Preserving Individual Privacy in Spatio-Temporal Data Analytics

By

Liyue Fan
Doctor of Philosophy

Computer Science and Informatics

Preserving Individual Privacy in Spatio-Temporal Data Analytics

By

Liyue Fan
B.S., Zhejiang University, 2008

Advisor: Li Xiong, Ph.D.

An abstract of
A dissertation submitted to the Faculty of the
James T. Laney School of Graduate Studies of Emory University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in Computer Science and Informatics
2014

Abstract

Preserving Individual Privacy in Spatio-Temporal Data Analytics
By Liyue Fan

We live in the age of big data. With an increasing number of people, devices, and sensors connected with digital networks, individual data now can be largely collected and analyzed by data mining applications for social good as well as for commercial interests. However, the data generated by individual users exhibit unique behavioral patterns and sensitive information, and therefore must be transformed prior to the release for analysis. The AOL search log release in 2006 is an example of privacy catastrophe, where the searches of an innocent citizen were quickly re-identified by a newspaper journalist. In this dissertation, I present a novel framework to release continuous aggregation of private data for an important class of real-time data mining tasks, such as disease outbreak detection and web mining, to name a few. The key innovation is that the proposed framework captures the underlying dynamics of the continual aggregate statistics with time series state-space models, and simultaneously adopts filtering techniques to correct the observed, noisy data. It can be shown that the new framework provides a rigorous, provable privacy guarantee to individual data contributors without compromising the output analysis results. Extensive empirical studies confirm that it will enable privacy-preserving data analytical tasks in a broad range of application domains.

Preserving Individual Privacy in Spatio-Temporal Data Analytics

By

Liyue Fan
B.S., Zhejiang University, 2008

Advisor: Li Xiong, Ph.D.

A dissertation submitted to the Faculty of the
James T. Laney School of Graduate Studies of Emory University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in Computer Science and Informatics
2014

Acknowledgments

Foremost I offer my sincerest gratitude to my advisor, Dr. Li Xiong, who has supported me throughout my dissertation while allowing me the room to work on my own. Without her this dissertation would not have been completed. I simply could not wish for a friendlier/more understanding advisor. My committee members, Dr. Vaidy Sumderam and Dr. Fusheng Wang, have provided me with much advice and insight during the course of my research. Their time and effort are highly appreciated. I would like to thank Dr. James Lu, for his patience and encouragement during my first two years at Emory and for the teaching and career advice he offered throughout my PhD study. Math&CS staff have been extremely helpful for all these years and I have enjoyed the company of a cheerful group of fellow graduate students.

To Mom and Dad

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

We live in the age of big data. With an increasing number of people, devices, and sensors connected with digital networks, individual data now can be largely collected and analyzed to understand important phenomena. An important class of data mining tasks which rely on real-time analysis of aggregated private data, are particularly investigated in this dissertation. A few example tasks include:

> **Disease Surveillance**: A health care provider gathers data from individual visitors. The aggregated data, e.g. the daily number of Influenza cases, is then shared with third parties, e.g., researchers, in order to monitor and to detect seasonal epidemic outbreaks at the earliest [5, 33, 62].
>
> **Web Mining**: A web server collects data from individual browsing sessions. The aggregated data, e.g. the number of requests to each web page during each time unit, can be mined to discover popular web pages [49], interesting contents [25], as well as suspicious behavior and attacks [11].

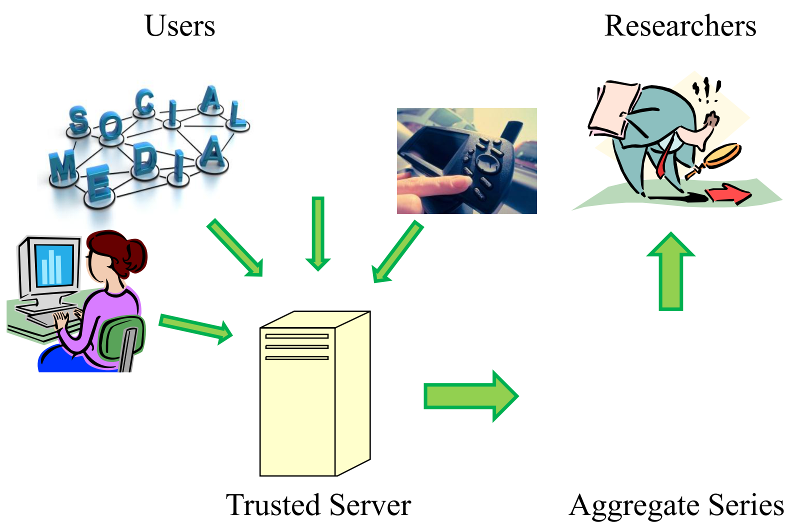


Figure 1.1: Real-Time Analysis of Aggregated Private Data

> **Traffic Monitoring**: A GPS service provider gathers data from individual users about their locations, speeds, mobility, etc. The aggregated data, for instance, the number of users at each region during each time period, can be mined for commercial interest, such as popular places, as well as public interest, such as congestion patterns on the roads [38].

Figure 1.1 illustrates how real-time aggregate monitoring tasks can be accomplished. A trusted central server gathers data from a large number of individual subscribers, i.e. dynamic information regarding their health, location, service usage, and online activities. The collected data may be then aggregated and continuously shared as aggregate time series with un-trusted entities for research purposes.

