *differential privacy* model for the *Société de transport de Montréal* (STM). We propose an efficient *data-dependent* yet differentially private transit data sanitization approach based on a *hybrid-granularity* prefix tree structure. Moreover, as a post-processing step, we make use of the inherent consistency constraints of a prefix tree to conduct constrained inferences, which lead to better utility. Our solution not only applies to general sequential data, but also can be seamlessly extended to trajectory data. To our best knowledge, this is the first paper to introduce a practical solution for publishing large volume of sequential data under differential privacy. We examine data utility in terms of two popular data analysis tasks conducted at the STM, namely count queries and frequent sequential pattern mining. Extensive experiments on real-life STM datasets confirm that our approach maintains high utility and is scalable to large datasets.

Categories and Subject Descriptors

H.2.7 [**Database Administration**]: [Security, integrity, and protection]; H.2.8 [**Database Applications**]: [Data mining]

General Terms

Algorithms, Performance, Security

Keywords

Differential privacy, transit data, non-interactive release, data mining

KDD'12, August 12–16, 2012, Beijing, China.

1. INTRODUCTION

Over the last few years, smart card automated fare collection (SCAFC) systems have been increasingly deployed in transportation systems as a secure method of user validation and fare collection. These systems generate and collect passengers' transit data every day, which, after being anonymized, needs to be shared for various reasons, such as administrative regulations, profit sharing and data analysis. Transit data usually contains individual-specific sensitive information, and publishing raw data would directly violate passengers' privacy. In this paper, we study the real-life transit data publishing scenario at the *Société de transport de Montréal* (STM, http://www.stm.info), the public transport agency in Montréal area, and propose an efficient solution to publishing transit data under the rigorous *differential privacy* model [10] while satisfying the data utility requirements specified by the STM.

The STM deployed SCAFC systems in its transportation network in 2008. Transit information, such as smart card number and station ID, is collected when a passenger swipes his smart card at a SCAFC terminal, and is then stored in a central database management system, where the transit information of a passenger is organized as a sequence of stations in time order, a kind of *sequential data* (see a formal definition in Section 3). The deployment of SCAFC systems allows the seamless integration with other transit networks of neighboring cities, for example, the *Agence métropolitaine de transport* (AMT), which consequently requires data sharing among several collaborating parties. In addition, periodically, the IT department of the STM shares transit data with other departments, e.g., the marketing department, for basic data analysis, and publishes its transit data to external research institutions for more complex data analysis tasks, such as marketing analysis [28], customer behavior analysis [4], and demand forecasting [28].

According to its preliminary research [4], [20], [5], the STM can substantially benefit from transit data analysis at strategic, tactical, and operational levels. Yet, it has also realized that the nature of transit data is raising major privacy concerns on the part of card users in information sharing [20]. This fact has been an obstacle to conducting further data analysis much less performing regular commercial operations. In this paper, we aim to provide a practical solution to such a real-life transit data sharing scenario. We point out that our solution also benefits many other sectors, for example cell phone communication and credit card payment, which have been facing a similar dilemma in sequential data publishing and individual privacy protection.

Previous efforts have been made in addressing the problem of

transit data publication at the STM. Chen et al. [6] propose local suppression techniques based on the $(K, C)_L$-privacy model (a composition of k-anonymity [26] and confidence bounding [27]). However, recent works have shown that *partition-based* privacy models, for example k-anonymity and confidence bounding, are vulnerable to many types of privacy attacks, such as *composition attack* [14], *deFinetti attack* [18] and *foreground knowledge attack* [29], demonstrating their vulnerability to an adversary's background knowledge. Due to their deterministic nature, it is foreseeable that more types of privacy attacks could be discovered on these privacy models in the future. For this reason, we employ *differential privacy* [10], one of the strongest privacy models. Differential privacy provides provable privacy guarantees independent of an adversary's background knowledge and computational power (this claim may not be valid in some special cases [19], but is still correct for the STM case, as discussed in Section 4.3).

Traditional differentially private *non-interactive* approaches [3], [11], [31] are *data-independent* in the sense that all possible entries in the output domain need to be *explicitly* considered no matter what the underlying database is. For high-dimensional data, such as transit data, this is computationally infeasible. Consider a transit database $\mathcal{D}$ with all stations drawn from a universe of size m. Suppose the maximum length of a record (the number of stations in a record) in $\mathcal{D}$ is l. These approaches need to generate $\sum_{i=1}^{l} m^i = \frac{m^{l+1}-m}{m-1}$ output entries. For a STM transit database with $m = 1,000$ and $l = 20$, it requires to generate 10^{60} entries. Hence, these approaches are not computationally applicable with today's systems to real-life transit databases.

To tackle the challenge, we develop a *data-dependent* solution by extending the ideas proposed in two very recent papers [23], [7]. The general idea of a data-dependent solution is to adaptively narrow down the output domain by using *noisy* answers obtained from the underlying database. However, the methods in [23], [7] cannot be directly applied to sequential data for two reasons. First, the inherent sequentiality of sequential data is not considered in [23], [7]. Second, the methods only work for *sets*, yet a record in a sequential database may contain a *bag* of locations. Therefore, non-trivial efforts are needed to develop a differentially private data publishing approach for sequential data.

Protecting individual privacy is one aspect of sanitizing data. Another equally important aspect is preserving utility in sanitized data for data analysis. In this paper, we consider two important data mining tasks conducted at the STM, namely *count queries* (see a formal definition in Section 3.3) and *frequent sequential pattern mining* [2]. Count queries, as a general data analysis task, are the building block of many advanced data mining tasks. In the STM case, with accurate answers to count queries, data recipients can answer questions, such as "how many passengers have visited both stations *Guy-Concordia* and *McGill* [1]". Frequent sequential pattern mining, as a concrete data mining task, helps, for example, the STM better understand passengers' transit patterns and consequently allows the STM to adjust its network geometry and schedules in order to better utilize its existing resources.

Contribution. This is the first paper that introduces a practical solution for publishing large volume of real-life sequential data via differential privacy in the *non-interactive* setting. We summarize the major contributions of the paper as follows. First, we study the real-life transit data sharing scenario at the STM and propose an efficient sanitization algorithm to generate a differentially private sequential data release by making use of a *hybrid-granularity* prefix tree. We design a statistical process for efficiently constructing such a noisy prefix tree under Laplace mechanism, which is vital to the scalability of our solution. We emphasize that our approach can be seamlessly extended to trajectory data (see Section 4.4). Second, we make use of two sets of inherent constraints of a prefix tree to conduct constrained inferences, which helps generate a more accurate release. Third, we conduct an extensive experimental study over different real-life STM datasets. We examine utility of sanitized data for two different data mining tasks performed by the STM, namely count queries (a generic data analysis task) and frequent sequential pattern mining (a concrete data mining task). Experimental results demonstrate that our approach maintains high utility and is scalable to large volume of real-life sequential data.

[1] *Guy-Concordia* and *McGill* are two metro stations on the green line of the STM metro network.

2. RELATED WORK

More broadly, sequential data can be considered as a special kind of trajectory data. There have been some recent works [1], [27], [32], [17], [13], [6], [24] on privacy-preserving sequential (trajectory) data publishing based on partition-based privacy models. Abul et al. [1] propose the (k, δ)-anonymity model based on the inherent imprecision of sampling and positioning systems, where δ represents possible location imprecision. Terrovitis and Mamoulis [27] model an adversary's background knowledge as a set of projections of sequences in a sequential database, and consequently propose a data suppression technique that limits the confidence of inferring the presence of a location in a sequence to a pre-defined probability threshold. Yarovoy et al. [32] propose to k-anonymize a moving object database (MOD) by considering timestamps as the quasi-identifiers (QIDs). Adversaries are assumed to launch privacy attacks based on *attack graphs*. Monreale et al. [24] present an approach based on spatial generalization in order to achieve k-anonymity. The novelty of their approach lies in a generalization scheme that depends on the underlying trajectory dataset rather than a fixed grid hierarchy.

Hu et al. [17] present the problem of k-anonymizing a trajectory database with respect to a sensitive event database. The goal is to make sure that every event is shared by at least k users. Specifically, they develop a new generalization mechanism known as *local enlargement*. Fung et al. [13] propose the $(K, C)_L$-privacy model that thwarts both identity linkages on trajectory data and attribute linkages via trajectory data. Based on the $(K, C)_L$-privacy model, Chen et al. [6] develop a generic solution for various data utility metrics by use of *local suppression*. Compared to all these approaches [1], [27], [32], [17], [13], [6], [24], the major contribution of our paper is the use of differential privacy, which provides significantly stronger privacy guarantees.

In the last few years, differential privacy has emerged as the de facto successor to partition-based privacy models. Currently most of the research on differential privacy concentrates on the *interactive setting*. Dwork [9] provides an overview of recent works on differential privacy in the interactive setting.

The works closest to ours are by Blum et al. [3], Dwork et al. [11], Xiao et al. [31], Mohammed et al. [23], and Chen et al. [7]. All these works consider *non-interactive* data publishing under differential privacy. Blum et al. [3] demonstrate that it is possible to release *synthetic* private databases that are useful for all queries over a discretized domain from a concept class with polynomial Vapnik-Chervonenkis dimension. However, their mechanism is not efficient, taking runtime complexity of $superpoly(|\mathcal{C}|, |I|)$, where $|\mathcal{C}|$ is the size of a concept class and $|I|$ the size of the universe. Dwork et al. [11] propose a recursive algorithm of generating a *synthetic* database with runtime complexity of $poly(|\mathcal{C}|, |I|)$. This improvement, however, is still insufficient to handle real-life sequential

Table 1: Sample sequential database

Rec. #	Path
1	$L_1 \rightarrow L_2 \rightarrow L_3$
2	$L_1 \rightarrow L_2$
3	$L_3 \rightarrow L_2 \rightarrow L_1$
4	$L_1 \rightarrow L_2 \rightarrow L_4$
5	$L_1 \rightarrow L_2 \rightarrow L_3$
6	$L_3 \rightarrow L_2$
7	$L_1 \rightarrow L_2 \rightarrow L_4 \rightarrow L_1$
8	$L_3 \rightarrow L_1$

datasets due to the exponential size of $|\mathcal{C}|$. Xiao et al. [31] propose a wavelet-transformation based approach for *relational data* to lower the magnitude of noise, rather than adding independent Laplace noise.

Two very recent papers [23], [7] point out that data-dependent approaches are more efficient and more effective for generating a differentially private release. Mohammed et al. [23] propose a generalization-based sanitization algorithm for *relational data* with the goal of classification analysis. Chen et al. [7] propose a probabilistic top-down partitioning algorithm for *set-valued data*. Due to the reasons mentioned in Section 1, they cannot be directly applied to sequential data.

3. PRELIMINARIES

Let $\mathcal{L} = \{L_1, L_2, \cdots, L_{|\mathcal{L}|}\}$ be the universe of locations, where $|\mathcal{L}|$ is the size of the universe. Without loss of generality, we consider locations as discrete spatial areas in a map. For STM transit data, $\mathcal{L}$ represents all stations in the STM transportation network. Each record in a sequential database consists of a *sequence* of time-ordered locations drawn from this universe. Formally, a sequence S of length $|S|$ is an ordered list of locations $S = L_1 \rightarrow L_2 \rightarrow \cdots \rightarrow L_{|S|}$, where $\forall 1 \leq i \leq |S|$, $L_i \in \mathcal{L}$. A location may occur multiple times in S, and may occur consecutively in S.

A sequential database $\mathcal{D}$ of size $|\mathcal{D}|$ is composed of a multiset of sequences $\mathcal{D} = \{S_1, S_2, \cdots, S_{|\mathcal{D}|}\}$. Each sequence represents the movement history of a record owner. Table 1 presents a sample sequential database with $\mathcal{L} = \{L_1, L_2, L_3, L_4\}$.

3.1 Prefix Tree

A sequential database can be represented in a more compact way in terms of a *prefix tree*. A prefix tree groups sequences with the same prefix into the same branch. A sequence $S' = L'_1 \rightarrow L'_2 \rightarrow \cdots \rightarrow L'_{|S'|}$ is a prefix of a sequence $S = L_1 \rightarrow L_2 \rightarrow \cdots \rightarrow L_{|S|}$, denoted by $S' \preceq S$, if and only if $|S'| \leq |S|$ and $\forall 1 \leq i \leq |S'|$, $L'_i = L_i$. For example, $L_1 \rightarrow L_2$ is a prefix of $L_1 \rightarrow L_2 \rightarrow L_4 \rightarrow L_3$, but $L_1 \rightarrow L_4$ is *not*. We formally define a prefix tree below.

DEFINITION 3.1 (PREFIX TREE). A prefix tree $\mathcal{PT}$ of a sequential database $\mathcal{D}$ is a triplet $\mathcal{PT} = (V, E, Root)$, where V is the set of nodes labeled with locations, each corresponding to a unique prefix in $\mathcal{D}$; E is the set of edges, representing transitions between nodes; $Root \in V$ is the virtual root of $\mathcal{PT}$. The unique prefix represented by a node $v \in V$, denoted by $prefix(v, \mathcal{PT})$, is a sequence of locations starting from $Root$ to v. ■

Each node $v \in V$ keeps a doublet in the form of $\langle tr(v), c(v)\rangle$, where $tr(v)$ is the set of sequences in $\mathcal{D}$ having $prefix(v, \mathcal{PT})$, that is, $\{S \in \mathcal{D} : prefix(v, \mathcal{PT}) \preceq S\}$, and $c(v)$ is a noisy version of $|tr(v)|$ (e.g., $|tr(v)|$ plus Laplace noise). $tr(Root)$ contains all sequences in $\mathcal{D}$. We call the set of all nodes of $\mathcal{PT}$ at a given depth i a *level* of $\mathcal{PT}$, denoted by $level(i, \mathcal{PT})$. $Root$ is at depth zero. When $\mathcal{PT}$ is clear in the context, it is omitted in the notation. Figure 1 illustrates the prefix tree of the sample database in Table 1, where each node v is labeled with its location and $|tr(v)|$.

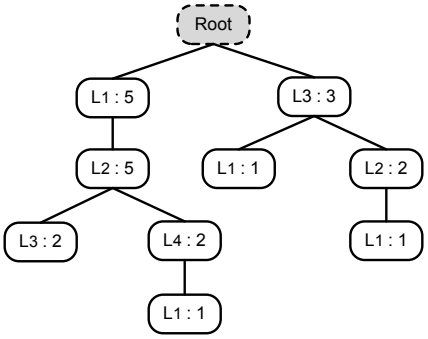


Figure 1: The prefix tree of the sample data

3.2 Differential Privacy

Differential privacy, in general, requires that the removal or addition of a single database record does not significantly affect the outcome of any analysis based on the database. Therefore, for a record owner, any privacy breach will not be a result of participating in the database since anything that can be learned from the database with his record can also be learned from the one without his record. We formally define differential privacy in the *non-interactive* setting [3] as follow.