Although analyzing individually generated data is clearly beneficial, user concerns rise from a privacy perspective, as they contribute an increasing amount of private information. As a matter of fact, the uniqueness of each person is promoted by the continuous collection of individual data, which represents behavioral patterns and personal preferences. Therefore, each individual is more likely to “stand out” in the data space and be easily distinguished from those of the rest. The AOL data release in 2006 is an unfortunate example of privacy catastrophe [3], in which the search logs of an innocent citizen were quickly identified by a newspaper journalist. A recent study by de Montjoye et al. [18] concludes that human mobility patterns are highly unique and four spatio-temporal points are enough to uniquely identify 95% of the individuals. In order to protect data contributors from re-identification attacks, their private data must be transformed prior to release for analysis.

The current state-of-the-art paradigm for privacy-preserving data publishing is *differential privacy* [4, 22], which requires that the aggregate statistics reported by a data publisher be perturbed by a randomized algorithm $\mathcal{A}$, so that the output of $\mathcal{A}$ remains roughly the same even if any single tuple in the input data is arbitrarily modified. Given the output of $\mathcal{A}$, an adversary will not be able to infer much about any single tuple in the input, and thus privacy is protected.

A plethora of works have been developed for differentially private data release, the majority of which enable one-time release of static data [40, 51, 66–68]. In the applications we consider, data values at successive time stamps are highly correlated, creating more challenges for privacy preservation. A straightforward application of differential privacy mechanism [22] which adds a Laplace noise to each aggregate value at each time stamp can lead to a very high perturbation error due to the sequential composition property [54]. Few works [12, 23, 59] studied the problem of releasing time series or continual statistics with differential privacy guarantee. The work of [59] relies on time series Discrete Fourier Transform (denoted as DFT), which is not compatible with real-time data mining tasks. Dwork et al. [23] and Chan et al. [12] studied a differentially private continual counter over a binary stream. However, both works adopt an event-level privacy model, with the perturbation mechanism designed to protect the presence of an individual event, i.e. a user's contribution to the data stream at a single time point, rather than the presence or privacy of a user.

1.2 Research Contributions

The following research questions are addressed in this dissertation:

- How to release accurate aggregates in real-time while providing *user-level* differential privacy guarantee?
- How to quantify individual privacy risk when releasing long series of aggregates?
- When releasing multi-variate aggregate time series, how to model the spatio-temporal correlation between variables and time stamps?

1.2.1 Real-Time Aggregate Monitoring with Differential Privacy (Chapter 3)

This chapter focuses on developing a generic framework for releasing aggregate time series data in real time with the guarantee of differential privacy, which hasn't been investigated by previous works. The goal of the framework is to continuously release accurate aggregated data under the notion of *user-level* differential privacy, by utilizing partial knowledge, as opposed to offline methods, about the aggregate data series.

To this end, two challenges are examined in this chapter: predictability and controllability. Firstly, we explore the possibility to release accurate values at every time stamp, given the underlying dynamics of the aggregate time series and the differential privacy mechanism. Secondly, we investigate sampling approaches in the temporal domain, reducing the overall perturbation error by only querying true aggregates at sampling time points.

To improve the accuracy of data release at each time stamp, we establish the state-space model for the time series to monitor and use filtering techniques to estimate the original data values from observed values. By assuming a process model that characterizes the time series, we can reduce the impact of perturbation error introduced by the differential privacy mechanism. This is achieved by combining the noisy observation with a prediction based on the process model. The combined value, also referred to as correction or posterior estimate, provides an educated guess rather than a pure perturbed answer. The posterior estimate is then fed back to the system for future predictions and for dynamically adjusting the sampling process.

To minimize the overall privacy cost, hence, the overall perturbation error, we propose to sample the time series data as needed. Besides the fixed-rate sampling method, we introduce an adaptive sampling algorithm which adjusts the sampling rate with PID control (stands for *Proportional*, *Integral*, and *Derivative*), which is the most common form of feedback control. We design a PID controller to detect data dynamics from the estimates derived by the filtering techniques. Ultimately, we increase the sampling frequency when data is going through rapid changes and vice versa.

1.2.2 Estimating Individual Privacy Risk with Domain Statistics (Chapter 4)

This chapter investigates the problem of estimating individual privacy risk when the proposed framework is applied to aggregate monitoring tasks over a long period of time. The risk parameter adopted by previous works, called *global sensitivity*, equals to the total number of time points in aggregate monitoring applications, leading to high perturbation error due to differential privacy requirement. The goal of this chapter is to bound the individual privacy risk over a long time period with a small constant, thus reducing the amount of differential privacy perturbation when releasing continual aggregates.

We propose to estimate individual privacy risk with publicly available domain statistics, which quantifies the probability of individuals contributing to the aggregate time series. We illustrate the estimation method with a case study on epidemic outbreak detection, where the monitoring period is often long, i.e. 10 years, and public survey statistics are available about patient visits and diagnosis. We show that for the majority of individuals, the contribution to the aggregate time series is a very small constant, compared to the length of series, with high probability. To evaluate the utility of released aggregates, we conduct empirical studies with simulated data sets with different characteristics. Three extensively used epidemic outbreak detection algorithms, i.e. C1, C2, and C3 [42], are performed on both the original data series and the private, released data series.

1.2.3 Modeling Spatio-Temporal Correlation of Multi-Variate Aggregate Time Series (Chapter 5)

This chapter extends the proposed framework to releasing multi-variate time series, in order to monitor multiple aggregates simultaneously, e.g. for web mining and traffic monitoring. Two challenges are present in this chapter. The first challenge is to model the spatio-temporal correlation between multiple aggregates at different time stamps. The second challenge is the scale of the problem, where the number of aggregates to monitor is high.