DEFINITION 3.2 (DIFFERENTIAL PRIVACY). A privacy mechanism $\mathcal{A}$ gives ϵ-differential privacy if for any database $\mathcal{D}_1$ and $\mathcal{D}_2$ differing on at most one record, and for any possible sanitized database $\widetilde{\mathcal{D}} \in Range(\mathcal{A})$,

$$Pr[\mathcal{A}(\mathcal{D}_1) = \widetilde{\mathcal{D}}] \leq e^{\epsilon} \times Pr[\mathcal{A}(\mathcal{D}_2) = \widetilde{\mathcal{D}}] \quad (1)$$

where the probability is taken over the randomness of $\mathcal{A}$. ■

Two principal techniques for achieving differential privacy are *Laplace mechanism* [10] and *exponential mechanism* [22]. A fundamental concept of both techniques is the *global sensitivity* of a function [10] mapping underlying databases to (vectors of) reals.

DEFINITION 3.3 (GLOBAL SENSITIVITY). For any function $f : \mathcal{D} \rightarrow \mathbb{R}^d$, the sensitivity of f is

$$\Delta f = \max_{\mathcal{D}_1, \mathcal{D}_2} ||f(\mathcal{D}_1) - f(\mathcal{D}_2)||_1 \quad (2)$$

for all $\mathcal{D}_1, \mathcal{D}_2$ differing in at most one record. ■

Laplace Mechanism. For the analysis whose outputs are real, a standard mechanism to achieve differential privacy is to add Laplace noise to the true output of a function. Dwork et al. [10] propose the Laplace mechanism which takes as inputs a database $\mathcal{D}$, a function f, and the privacy parameter ϵ. The noise is generated according to a Laplace distribution with the probability density function (pdf) $p(x|\lambda) = \frac{1}{2\lambda} e^{-|x|/\lambda}$, where λ is determined by both Δf and the desired privacy parameter ϵ.

THEOREM 3.1. *For any function $f : \mathcal{D} \rightarrow \mathbb{R}^d$, the mechanism $\mathcal{A}$*

$$\mathcal{A}(\mathcal{D}) = f(\mathcal{D}) + Laplace(\Delta f/\epsilon) \quad (3)$$

gives ϵ-differential privacy. ■

Exponential Mechanism. For the analysis whose outputs are not real or make no sense after adding noise, McSherry and Talwar [22] propose the exponential mechanism that selects an output from the output domain, $r \in \mathcal{R}$, by taking into consideration its score of a given utility function q in a differentially private manner. The exponential mechanism assigns exponentially greater probabilities of being selected to outputs of higher scores so that the final output would be close to the optimum with respect to q. The chosen utility function q should be insensitive to changes of any particular record, that is, has a low sensitivity. Let the sensitivity of q be $\Delta q = \max_{\forall r, \mathcal{D}_1, \mathcal{D}_2} |q(\mathcal{D}_1, r) - q(\mathcal{D}_2, r)|$.

THEOREM 3.2. *Given a utility function $q : (\mathcal{D} \times \mathcal{R}) \to \mathbb{R}$ for a database $\mathcal{D}$, the mechanism $\mathcal{A}$,*

$$\mathcal{A}(\mathcal{D}, q) = \left\{ return\ r\ with\ probability \propto exp(\frac{\epsilon q(\mathcal{D}, r)}{2\Delta q}) \right\} \quad (4)$$

gives ϵ-differential privacy. ■

3.3 Utility Requirements

In the STM case, sanitized data is mainly used to perform two data mining tasks, namely *count query* and *frequent sequential pattern mining* [2]. Count queries, as a general data analysis task, are the building block of many data mining tasks.

DEFINITION 3.4 (COUNT QUERY). For a given set of locations $\mathbb{L}$ drawn from the universe $\mathcal{L}$, a count query Q over a database $\mathcal{D}$ is defined to be $Q(\mathcal{D}) = |\{S \in \mathcal{D} : \mathbb{L} \subseteq ls(S)\}|$, where $ls(S)$ returns the set of locations in S. ■

Note that sequentiality among locations is not considered in count queries, because the major users of count queries are, for example, the personnel of the marketing department of the STM, who are merely interested in users' presence in certain stations for marketing analysis, known as *passenger counting*, but *not* the sequentiality of visiting. Instead, the preservation of sequentiality in sanitized data is examined by frequent sequential pattern mining. We measure the utility of a count query Q over the sanitized database $\widetilde{\mathcal{D}}$ by its *relative error* [31], [30], [7] with respect to the true result over the original database $\mathcal{D}$, which is computed as:

$$error(Q(\widetilde{\mathcal{D}})) = \frac{|Q(\widetilde{\mathcal{D}}) - Q(\mathcal{D})|}{max\{Q(\mathcal{D}), s\}},$$

where s is a *sanity bound* used to mitigate the influences of queries with extremely small *selectivities* [31], [30], [7].

For frequent sequential pattern mining, we measure the utility of sanitized data in terms of *true positive* (TP), *false positive* (FP) and *false drop* (FD) [12]. Given a positive number k, we denote the set of top k most frequent sequential patterns with size greater than 1 [2] on the original database $\mathcal{D}$ by $\mathcal{F}_k(\mathcal{D})$ and the set of frequent sequential patterns on the sanitized database $\widetilde{\mathcal{D}}$ by $\mathcal{F}_k(\widetilde{\mathcal{D}})$. True positive is the number of frequent sequential patterns in $\mathcal{F}_k(\mathcal{D})$ that are correctly identified in $\mathcal{F}_k(\widetilde{\mathcal{D}})$, that is, $|\mathcal{F}_k(\mathcal{D}) \cap \mathcal{F}_k(\widetilde{\mathcal{D}})|$. False positive is the number of infrequent sequential patterns in $\mathcal{D}$ that are mistakenly included in $\mathcal{F}_k(\widetilde{\mathcal{D}})$, that is, $|\mathcal{F}_k(\widetilde{\mathcal{D}}) - \mathcal{F}_k(\mathcal{D}) \cap \mathcal{F}_k(\widetilde{\mathcal{D}})|$. False drop is the number of frequent sequential patterns in $\mathcal{F}_k(\mathcal{D})$ that are wrongly omitted in $\mathcal{F}_k(\widetilde{\mathcal{D}})$, that is, $|\mathcal{F}_k(\mathcal{D}) \cup \mathcal{F}_k(\widetilde{\mathcal{D}}) - \mathcal{F}_k(\widetilde{\mathcal{D}})|$. Since in our setting $|\mathcal{F}_k(\mathcal{D})| = |\mathcal{F}_k(\widetilde{\mathcal{D}})| = k$, false positives always equal false drops.

[2]Nearly every single location forms a size-1 frequent sequential pattern.

Algorithm 1 Sequential Data Sanitization Algorithm

Input: Raw sequential dataset $\mathcal{D}$
Input: Privacy budget ϵ
Input: Height of the prefix tree h
Input: Location taxonomy tree $\mathcal{T}$
Output: Sanitized dataset $\widetilde{\mathcal{D}}$
1: Noisy prefix tree $\mathcal{PT} \leftarrow BuildNoisyPrefixTree(\mathcal{D}, \epsilon, h, \mathcal{T})$;
2: Sanitized dataset $\widetilde{\mathcal{D}} \leftarrow GeneratePrivateRelease(\mathcal{PT})$;
3: **return** $\widetilde{\mathcal{D}}$;

4. SANITIZATION ALGORITHM

We first provide an overview of our two-step sanitization algorithm in Algorithm 1. Given a raw sequential dataset $\mathcal{D}$, a privacy budget ϵ, a user specified height of the prefix tree h and a location taxonomy tree $\mathcal{T}$, it returns a sanitized dataset $\widetilde{\mathcal{D}}$ satisfying ϵ-differential privacy. *BuildNoisyPrefixTree* constructs a *noisy hybrid-granularity* prefix tree $\mathcal{PT}$ of $\mathcal{D}$ using a set of count queries based on the given taxonomy tree $\mathcal{T}$, which defines multiple levels of granularities over the location universe. It can be either public knowledge or generated from the location universe on-the-fly by specifying a fan-out value. In the STM case, we use a two-level taxonomy tree where each station can be generalized to the metro/bus line on which it locates. For the simplicity of illustration, we give our algorithm based on a two-level taxonomy tree. The extension to a multiple-level taxonomy tree is straightforward. *GeneratePrivateRelease* employs utility boosting techniques on $\mathcal{PT}$ based on two sets of consistency constraints, and then generates a differentially private release.

4.1 Noisy Prefix Tree Construction

Our strategy for *BuildNoisyPrefixTree* is to recursively group sequences in $\mathcal{D}$ into *disjoint* sub-datasets based on their prefixes. Procedure 1 presents the details of *BuildNoisyPrefixTree*. We first create a prefix tree $\mathcal{PT}$ with a virtual root $Root$ (Lines 2-3). To build $\mathcal{PT}$, we employ a *uniform* privacy budget allocation scheme, that is, divide the total privacy budget ϵ into equal portions $\bar{\epsilon} = \frac{\epsilon}{h}$, each is used for constructing a level of $\mathcal{PT}$ (Line 4). In Lines 6-20, we iteratively construct each level of $\mathcal{PT}$ in a noisy way.

To satisfy differential privacy, we need to guarantee that every sequence that can be derived from the location universe (either in or not in $\mathcal{D}$) has a non-zero probability to appear in the noisy prefix tree. Therefore, at each level, for each node, we need to consider *every* possible location as its potential child. Our goal is to identify the children that are associated with non-zero number of sequences (referred to as *non-empty node*) so that we can continue to expand them. Here decisions have to be made based on noisy counts.

In order to achieve good utility, it is critical to prune out nodes associated with zero number of sequences (*empty node*) reliably as early as possible. For this reason, instead of using a simple prefix tree, we divide a level of $\mathcal{PT}$ into two sub-levels with different location granularities. The first sub-level consists of nodes associated with *generalized* location information (*generalized node*), and then, depending on noisy counts of these nodes, we decide whether to further expand them to create the second sub-level in which nodes are associated with non-generalized locations (e.g., ask the noisy count of passengers in a metro line and then decide whether to ask the count of each station on this line). $\bar{\epsilon}$ is then allocated to the two sub-levels as a function of the fan-out f of the location taxonomy tree $\mathcal{T}$: the first sub-level receives $\bar{\epsilon}_1 = \frac{2\bar{\epsilon}}{f}$ and the second receives $\bar{\epsilon}_2 = \frac{(f-2)\bar{\epsilon}}{f}$. One important observation is that all nodes on the same sub-level are associated with *disjoint* sequence subsets, and

Procedure 1 $BuildNoisyPrefixTree$ Procedure

Input: Raw sequential dataset $\mathcal{D}$
Input: Privacy budget ϵ
Input: Height of the prefix tree h
Input: Location taxonomy tree $\mathcal{T}$
Output: Noisy prefix tree $\mathcal{PT}$

$i = 0$;
Create a prefix tree $\mathcal{PT}$ with a virtual root $Root$;
Add all sequences in $\mathcal{D}$ to $tr(Root)$;
$\bar{\epsilon} = \frac{\epsilon}{h}$;
Calculate $\bar{\epsilon}_1$ and $\bar{\epsilon}_2$ s.t. $\bar{\epsilon}_1 + \bar{\epsilon}_2 = \bar{\epsilon}$;
while $i < h$ **do**
  for each non-generalized node $v \in level(i)$ **do**
    $\mathcal{U}_g \leftarrow$ the set of generalized nodes from $\mathcal{T}$;
    for each node $u \in \mathcal{U}_g$ **do**
      Add sequences S with $prefix(u) \preceq S$ to $tr(u)$;
      $c(u) = NoisyCount(|tr(u)|, \bar{\epsilon}_1)$;
      if $c(u) \geq \theta_g$ **then**
        Add u to $\mathcal{PT}$;
        $\mathcal{U}_{ng} \leftarrow u$'s non-generalized children in $\mathcal{T}$;
        for each node $w \in \mathcal{U}_{ng}$ **do**
          Add sequences S with $prefix(w) \preceq S$ to $tr(w)$;
          $c(w) = NoisyCount(|tr(w)|, \bar{\epsilon}_2)$;
          if $c(w) \geq \theta_{ng}$ **then**
            Add w to $\mathcal{PT}$;
  $i++$;
return $\mathcal{PT}$;

therefore the privacy budget allocated to a sub-level can be used *in full* for each node in it. We provide formal analysis on utility improvement of a hybrid-granularity prefix tree after presenting Theorem 4.1.

For a dataset with a very large location universe $\mathcal{L}$, processing all locations explicitly may be slow. We provide an efficient implementation by *separately* handling potential non-empty and empty nodes. For a non-empty node u, we add Laplace noise to $|tr(u)|$ and use the noisy answer $c(u)$ to decide if it is non-empty. If $c(u)$ is greater than or equal to the pre-defined threshold θ, we deem that u is non-empty and insert u to $\mathcal{PT}$. In the STM case, the threshold of a non-generalized node $\theta_{ng} = \frac{2\sqrt{2}}{\bar{\epsilon}_2}$ (two times of the standard deviation of noise) while the threshold of a generalized node $\theta_g = \frac{4\sqrt{2}}{\bar{\epsilon}_1}$. Intuitively, this setting more reliably eliminates empty nodes while having very limited effect on non-empty nodes. Since non-empty nodes are typically of a small number, this process can be done efficiently.

For empty nodes, we need to conduct a series of *independent* boolean tests, each calculates $NoisyCount(0, \bar{\epsilon}')$ to check if it passes θ, where $\bar{\epsilon}'$ is the privacy budget assigned to a node (either $\bar{\epsilon}_1$ or $\bar{\epsilon}_2$). The number of empty nodes that pass θ, k, follows the *binomial distribution* $B(m, p_\theta)$, where m is the total number of empty nodes we need to check and p_θ is the probability for a single experiment to succeed. We design a statistical process for Laplace mechanism to directly extract k empty nodes without explicitly processing every empty node. This is inspired by [8], in which a statistical process is designed for *geometric mechanism* [15].

THEOREM 4.1. *Independently conducting m pass/not pass experiments based on Laplace mechanism with privacy budget $\bar{\epsilon}'$ and threshold θ is equivalent to the following steps: 1) generate a value k from the binomial distribution $B(m, p_\theta)$, where $p_\theta = \frac{exp(-\bar{\epsilon}'\theta)}{2}$; 2) select k uniformly random empty nodes without replacement with noisy counts sampled from the distribution*

$$P(x) = \begin{cases} 0 & \forall x < \theta \\ 1 - exp(\bar{\epsilon}'\theta - \bar{\epsilon}'x) & \forall x \geq \theta \end{cases}$$ ■

Proof. The probability of a single experiment passing the threshold θ is

$$Pr[PASS] = \int_\theta^\infty \frac{\bar{\epsilon}'}{2} exp(-\bar{\epsilon}'x)dx = \frac{exp(-\bar{\epsilon}'\theta)}{2}.$$

Since the experiments are independent, the number of successful experiments, k, follows the binomial distribution $B(m, \frac{exp(-\bar{\epsilon}'\theta)}{2})$. Once k is determined, we can uniformly at random select k empty nodes. The probability density function of the noisy counts x for the k empty nodes, conditional on $x \geq \theta$, is:

$$p(x|x \geq \theta) = \begin{cases} 0 & \forall x < \theta \\ \frac{\frac{\bar{\epsilon}'}{2} exp(-\bar{\epsilon}'x)}{p_\theta} = \bar{\epsilon}' exp(\bar{\epsilon}'\theta - \bar{\epsilon}'x) & \forall x \geq \theta \end{cases}$$

The corresponding cumulative distribution function is:

$$P(x) = \begin{cases} 0 & \forall x < \theta \\ \int_\theta^x \bar{\epsilon}' exp(\bar{\epsilon}'\theta - \bar{\epsilon}'x)dx = 1 - exp(\bar{\epsilon}'\theta - \bar{\epsilon}'x) & \forall x \geq \theta \end{cases}$$ ■

Now we give a theoretical analysis on the utility improvement due to a hybrid-granularity prefix tree in terms of reduction of number of empty non-generalized nodes that are mistakenly generated. This number directly reflects the level of noise in sanitized data.