We first explore the possibility of modeling the spatio-temporal correlation with individual behavioral patterns for web mining tasks. We propose to incorporate the Markov property of web browsing behavior in the process model, when applying the proposed framework to release web visit aggregates. For efficient learning of model parameters, we propose to obtain a small sample of browsing data and to set entry values in the high dimensional covariance matrix with genetic algorithms. We conduct extensive empirical studies with real-world dataset and examine three differential utility metrics of the released multi-variate aggregate time series data.

We then investigate complex spatio-temporal correlations, i.e. traffic histogram over a road network, which cannot be captured by a linear process model in the proposed framework. Furthermore, when monitoring traffic count on a two-dimensional spatial grid, data sparsity becomes another issue for privacy-preserving mechanisms, as a result of large domain size. We propose to model the spatial correlation between grid cells based on the background road network and to group cells with similar density to overcome data sparsity. A Quadtree based space partitioning algorithm is developed and evaluated with simulated moving object data over a real-world road network.

Chapter 2

Related Works

Here we briefly review the recent, relevant works on differential privacy, time series analysis, filtering techniques, and anomaly detection applications.

2.1 Differential Privacy

Dwork et al. [22] established the guideline to guarantee differential privacy for individual aggregate queries by calibrating the Laplacian noise to the global sensitivity of each query. Since then, various mechanisms have been proposed to enhance the accuracy of differentially private data release. Blum et al. [4] proved the possibility of non-interactive data release satisfying differential privacy for queries with polynomial VC-dimension, such as predicate queries. Dwork et al. [24] further proposed more efficient algorithms to release private sanitization of a data set with hardness results obtained. The work of Hay et al. [40] improved the accuracy of a tree of counting queries through consistency check, which is done as a post-processing procedure after adding Laplace noise.

Several recent works [16, 40, 51, 66–68] study the counting queries on multi-dimensional data, also referred to as histograms or contingency tables, where the multi-dimensional data can be indexed by a tree structure and each level in the tree is an increasingly fine-grained summary/count. Cormode et al [16] propose the class of "private spatial decompositions" and conclude that the hybrid structure *kd-hybrid* provides an accurate yet efficient solution compared to alternatives. A recent study [65], aiming to reduce the relative error, suggests to inject different amount of Laplace noise based on the query result and works well with multidimensional data. Several other works studied differentially private mechanisms for particular kinds of data, such as search logs [48] [41], sparse data [17], and set-valued data [14]. When applied to highly self-correlated time series data, all the above methods, designed to perturb static data, become problematic because of highly compound Laplace perturbation error.

2.2 Time Series Analysis and Privacy

Time series data is pervasively encountered in the fields of engineering, science, sociology, and economics. Various techniques [7], such as ARIMA modeling, exponential smoothing, ARAR, and Holt-Winters methods, have been studied for time series forecasting. Papadimitriou et al. [56] studied the trade-offs between time series compressibility property and perturbation. They proposed two algorithms based on Fast Fourier Transform (FFT) and Discrete Wavelet Transform (DWT) respectively to perturb time series frequencies. However, the proposed additive noise does not guarantee differential privacy, leaving sensitive information vulnerable to adversaries with strong background knowledge.

Rastogi and Nath [59] proposed a Discrete Fourier Transform (DFT) based algorithm which guarantees differential privacy by perturbing the discrete Fourier coefficients. However, this algorithm cannot release real-time count data in a streaming environment. The recent works [12] [23] on continuous data streams defined the *event-level* privacy to protect an event, i.e. one user's presence at a particular time point, rather than the presence of that user. For example, if one user contributes to the aggregation at time $k-1$, k, and $k+1$, the event-level privacy protects the user's presence at only one of the three time points, resulting the rest two open to attack.

Gőtz et al. [37] proposed a novel approach (MaskIt) for releasing user context streams with suppression. MaskIt allows user-defined sensitive contexts that are carefully checked in the released stream, with the goal of limiting the adversary's ability of learn about user sensitive contexts from the released stream. They also propose to model user movement patterns by a Markov chain. In the event stream domain, a similar notion of privacy based on suppression has been proposed in [64]. In this approach, the sensitive information is related to the presence of private patterns in the data stream. The suppression of these patterns is performed over the stream to maximize the utility for the useful non-sensitive patterns reported in the released series.

2.3 Filtering

In our context, "filtering" refers to the derivation of posterior estimates based on a sequence of noisy measurements, in hope of removing background noise from a signal. The Kalman filter, which is based on the use of state-space techniques and recursive algorithms, provides an optimal estimator for linear Gaussian problems. R.E. Kalman published the seminal paper on the Kalman filter [45] in 1960. Since then, it has become widely applied to areas of signal processing [8] and assisted navigation systems [2]. It has also gained popularity in other areas of engineering. One particular application is to wireless sensor networks. Jain et al. [44] adopted a dual Kalman filter model on both server and remote sensors to filter out as much data as possible to conserve resources. Their main concern was to minimize memory usage and communication overhead between sensors and the central server by storing dynamic procedures instead of static data. Increasingly, it has become important to include nonlinearity and non-Gaussianity in order to model the underlying dynamics of a system more accurately. A very widely used estimator for nonlinear systems is the extended Kalman filter (EKF) [60] which linearizes the current mean and error covariance. Masreliez [53] proposed solutions to the non-Gaussian filtering problems with linear models in 1975. His algorithms retain the computationally attractive structure of the Kalman filter but require convolution operations which are hard to implement in all but very simple situations, for instance, when noises can be modeled as a Gaussian mixture. Gordon et. al [36] introduced particle filters for solving non-linear non-Gaussian estimation problems in 1993. Since then, particle methods have become very popular due to the advantage that they do not rely on any local linearisation or any crude functional approximation. Pitt and Shephard [58] introduced auxiliary particle filter and adaptation. Doucet et al. [20] proposed the optimal importance distribution, approximation, smoothing, and Rao-Blackwellization. A few tutorials, for instance [1] [21], on particle methods have been published and cover most sequential Monte Carlo algorithms in particle filtering to facilitate implementation.