THEOREM 4.2. *For an empty node v at level i, a noisy hybrid-granularity prefix tree of height h reduces the number of empty non-generalized nodes mistakenly generated due to identifying v as non-empty by a factor of $O(2^{h-i}exp(4\sqrt{2}(h-i)))$.* ■

Proof. From Theorem 4.1, we learn that $p_{\theta_{ng}} = \frac{exp(-2\sqrt{2})}{2}$ and $p_{\theta_g} = \frac{exp(-4\sqrt{2})}{2}$. Consider an empty node v at level i. In a simple noisy prefix tree, if v is mistakenly considered as non-empty, the expected value of number of descendants of v, which are all empty nodes, is $\mathbb{E}_1 = (|\mathcal{L}|p_{\theta_{ng}})^{h-i}$. In a hybrid-granularity prefix tree, the expected value of number of descendants of v is

$$\mathbb{E}_2 = (fp_{\theta_{ng}})(\frac{|\mathcal{L}|}{f}p_{\theta_g} \cdot fp_{\theta_{ng}})^{h-i} = (fp_{\theta_{ng}})(|\mathcal{L}|p_{\theta_g}p_{\theta_{ng}})^{h-i}.$$

where f is the fan-out of the location taxonomy tree. This implies a reduction of

$$\frac{\mathbb{E}_1}{\mathbb{E}_2} = \frac{1}{fp_{\theta_{ng}}p_{\theta_g}^{h-i}} = O(2^{h-i}exp(4\sqrt{2}(h-i))).$$ ■

EXAMPLE 4.1. Consider the sequential database $\mathcal{D}$ in Table 1, the height $h = 2$, and the calculated threshold $\theta = 3$. Suppose that L_1 and L_2 can be generalized to $L_{\{1,2\}}$ and L_3 and L_4 can be generalized to $L_{\{3,4\}}$. The construction of a possible noisy hybrid-granularity prefix tree $\mathcal{PT}$ is illustrated in Figure 2. A path of $\mathcal{PT}$ may be of a length shorter than h if it has been considered "empty" before h is reached. ■

4.2 Private Release Generation

We can generate the sanitized database by traversing $\mathcal{PT}$ once in *postorder* (ignore generalized nodes), calculating the number n of sequences terminated at each non-generalized node v and appending n copies of $prefix(v, \mathcal{PT})$ to the output. However, due to the noise added to ensure differential privacy, we may not be able to obtain a meaningful and consistent release. If we leave such inconsistencies unsolved, the resulting release may not be meaningful and therefore provides poor utility.

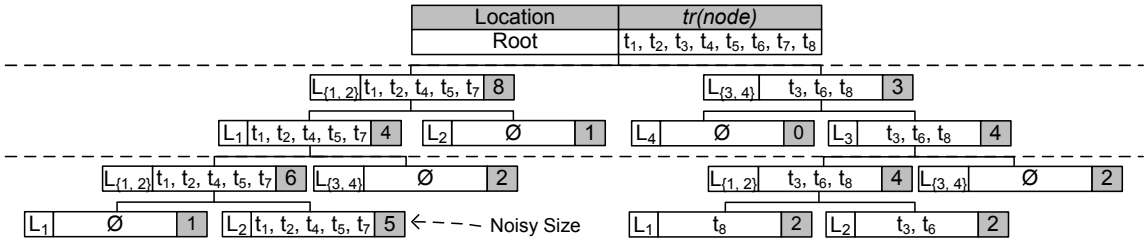


Figure 2: The noisy hybrid-granularity prefix tree of the sample data

DEFINITION 4.1 (CONSISTENCY CONSTRAINT). In a prefix tree, there exist two sets of consistency constraints:

1. For any root-to-leaf path p, $\forall v_i \in p, |tr(v_i)| \leq |tr(v_{i+1})|$, where v_i is a child of v_{i+1};
2. For each node v, $|tr(v)| \geq \sum_{u \in children(v)} |tr(u)|$. ■

Our goal is to enforce such consistency constraints on the noisy prefix tree (with all generalized nodes removed [3]) in order to produce a consistent and more accurate private release. We adapt the constrained inference technique proposed in [16] to adjust the noisy counts of nodes in the noisy prefix tree so that the constraints defined in Definition 4.1 are respected. Note that the technique proposed in [16] cannot be directly applied to our case because: 1) the noisy prefix tree has an irregular structure (rather than a complete tree with a fixed degree); 2) the noisy prefix tree has different constraints $|tr(v)| \geq \sum_{u \in children(v)} |tr(u)|$ (rather than $|tr(v)| = \sum_{u \in children(v)} |tr(u)|$). Consequently, we propose a two-phase procedure to obtain a *consistent* estimate with respect to Definition 4.1 for each node (except the virtual root) in the noisy prefix tree $\mathcal{PT}$.

We first generate an intermediate estimate for the noisy count of each node v (except the virtual root) in $\mathcal{PT}$. Consider a root-to-leaf path p of $\mathcal{PT}$. Let us organize the noisy counts of nodes $v_i \in p$ into a sequence $\mathcal{S} = \langle c(v_1), c(v_2), \cdots, c(v_{|p|})\rangle$, where v_i is a child of v_{i+1}. Let $mean[i, j]$ denote the mean of a subsequence of $\mathcal{S}$, $\langle c(v_i), c(v_{i+1}), ..., c(v_j)\rangle$, that is, $mean[i, j] = \frac{\sum_{m=i}^{j} c(v_m)}{j-i+1}$. We compute the intermediate estimate $\widetilde{\mathcal{S}}$ by Theorem 4.3 [16].

THEOREM 4.3. *Let* $I_m = min_{j \in [m,|p|]} max_{i \in [1,j]} mean[i, j]$ *and* $U_m = max_{i \in [1,m]} min_{j \in [i,|p|]} mean[i, j]$. *The minimum* L_2 *solution* $\widetilde{\mathcal{S}} = \langle I_1, I_2, ..., I_{|p|}\rangle = \langle U_1, U_2, ..., U_{|p|}\rangle$. ■

The result of Theorem 4.3 satisfies the first type of constraints in Definition 4.1. However, a node v in $\mathcal{PT}$ appears in $|leaves(v)|$ root-to-leaf paths, where $leaves(v)$ denotes the leaves of the subtree of $\mathcal{PT}$ rooted at v, and therefore, has $|leaves(v)|$ intermediate estimates, each being an independent observation of the true count $|tr(v)|$. We compute the *consolidated* intermediate estimate of v as the mean of the estimates, normally the best estimate for $|tr(v)|$. We denote the consolidated intermediate estimate of v by $\widetilde{c}(v)$.