2.4 Anomaly Detection

At an abstract level, an anomaly is defined as a pattern that does not conform to expected normal behavior [13]. Anomaly detection has been researched within diverse research areas and application domains, such as epidemic outbreak detection for syndrome surveillance [43] and intrusion detection for cyber-security [9]. Both tasks are conducted on continual aggregate statistics, i.e. aggregate time series. The aberration detection algorithms studied by Jackson et al. [43] aim to find disease outbreaks from daily counts of diagnosed cases. The work of Caberera et al. [9] addressed network intrusion detection by establishing statistical models for the number of incoming connections within a given time interval.

Specifically for outbreak detection, there are a multitude of algorithms that have been reported and applied to a variety of disease studies. Most algorithms compare the current signal, i.e. current *count* of diagnosed cases, with a baseline period, i.e. previously released *counts*, in order to determine whether there is an outbreak or not. Three control-chart based algorithms proposed by Hutwagner et al. [42], commonly referred to as EARS [33] C1, C2 and C3, require little baseline data and have been found to provide early detection of outbreaks. The Negative Binomial Cusum (NBC) method, originally proposed in Hawkins and Olwell [39], is reported to reduce the number of false positives generated by other cusum methods. Unlike the cusum methods, the Historical Limit Method (HLM) [61] incorporates historical data and accounts for seasonality by design. Gatton et al. [34] proposed to model the number disease cases in a time period as a Poisson process and their method considers years of historical data as baseline.

Chapter 3

Real-Time Aggregate Monitoring with Differential Privacy

In this chapter, we formally define the problem of real-time aggregate monitoring with differential privacy and introduce FAST, a generic framework with Filtering and Adaptive Sampling to release differentially private aggregate Time series.

3.1 Preliminaries

3.1.1 Problem Statement

We formally define an aggregate time series as follows:

Definition 3.1 (Aggregate Time Series)**.** A univariate, discrete time series $\mathbf{X} = \{x_k\}$ is a set of values of an aggregate variable x at discrete time k, where $0 \leq k < T$ and T is the length of the series.

In particular, $\mathbf{X}$ is a *count* series in our example applications, such as the daily count of patients diagnosed of Influenza, or the hourly count of vehicles passing by a gas station. This assumption will hold true for the rest of this dissertation. However, other types of aggregate function, such as *min*, *max*, *average*, and *sum*, can also be monitored by FAST with slight adaptations. Our goal is to release in real-time a private version of $\mathbf{X}$, denoted as $\mathbf{R} = \{r_k\}$, that satisfies differential privacy.

Definition 3.2 (Utility Metric)**.** We measure the quality of a private, released series $\mathbf{R}$ by the average relative error E between $\mathbf{R}$ and the original series $\mathbf{X}$:

$$E = \frac{1}{T}\sum_{k}\frac{|r_k - x_k|}{max\{x_k, \delta\}} \tag{3.1}$$

where δ is a user-specified constant (also referred to as *sanitary bound* in [65]) to mitigate the effect of excessively small query results. We set $\delta = 1$ throughout the entire series for *count* data.

Intuitively, the utility of $\mathbf{R}$ increases as each r_k approaches x_k, the extreme case of which would have $r_k = x_k$ for each k. However, a privacy-preserving algorithm is likely to perturb original data values in order to protect individual privacy. Therefore, a mechanism that guarantees user privacy and yields high utility is highly desirable.

3.1.2 Differential Privacy

The privacy guarantee provided by FAST is *differential privacy* [4]. Simply put, a mechanism is differentially private if its outcome is not significantly affected by the removal or addition of a single user. An adversary thus learns approximately the same information about any individual user, irrespective of his/her presence or absence in the original database.

Definition 3.3 (α-Differential Privacy [4])**.** A non-interactive privacy mechanism $\mathcal{A}$ gives α-differential privacy if for any dataset D_1 and D_2 differing on at most one record, and for any possible anonymized dataset $\widetilde{D} \in Range(\mathcal{A})$,

$$Pr[\mathcal{A}(D_1) = \widetilde{D}] \leq e^{\alpha} \times Pr[\mathcal{A}(D_2) = \widetilde{D}] \tag{3.2}$$

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

The privacy parameter α, also called the *privacy budget* [54], specifies the degree of privacy offered. Intuitively, a lower value of α implies stronger privacy guarantee and a larger perturbation noise, and a higher value of α implies a weaker guarantee while possibly achieving higher accuracy.

Laplace Mechanism. Dwork et al. [22] show that α-differential privacy can be achieved by adding i.i.d. noise $\tilde{N}$ to the query result $q(D)$:

$$\tilde{q}(D) = q(D) + \tilde{N} \tag{3.3}$$

The magnitude of $\tilde{N}$ conforms to a Laplace distribution with probability $p(x|\lambda) = \frac{1}{2\lambda}e^{-|x|/\lambda}$ and $\lambda = GS(q)/\alpha$, where $GS(q)$ represents the *global sensitivity* [22] of query q. In our target applications, each aggregate value is a *count* query and it is known $GS(count) = 1$. Later on, we denote the Laplace distribution with 0 mean and λ scale as $Lap(0, \lambda)$.

Composition. The composition properties of differential privacy provide privacy guarantees for a sequence of computations, e.g. a sequence of *count* queries.

Theorem 3.4 (Sequential Composition [54])**.** Let $\mathcal{A}_i$ each provide α_i-differential privacy. A sequence of $\mathcal{A}_i(D)$ over the dataset D provides $(\sum_i \alpha_i)$-differential privacy.

User-level privacy vs. Event-level privacy. The work [23] proposed a differentially private continual counter with the notion of *event*-level privacy, where the neighboring databases differ at u_i, a user u's contribution at time stamp i. In our study, we provide a stronger privacy guarantee, *user*-level

Algorithm 1 Laplace Perturbation Algorithm(LPA)

Input: Raw series $\mathbf{X}$, privacy budget α

Output: Released series $\mathbf{R}$

```
1: for each $k \in 0, 1, ..., T-1$ do
2:     $r_k \leftarrow$ perturb $x_k$ by $Lap(0, \frac{T}{\alpha})$;
```

privacy, where the neighboring databases differ at the user u, i.e. u's contribution at all time stamps, thus protecting sensitive information about user u at any time.

3.1.3 Existing Solutions

Here we present the baseline Laplace perturbation algorithm and a recently proposed transformation-based algorithm. Empirical studies of the two algorithms against our proposed solution are included in Section 5.

Laplace Perturbation Algorithm. A baseline solution to sharing differentially private time series is to apply the standard Laplace perturbation at each time stamp. If every query satisfies α/T-differential privacy, by Theorem 1 the sequence of queries guarantees α-differential privacy. We summarize the baseline algorithm in Algorithm 1 and Line 2 represents the Laplace mechanism to guarantee α/T-differential privacy for each released aggregate.

Discrete Fourier Transform. Algorithm 2 outlines the Fourier Perturbation Algorithm proposed by Rastogi and Nath [59]. It begins by computing $\mathbf{F}^d$, which is composed of the first d Fourier coefficients in the Discrete Fourier Transform (DFT) of $\mathbf{X}$, with the j^{th} coefficient given as: $DFT(\mathbf{X})_j$ $= \sum_{i=0}^{T-1} e^{\frac{2\pi\sqrt{-1}}{T}ji} x_i$. Then it perturbs $\mathbf{F}^d$ using LPA algorithm with privacy budget α, resulting a noisy estimate $\widetilde{\mathbf{F}}^d$. This perturbation is to guarantee differential privacy. Denote $\mathbf{PAD}^T(\widetilde{\mathbf{F}}^d)$ the sequence of length T by append-

Algorithm 2 Discrete Fourier Transform(DFT)

Input: Raw series $\mathbf{X}$, privacy budget α, parameter d

Output: Released time-series $\mathbf{R}$

```
1: compute $\mathbf{F}^d \leftarrow \mathbf{DFT}^d(X)$
2: compute $\widetilde{\mathbf{F}}^d \leftarrow LPA(\mathbf{F}^d, \alpha)$;
3: compute $\mathbf{R} \leftarrow \mathbf{IDFT}(\mathbf{PAD}^T(\widetilde{\mathbf{F}}^d))$;
```

ing $T - d$ zeros to $\widetilde{\mathbf{F}}^d$. The algorithm finally computes the Inverse Discrete Fourier Transform(IDFT) of $\mathbf{PAD}^T(\widetilde{\mathbf{F}}^d)$ to get $\mathbf{R}$. The j^{th} element of the inverse is given as: $IDFT(\mathbf{X})_j = \frac{1}{T}\sum_{i=0}^{T-1} e^{-\frac{2\pi\sqrt{-1}}{T}ji} x_i$.

The number of coefficients d is a user-specified parameter. In our empirical study, we set $d = 20$ according to the original paper [59]. As each $IDFT(\mathbf{X})_j$ may be a complex number due to truncation and perturbation, the authors proposed to use L_1 distance to measure the quality of the inverse series. To be consistent, we adopt the same metric in our empirical study for this algorithm. We slightly abuse the term and refer to their algorithm as DFT in the rest of the chapter.

3.2 FAST

We propose FAST, a novel solution to sharing time-series data under differential privacy. It allows fully automated adaptation to data dynamics and highly accurate time-series prediction and correction. Figure 3.1 illustrates the framework of FAST. The key steps are outlined below:

- For each time stamp k, the **adaptive sampling** component determines whether to sample/query the **input** time-series or not.