After obtaining $\widetilde{c}(v)$ for each node v, we compute its consistent estimate $\bar{c}(v)$ in a top-down fashion as follows:

$$\bar{c}(v) = \begin{cases} \widetilde{c}(v) & if v \in level(1, \mathcal{PT}) \\ \widetilde{c}(v) + min(0, \frac{\bar{c}(w) - \sum_{u \in children(w)} \widetilde{c}(u)}{|children(w)|}) & otherwise \end{cases}$$

where w is the parent of v. It follows the intuition that the observation $\sum_{u \in children(w)} \widetilde{c}(u) > \bar{c}(w)$ is strong evidence that excessive noise is added to the children. Since the variance of noise in $\bar{c}(w)$ is approximately $|children(w)|$ times smaller than $\sum_{u \in children(w)} \widetilde{c}(u)$, it is reasonable to decrease the children's counts according to $\bar{c}(w)$. However, we never increase the children's counts based on $\bar{c}(w)$ because a large $\bar{c}(w)$ simply indicates that many sequences actually terminate at w. It is easy to see that the consistency constraints in Definition 4.1 are respected among consistent estimates, and therefore the proof is omitted here.

[3]The utility improvements of constrained inferences on a prefix tree with and without generalized nodes are almost identical.

4.3 Analysis

Privacy Analysis. Kifer and Machanavajjhala [19] point out that differential privacy must be applied with caution. The privacy protection provided by differential privacy relates to the data generating mechanism and deterministic aggregate-level background knowledge. In the STM case, transit data is independent of each other and no deterministic statistics of the raw database will ever be released. Hence differential privacy is appropriate for our problem. We now show that Algorithm 1 satisfies ϵ-differential privacy.

THEOREM 4.4. *Given the total privacy budget* ϵ*, Algorithm 1 ensures* ϵ*-differential privacy.* ■

Proof. Algorithm 1 consists of two steps, namely *BuildNoisyPrefixTree* and *GeneratePrivateRelease*. In the procedure *BuildNoisyPrefixTree*, our approach appeals to the well-understood query model to construct the noisy prefix tree $\mathcal{PT}$. Consider a level of $\mathcal{PT}$, which is composed of two sub-levels. Since all nodes on the same sub-level contain *disjoint* sets of sequences. According to the *parallel composition* (Theorem 4.5 [21]), the entire privacy budget needed for a sub-level is bounded by the worst case, that is, $\frac{2\bar{\epsilon}}{f}$ for the first sub-level and $\frac{(f-2)\bar{\epsilon}}{f}$ for the second sub-level.

THEOREM 4.5. *Let* $\mathcal{A}_i$ *each provide* ϵ_i*-differential privacy. A sequence of* $\mathcal{A}_i(\mathcal{D}_i)$ *over a set of disjoint datasets* $\mathcal{D}_i$ *provides* $max(\epsilon_i)$*-differential privacy.* ■

The use of privacy budget on different sub-levels follows *sequential composition* (Theorem 4.6 [21]).

THEOREM 4.6. *Let* $\mathcal{A}_i$ *each provide* ϵ_i*-differential privacy. A sequence of* $\mathcal{A}_i(\mathcal{D})$ *over the database* $\mathcal{D}$ *provides* $(\sum_i \epsilon_i)$*-differential privacy.* ■

Since there are at most h levels, the total privacy budget needed to build the noisy prefix tree $\leq h \times (\frac{2\bar{\epsilon}}{f} + \frac{(f-2)\bar{\epsilon}}{f}) = \epsilon$.

For the procedure *GeneratePrivateRelease*, we make use of the inherent constraints of a prefix tree to boost utility. The procedure only accesses a differentially private noisy prefix tree, not the underlying database. As proven by Hay et al. [16], a post-processing of differentially private results remains differentially private. Therefore, Algorithm 1 as a whole maintains ϵ-differential privacy. ■

Complexity Analysis. Algorithm 1 is of runtime complexity $O(|\mathcal{D}| \cdot |\mathcal{L}|)$, where $|\mathcal{D}|$ is the number of sequences in the input database $\mathcal{D}$ and $|\mathcal{L}|$ is the size of the location universe. For *BuildNoisyPrefixTree*, the major computational cost is node generation and sequence distribution. For each level of the noisy prefix tree, the number of

Table 2: Experimental dataset statistics.

Datasets	$\|\mathcal{D}\|$	$\|\mathcal{L}\|$	$max\|S\|$	$avg\|S\|$
Metro	847,668	68	90	4.21
Bus	778,724	944	121	5.67

nodes to generate approximates $k(1 + \frac{1}{f})|\mathcal{D}|$, where $k \ll |\mathcal{L}|$ is a number depending on $|\mathcal{L}|$. For each level, we need to distribute *at most* $2|\mathcal{D}|$ sequences to the newly generated nodes. Hence, the complexity of constructing a single level is $O(|\mathcal{D}| \cdot |\mathcal{L}|)$. Therefore, the total runtime complexity of *BuildNoisyPrefixTree* for constructing a noisy prefix tree of height h is $O(h|\mathcal{D}| \cdot |\mathcal{L}|)$. In *GeneratePrivateRelease*, the complexity of calculating the intermediate estimates for a single root-to-leaf path is $O(h)$. Since there can be at most $|\mathcal{D}|$ distinct root-to-leaf paths, the complexity of computing all intermediate estimates is $O(h|\mathcal{D}|)$. To compute consistent estimates, we need to visit every node exactly twice, which is of complexity $O(|\mathcal{D}| \cdot |\mathcal{L}|)$. Similarly, the computational cost of generating the private release by traversing the noisy prefix tree once in postorder is $O(|\mathcal{D}| \cdot |\mathcal{L}|)$. Since h is a very small constant compared to $|\mathcal{D}|$ and $|\mathcal{L}|$, the total complexity of Algorithm 1 is $O(|\mathcal{D}| \cdot |\mathcal{L}|)$.

4.4 Extensions

Our solution can be seamlessly applied to trajectory data. A trajectory is composed of a sequence of *location-timestamp* pairs in the form of $loc_1t_1 \rightarrow loc_2t_2 \rightarrow \cdots \rightarrow loc_nt_n$, where $t_1 \leq t_2 \leq \cdots \leq t_n$. The time factor is often discretized into intervals at different levels of granularity, e.g., hour, which is typically determined by the data publisher. All timestamps of a trajectory database form a timestamp universe.

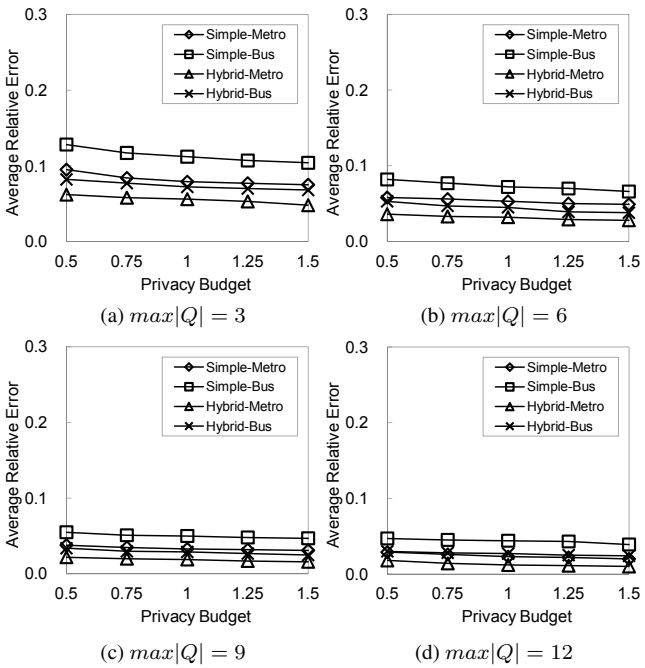
In this case, we can label each node in the prefix tree by both a location and a timestamp. Therefore, two trajectories with the same sequence of locations but different timestamps are considered different. Consequently, when constructing the noisy prefix tree, in order to expand a node loc_it_i, we have to consider the combinations of all locations and the timestamps in the time universe that are greater than t_i (because the timestamps in a trajectory are non-decreasing), resulting in a larger candidate set. Due to the efficient implementation we propose in Section 4.1, the computational cost will remain moderate.

5. EXPERIMENTAL EVALUATION

In this section, we examine the utility of sanitized data in terms of count queries and frequent sequential pattern mining, and evaluate the scalability of our approach for processing large-scale real-life data. In particular, we compare the utility improvements of the method using a hybrid-granularity prefix tree (referred to as *Hybrid*) over the method using a simple prefix tree (referred to as *Simple*). Our implementation was done in C++, and all experiments were performed on an Intel Core 2 Duo 2.26GHz PC with 2GB RAM. Extensive experiments were conducted on two real-life STM transit datasets, *Metro* and *Bus*, which record the transit history of passengers in the STM metro and bus networks, respectively. The characteristics of the datasets are summarized in Table 2, where $max|S|$ is the maximum sequence length and $avg|S|$ the average length.

5.1 Utility

Count Query. In our first set of experiments, we examine relative errors of count queries with respect to two different parameters, namely the privacy budget ϵ and the noisy prefix tree height h. We follow the evaluation scheme from previous works [31], [7]. For each privacy budget, we generate 40,000 random count queries



(a) $max|Q| = 3$ (b) $max|Q| = 6$

(c) $max|Q| = 9$ (d) $max|Q| = 12$

Figure 3: Average relative error vs. privacy budget.

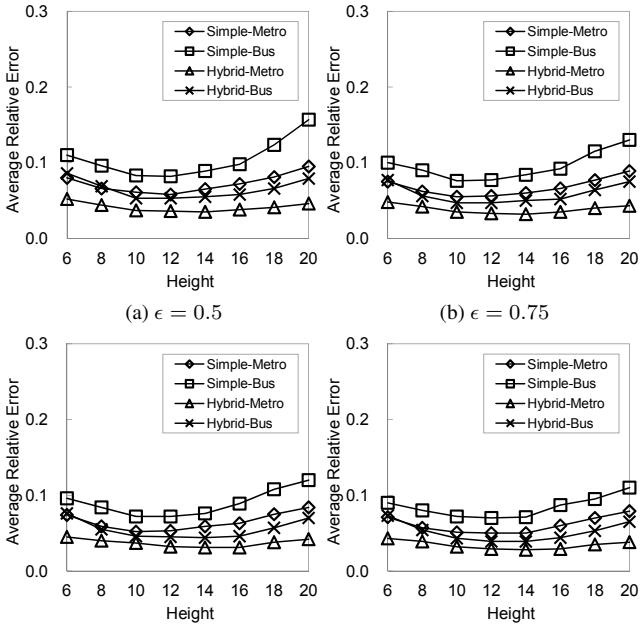
with varying numbers of locations. We call the number of locations in a query the *length* of the query. We divide the query set into 4 subsets such that the query length of the i-th subset is uniformly distributed in $[1, \frac{ih}{4}]$ and each location is randomly drawn from the location universe $\mathcal{L}$. The sanity bound s is set to 0.1% of the dataset size, the same as [31], [7].

Figure 3 examines average relative errors under varying privacy budgets from 0.5 to 1.5 with $h = 12$. The X-axes represent the different query subsets by their maximum length $max|Q|$. As expected, the average relative errors decrease when ϵ increases because less noise is added and the construction process is more precise. In general, our approach maintains high utility for count queries. Even in the worst case ($\epsilon = 0.5$ and $max|Q| = 3$), the average relative error of *Hybrid* is still less than 8.2% on both datasets. Such level of relative errors is acceptable for data analysis at the STM. We can also observe that a hybrid-granularity structure can substantially reduce average relative errors (with 33%-48% improvement), especially on *Bus*, which is more sparse.

Figure 4 studies how average relative errors vary under different h values with maximum query length fixed to 6. We can observe that with the increase of h, the relative errors do not decrease monotonically. Initially, the relative errors decrease when h increases because the increment of h allows to retain more information from the underlying database. However, after a certain threshold, the relative errors become larger with the increase of h, because when h gets larger, the noise added to each level grows quickly. It is interesting to see that the hybrid-granularity structure indeed better eliminates noise, and makes relative errors less sensitive to varying h values. This allows a wider range of h values (e.g., 10-16) to be used in order to obtain desirable relative errors.

Frequent Sequential Pattern Mining. In the second set of experiments, we demonstrate the utility of sanitized data by frequent sequential pattern mining. Specifically, we employ PrefixSpan to mine frequent sequential patterns [4].

[4]An implementation of the PrefixSpan algorithm proposed in [25], available at http://code.google.com/p/prefixspan/.



(a) $\epsilon = 0.5$ (b) $\epsilon = 0.75$

(c) $\epsilon = 1.0$ (d) $\epsilon = 1.25$

Figure 4: Average relative error vs. prefix tree height.

Table 3: Utility for frequent sequential pattern mining vs. k

k	TP (M/B) *Simple*	FP(FD) (M/B) *Simple*	TP (M/B) *Hybrid*	FP(FD) (M/B) *Hybrid*
100	99/97	1/3	100/100	0/0
150	143/139	7/11	149/144	1/6
200	178/168	22/32	185/177	15/23
250	209/195	41/55	220/209	30/41
300	241/212	59/88	257/233	43/67

Table 3 shows how the utility changes with different top k values while fixing $\epsilon = 1.0$ and $h = 12$. When $k = 100$, the sanitized data generated by *Hybrid* is able to give the exact top 100 most frequent patterns that are of size greater than 1. With the increase of k values, the *accuracy* (the ratio of true positive to k) decreases. However, even when $k = 300$, the accuracy of *Hybrid* is still as high as $257/300 = 85.7\%$ on *Metro* and $233/300 = 77.7\%$ on *Bus*. Again we can observe that $Hybrid$ always outperforms $Simple$ on both datasets under all k values. When $k = 300$, the improvement due to the hybrid-granularity structure is 6.6% on *Metro* and 9.9% on *Bus*.

Table 4 presents the utility for frequent sequential pattern mining under different ϵ values while fixing $h = 12$ and $k = 300$. Generally, larger privacy budgets lead to more true positives and fewer false positives (false drops). This conforms to the theoretical analysis that a larger privacy budget results in less noise and therefore a more accurate result. Since the most frequent sequential patterns are of small length, they have large supports from the underlying database. As a result, the utility is relatively insensitive to varying privacy budgets, and the accuracy is high even when the privacy budget is small.

Table 5 studies how the utility varies under different h values with $\epsilon = 1.0$ and $k = 300$. It is interesting to see that in general frequent sequential pattern mining is also insensitive to varying h values. This can be similarly explained by the large supports of frequent sequential patterns, which make them more resistant to noise. In addition, we can observe that good performance for frequent sequential pattern mining can be obtained by a h value between 10 and 16, the same range as that for count queries.