- If k is a sampling point, the data value at k is perturbed by the **Laplace**

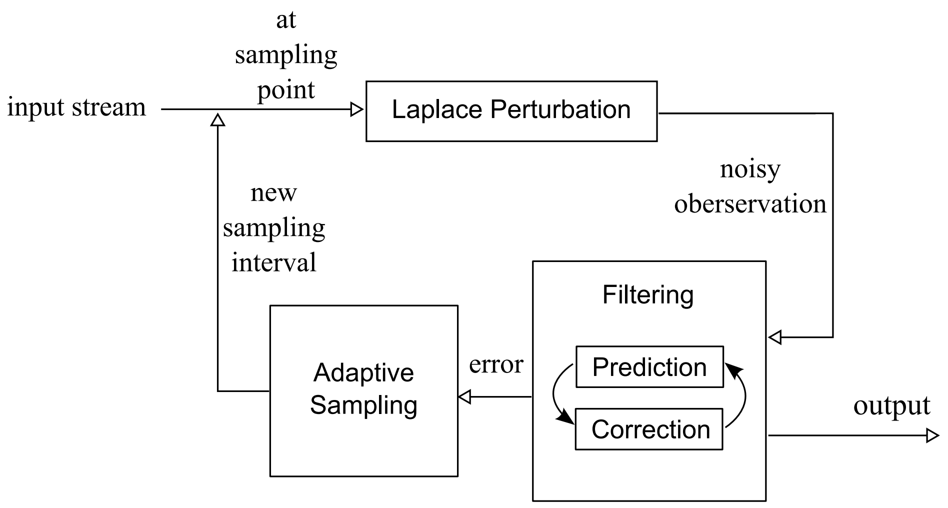


Figure 3.1: FAST Framework

mechanism to guarantee differential privacy. This perturbed value is then received by the **filtering** component for posterior estimation.

- The **filtering** component produces a *prediction* of data value based on an internal state model at every time stamp. The prediction, i.e. prior estimate, is released to **output** at a non-sampling point, while a *correction*, i.e. posterior estimate based on both the noisy observation and the prediction, is released at a sampling point.

- The error between the prior and the posterior estimates is then fed to the **adaptive sampling** component to adjust the sampling rate. Once the user-specified privacy budget α is used up, the system will stop sampling the input series and will predict at each onward time stamp.

Algorithm 3 summarizes our proposed framework. Given a raw series $\mathbf{X}$, overall privacy budget α, and maximum number of samples allowed M ($M \leq T$), FAST provides real-time estimates of data values by the **Prediction** and **Correction** procedures and dynamically adjusts the sampling

Algorithm 3 FAST Algorithm

Input: Raw series **X**, privacy budget α, maximum number of samples allowed M

Output: Released series **R**

```
1: for each $k$ do
2:     obtain prior estimate from Prediction;
3:     if $k$ is a sampling point & $numSamples < M$
4:         $z_k \leftarrow$ perturb $x_k$ by $Lap(0, \frac{M}{\alpha})$;
5:         $numSamples$++;
6:         obtain posterior estimate from Correction;
7:         $r_k \leftarrow posterior$
8:         adjust sampling rate by Adaptive Sampling;
9:     else
10:        $r_k \leftarrow prior$;
```

rate with the **Adaptive Sampling** procedure. The details will be discussed in the next two subsections. Note that FAST evenly distributes the overall privacy budget α to each sample and the exhaustion of α can be detected if $numSamples \geq M$ (Line 3).

Theorem 3.5. FAST satisfies α-differential privacy.

Proof. Given the maximum number of samples M and the overall privacy budget α, each sample is α/M-differentially private. According to Theorem 1, Algorithm 3 satisfies α-differential privacy. □

There are two types of error to balance in our solution: *perturbation* error by the Laplace perturbation mechanism at sampling points and *prediction* error by the filtering prediction step at non-sampling points. The more we sample, the more *perturbation* error is introduced, while the *prediction* error might be reduced due to more feedback, and vice versa. FAST successfully strikes a balance between the two types of error. The adaptive sampling component reduces the perturbation error from $\Theta(T)$ (error of the baseline LPA) to $\Theta(M)$ and dynamically adjusts the sampling rate. The filtering component reduces the prediction error by model-based estimation and achieves great accuracy especially when data fits the underlying process model. Technical details of the two components are described below and empirical results which confirm the superiority of FAST are presented in Section 5.

3.2.1 Filtering

The filtering component in FAST provides estimates of monitored aggregates in order to improve the accuracy of released data per time stamp. We first establish a state-space model for the aggregate series to monitor. Then we propose and present the details of two filtering algorithms for estimation.

Process Model. Suppose the original aggregate series is generated by a underlying process. Let x_k denote the internal state, i.e. true value, of the process at time k. The states at consecutive time stamps can be modeled by the following equations:

$$x_k = x_{k-1} + \omega \tag{3.4}$$

$$\omega \sim \mathcal{N}(0, Q) \tag{3.5}$$

This constant process model indicates that adjacent values from the original time series should be consistent except for a white Gaussian noise ω, called the *process noise*, with variance Q.

Measurement Model. The noisy observation, which is obtained from the Laplace mechanism, can be represented as follows:

$$z_k = x_k + \nu \tag{3.6}$$

$$\nu \sim Lap(0, 1/\alpha_0) \tag{3.7}$$

where ν is called the *measurement noise*. Clearly, the noisy observation z_k is the true state plus the perturbation noise. Note that α_0 denotes the differential privacy budget for each sample, e.g. $\alpha_0 = \alpha/T$ if sampling at every time stamp; $\alpha_0 = \alpha/M$ if no more than M samples are allowed.

Posterior Estimate. Instead of releasing the noisy observation z_k as the baseline LPA does, we propose to release the *aposteriori* estimate of the true state x_k after obtaining z_k. The posterior estimate, denoted by $\hat{x}_k$, can be given by the following conditional expectation:

$$\hat{x}_k = E(x_k|\mathbb{Z}^k) \tag{3.8}$$

where $\mathbb{Z}^k = \{z_0, z_1, ..., z_k\}$ denotes the set of observations obtained so far. Therefore, we can derive $\hat{x}_k$ only if the *aposteriori* probability density function $f(x_k|\mathbb{Z}^k)$ can be determined. According to Bayes' theorem, we obtain the following relation between two consecutive time stamps:

$$f(x_k|\mathbb{Z}^k) = \frac{f(x_k|\mathbb{Z}^{k-1})f(z_k|x_k)}{f(z_k|\mathbb{Z}^{k-1})} \tag{3.9}$$

where the prior and the normalizing constant are given by:

$$f(x_k|\mathbb{Z}^{k-1}) = \int f(x_{k-1}|\mathbb{Z}^{k-1})f(x_k|x_{k-1})dx_{k-1} \tag{3.10}$$

$$f(z_k|\mathbb{Z}^{k-1}) = \int f(x_k|\mathbb{Z}^{k-1})f(z_k|x_k)dx_k \; . \tag{3.11}$$

In general, Equations 3.8, 3.9, and 3.10 are difficult to carry out when $f(z_k|x_k)$, i.e. $f(\nu = z_k - x_k)$, is non-Gaussian. Therefore, the posterior density cannot be analytically determined without the Gaussian assumption about the measurement noise.

We propose two solutions to the posterior estimation challenge discussed above. One is to model the Laplace perturbation noise with a Gaussian measurement noise; the other is to simulate the posterior density function with Monte Carlo methods. The details are presented below, respectively.

Gaussian Approx. of Measurement Noise

In our previous work [28], we propose to model the Laplace noise ν with an approximate, white Gaussian error

$$\nu \sim \mathcal{N}(0, R) \tag{3.12}$$

and therefore the estimation of $\hat{x}_k$ in Equation 3.12 can be solved with the classic Kalman filter [45].

The Kalman Filter. At time stamp k, the prior state estimate $\hat{x}_k^-$ is made according to the process model in Equation 3.4 and is related to the posterior estimate of the previous step:

$$\hat{x}_k^- = \hat{x}_{k-1} \ . \tag{3.13}$$

The posterior estimate $\hat{x}_k$ can be given as a linear combination of the prior $\hat{x}_k^-$ and the observation z_k:

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - \hat{x}_k^-) \ . \tag{3.14}$$

where K_k, the *Kalman Gain*, is adjusted at every time stamp to minimize the posterior error variance. Below we briefly show how K_k is derived for each time stamp k.

Let P_k^- and P_k denote the *apriori* and *aposteriori* error variance, respectively. They are defined as

$$P_k^- = E[(x_k - \hat{x}_k^-)(x_k - \hat{x}_k^-)^T] \tag{3.15}$$

$$P_k = E[(x_k - \hat{x}_k)(x_k - \hat{x}_k)^T] \ . \tag{3.16}$$

By the Gaussian assumption regarding ω and ν and given the prior error variance P_k^- at time stamp k, we can substitute Equation 3.14 and 3.15 into Equation 3.16 and apply the gradient descendant method to minimize P_k. Therefore, we obtain an optimal value for K_k as

$$K_k = P_k^-(P_k^- + R)^{-1} \tag{3.17}$$

Algorithm 4 KFPredict(k)

```
Input: Previous release $r_{k-1}$
Output: Prior estimate $\hat{x}_k^-$
1: $\hat{x}_k^- \leftarrow r_{k-1}$;
2: $P_k^- \leftarrow P_{k-1} + Q$;
```

Algorithm 5 KFCorrect(k)

```
Input: Prior $\hat{x}_k^-$ , noisy measurement $z_k$
Output: Posterior estimate $\hat{x}_k$
1: $K_k \leftarrow P_k^-(P_k^- + R)^{-1}$;
2: $\hat{x}_k \leftarrow \hat{x}_k^- + K_k(z_k - \hat{x}_k^-)$;
3: $P_k \leftarrow (1 - K_k)P_k^-$;
```

and thus the optimal P_k is

$$P_k = (1 - K_k)P_k^- \ . \tag{3.18}$$

Similarly, given P_k, we can easily project the prior error variance at $k+1$ according to Equation 3.4:

$$P_{k+1}^- = P_k + Q \ . \tag{3.19}$$

The classic Kalman filter recursively performs two operations: *Prediction* and *Correction*, which correspond to prior and posterior estimation respectively. Algorithm 4 and 5 provide details of the two estimation steps used in FAST framework.

Accuracy. Here we study the performance of the Kalman filter based algorithm without sampling. Therefore, a noisy observation is obtained and the *Prediction* and *Correction* pair is performed at every time stamp. Theoretically, the Kalman filter is optimal when the Gaussian assumption regarding the measurement noise holds, i.e. Equation 3.12. However, additional approximation error is introduced since we explicitly model the Laplace perturbation noise as Gaussian in posterior estimation. Below we analyze the posterior error $var(x_k - \hat{x}_k)$ where z_k is obtained from the Laplace mechanism and $\hat{x}_k$ is derived under the Gaussian assumption. The goal is to find the optimal approximate Gaussian noise, i.e. the optimal value of R, in order to achieve minimum variance posterior estimate. Due to the recursive nature of filtering, it's difficult to obtain a closed form for the optimal R value. We conclude our main finding in the theorem below and refer readers to Appendix A for the detailed least square analysis.

Theorem 3.6 (Optimal Approximation)**.** Given the perturbation noise distribution $Lap(0, T/\alpha)$ at every time stamp, using an approximate Gaussian noise that follows $\mathcal{N}(0, R)$ leads to the following posterior error:

$$\begin{aligned} &var(x_k - \hat{x}_k) = \\ &\quad \frac{R^2[var(x_{k-1} - \hat{x}_{k-1}) + Q]}{(P_k^- + R)^2} + \frac{2P_k^{-2}T^2}{(P_k^- + R)^2\alpha^2} \end{aligned} \tag{3.20}$$

and optimal posterior estimation requires $R \propto \frac{T^2}{\alpha^2}$.