Table 4: Utility for frequent sequential pattern mining vs. ϵ

ϵ	TP (M/B) *Simple*	FP(FD) (M/B) *Simple*	TP (M/B) *Hybrid*	FP(FD) (M/B) *Hybrid*
0.5	227/194	73/106	244/215	56/85
0.75	239/206	61/94	253/224	47/76
1.0	241/212	59/88	257/233	43/67
1.25	243/216	57/84	259/238	41/62
1.5	248/224	52/76	261/242	39/58

Table 5: Utility for frequent sequential pattern mining vs. h

h	TP (M/B) *Simple*	FP(FD) (M/B) *Simple*	TP (M/B) *Hybrid*	FP(FD) (M/B) *Hybrid*
6	234/212	66/88	241/221	59/79
8	240/217	60/83	254/232	46/68
10	241/215	58/85	255/236	45/64
12	241/212	59/88	257/233	43/67
14	241/212	59/88	258/233	42/67
16	240/210	60/90	258/231	42/69
18	240/209	60/91	255/230	45/70
20	238/206	62/94	254/228	46/72

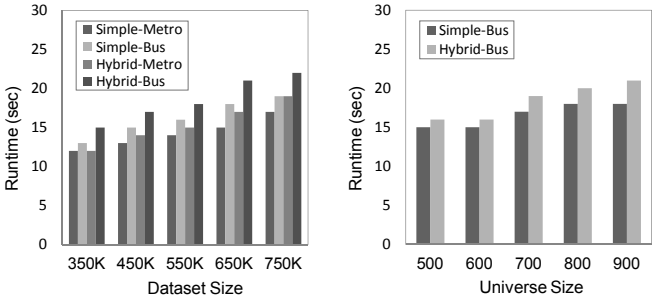
5.2 Scalability

In the last set of experiments, we examine the scalability of our approach. Recall that the runtime complexity of our approach is dominated by the database size $|\mathcal{D}|$ and the location universe size $|\mathcal{L}|$. Therefore, we study the runtime under different database sizes and different location universe sizes. Figure 5.a presents the runtime under different database sizes with $\epsilon = 1.0$ and $h = 20$ for both *Simple* and *Hybrid*. The test sets are generated by randomly extracting records from *Metro* and *Bus*. We can observe that the runtime is linear to the database size. Moreover, it can be seen that the computational cost of a hybrid-granularity prefix tree structure is negligible. This further confirms the benefit of employing a hybrid-granularity prefix tree.

Figure 5.b shows how runtime varies under different location universe sizes. Since *Metro* is of a small universe size, we only study the effect of universe sizes on *Bus*. For each universe size, we remove all locations falling out of the universe from *Bus*. This results in a smaller database size. Consequently, we fix the database size for all test sets to 600,000. Again, it can be observed that the runtime scales linearly with the location universe size and that the computational cost of *Hybrid* is comparable to that of *Simple* under different location universe sizes. As a summary, our approach is scalable to large sequential datasets. It takes less than 22 seconds to sanitize both datasets in all previous experiments.

6. CONCLUSION AND LESSON LEARNED

In this paper, we have studied the problem of publishing transit data at the STM in the framework of differential privacy. For the first time, we present a practical solution for sanitizing large-scale sequential data, which can also be seamlessly applied to trajectory data. Our solution has been tested on real-life STM transit data for two fundamental data analysis tasks performed at the STM and exhibits satisfactory effectiveness and efficiency. We believe that our solution could benefit many other sectors that are facing the dilemma between the demands of sequential data publishing and privacy protection.



(a) Runtime vs. $|\mathcal{D}|$ (b) Runtime vs. $|\mathcal{L}|$

Figure 5: Runtime vs. different parameters.

Finally, we would like to share our collaborative experience with the public transport sector. Transit data is useful for many data analysis tasks. So far, we have focused on two most fundamental ones. Though many data analysis tasks can be done based on accurate count queries, it is still worth exploring the utility of differentially private release for more complex data analysis. Besides the technical aspect, it is equally important to educate transport service practitioners about the latest privacy-preserving technology. This is especially important for a solution based on differential privacy because any accidental release of *deterministic* aggregate-level information may expose previously sanitized data to privacy threats [19].

7. ACKNOWLEDGMENT

The research is supported in part by the Discovery Grants (356065-2008) and Canada Graduate Scholarship from the Natural Sciences and Engineering Research Council of Canada (NSERC).

8. REFERENCES

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

[2] R. Agrawal and R. Srikant. Mining sequential patterns. In *ICDE*, 1995.

[3] A. Blum, K. Ligett, and A. Roth. A learning theory approach to non-interactive database privacy. In *STOC*, 2008.

[4] A. Bruno, M. Catherine, and T. Martin. Mining public transport user behaviour from smart card data. In *IFAC SICPM*, 2006.

[5] M. Catherine, T. Martin, and A. Bruno. Measuring transit performance using smart card data. In *World Conference on Transportation Research*, 2007.

[6] R. Chen, B. C. M. Fung, N. Mohammed, and B. C. Desai. Privacy-preserving trajectory data publishing by local suppression. *Information Sciences*, in press.

[7] R. Chen, N. Mohammed, B. C. M. Fung, B. C. Desai, and L. Xiong. Publishing set-valued data via differential privacy. *PVLDB*, 2011.

[8] G. Cormode, M. Procopiuc, and D. Srivastava. Differentially private summaries for sparse data. In *ICDT*, 2012.

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

[10] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, 2006.

[11] C. Dwork, M. Naor, O. Reingold, G. N. Rothblum, and S. Vadhan. On the complexity of differentially private data release: Efficient algorithms and hardness results. In *STOC*, 2009.

[12] A. V. Evfinievski, R. Srikant, R. Agrawal, and J. Gehrke. Privacy preserving mining of association rules. *Information Systems*, 29(4):343–364, 2004.

[13] B. C. M. Fung, M. Cao, B. C. Desai, and H. Xu. Privacy protection for RFID data. In *SAC*, 2009.

[14] S. R. Ganta, S. P. Kasiviswanathan, and A. Smith. Composition attacks and auxiliary information in data privacy. In *SIGKDD*, 2008.

[15] A. Ghosh, T. Roughgarden, and M. Sundararajan. Universally utility-maximizing privacy mechanisms. In *STOC*, 2009.

[16] M. Hay, V. Rastogi, G. Miklau, and D. Suciu. Boosting the accuracy of differentially private histograms through consistency. *PVLDB*, 2010.

[17] H. Hu, J. Xu, S. T. On, J. Du, and J. K.-Y. Ng. Privacy-aware location data publishing. *ACM TODS*, 35(3):17, 2010.

[18] D. Kifer. Attacks on privacy and definetti's theorem. In *SIGMOD*, 2009.

[19] D. Kifer and A. Machanavajjhala. No free lunch in data privacy. In *SIGMOD*, 2011.

[20] P. Marie-Pierre, T. M., and M. Catherine. Smart card data use in public transit: A literature review. *Transportation Research C: Emerging Technologies*, 19(4):557–568, 2011.

[21] F. McSherry. Privacy integrated queries: An extensible platform for privacy-preserving data analysis. In *SIGMOD*, 2009.

[22] F. McSherry and K. Talwar. Mechanism design via differential privacy. In *FOCS*, 2007.

[23] N. Mohammed, R. Chen, B. C. M. Fung, and P. S. Yu. Differentially private data release for data mining. In *SIGKDD*, 2011.

[24] 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.

[25] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M. Hsu:. Prefixspan: Mining sequential patterns by prefix-projected growth. In *ICDE*, 2001.

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

[27] M. Terrovitis and N. Mamoulis. Privacy preservation in the publication of trajectories. In *MDM*, 2008.

[28] M. Utsunomiya, J. Attanucci, and N. Wilson. Potential uses of transit smart card registration and transaction data to improve transit planning. *Transportation Research Record: Journal of the Transportation Research Board*, (1971):119–126, 2006.

[29] R. C. W. Wong, A. Fu, K. Wang, P. S. Yu, and J. Pei. Can the utility of anonymized data be used for privacy breaches. *ACM TKDD*, 5(3):16, 2011.

[30] X. Xiao, G. Bender, M. Hay, and J. Gehrke. ireduct: Differential privacy with reduced relative errors. In *SIGMOD*, 2011.

[31] X. Xiao, G. Wang, and J. Gehrke. Differential privacy via wavelet transforms. In *ICDE*, 2010.

[32] R. Yarovoy, F. Bonchi, L. V. S. Lakshmanan, and W. H. Wang. Anonymizing moving objects: How to hide a MOB in a crowd? In *EDBT*, 2009.