Proof. See Appendix A. □

Theorem 3 provides guidance for choosing the Gaussian measurement noise, i.e. the value of R, to approximate the perturbation noise introduced by differential privacy mechanism. The result confirmed that the optimal R is proportional to the variance of the Laplace perturbation noise, given that the privacy budget is uniformly allocated to every time stamp.

Estimation with Sampling. Note that the posterior estimate $\hat{x}_k$ cannot be determined when noisy observation z_k is absent. When combined with sampling in our overall solution, we propose to estimate as follows: at sampling points, i.e. when noisy observations are available, both *Prediction* and

Correction will be performed and the posterior estimates will be released; at non-sampling points, i.e. when noisy observations are absent, only the *Prediction* step will be performed and the prior estimates will be released.

One advantage of the Kalman filter based algorithm is it estimates the internal state by properly weighing and combining all available data (prior and noisy observation). Another advantage is its computation efficiency: only $O(1)$ computations are required for each time stamp according to Algorithm 4 and 5.

Monte Carlo Estimation of Posterior Density

Besides modeling the Laplace perturbation with an approximate Gaussian noise, Monte Carlo methods can be used to represent the posterior density function $f(x_k|\mathbb{Z}^k)$ by simulation. In this section, we will show the solution to posterior estimation based on the Sampling-Importance-Resampling(SIR) particle filter, which is also known as bootstrap filtering [36] and condensation algorithm [52].

SIR Particle Filtering. With a collection of N weighted samples or particles, $\{x_k^i, \pi_k^i\}_{i=1}^N$, where π_k^i is the weight of particle x_k^i, the posterior density at time k can be represented as follows:

$$f(x_k|\mathbb{Z}^k) = \sum_{i=i}^{N} \pi_k^i \delta(x_k - x_k^i) \tag{3.21}$$

where $\delta(\cdot)$ is Dirac delta measure.

The weights $\{\pi_k^i\}_{i=1}^N$ are chosen according to the importance sampling method, where particles $\{x_k^i\}_{i=1}^N$ can be easily generated from a proposal $q(\cdot)$ called an *importance density.* The details of the importance sampling method are omitted here and can be found in [19]. By assuming that the importance density $q(\cdot)$ depends only on the previous state and current measurement, the following weight relationship between two successive time stamps can be

Algorithm 6 PFPredict(k)

Input: Particles at time $k-1$ $\{x_{k-1}^i, \pi_{k-1}^i\}_{i=1}^N$

Output: Prior estimate $\hat{x}_k^-$

```
1: for each $i \in 1, ..., N$ do
2:     draw $x_k^i \sim f(x_k|x_{k-1}^i)$
3: $\hat{x}_k^- \leftarrow \frac{1}{N}\sum_{i=1}^N x_k^i$
```

derived:

$$\pi_k^i \propto \pi_{k-1}^i \frac{f(z_k|x_k^i)f(x_k^i|x_{k-1}^i)}{q(x_k^i|x_{k-1}^i, z_k)} \ . \tag{3.22}$$

According to Arulampalam et al [1], it is offen convenient to choose the importance density $q(\cdot)$ to be the prior density $f(x_k|x_{k-1}^i)$. Substituting it into Equation 3.22 then yields

$$\pi_k^i \propto \pi_{k-1}^i f(z_k|x_k^i) \tag{3.23}$$

where

$$f(z_k|x_k^i) = f(\nu = z_k - x_k^i) \tag{3.24}$$

and ν follows the Laplace distribution in Equation 3.7.

The SIR particle filter explicitly employs a resampling step at every time stamp in order to circumvent degeneracy phenomenon, where after a few iterations, all but one particle will have negligible weights. In our solution, we adopt systematic resampling as recommended in [1]. Since $\pi_{k-1}^i = 1/N$ for every i after resampling, weights at time k can be simplified as follows:

$$\pi_k^i \propto f(z_k|x_k^i) \ . \tag{3.25}$$

We will use the above result for correction in the overall algorithm.

Prediction and Correction. Algorithm 6 and 7 provide details of the particle filtering based estimation algorithm used in FAST framework.

Algorithm 7 PFCorrect(k)

Input: Particles $\{x_k^i\}_{i=1}^N$, noisy measurement z_k

Output: Posterior estimate $\hat{x}_k$

```
1: for each $i \in 1, ..., N$ do
2:     assign particle weight $\pi_k^i$ according to (25)
3: normalize $\{\pi_k^i\}_{i=1}^N$
4: $\hat{x}_k \leftarrow \sum_{i=1}^N \pi_k^i x_k^i$
5: resample from $\{x_k^i, \pi_k^i\}_{i=1}^N$
```

For each particle i, the *Prediction* step (Line 1-2 in Algorithm 6) projects its value for the next time stamp according to the process model $f(x_k|x_{k-1}^i)$. Note that $f(x_k|x_{k-1}^i)$ represents the distribution $\mathcal{N}(x_{k-1}^i, Q)$, according to Equation 3.4. Once the particles are drawn, the prior estimate can be given with the uncorrected weights (Line 3 in Algorithm 6):

$$\hat{x}_k^- = \sum_{i=1}^N \pi_{k-1}^i x_k^i = \frac{1}{N} \sum_{i=1}^N x_k^i \ . \tag{3.26}$$

The *Correction* step adjusts particle weights according to the noisy observation z_k. After weight adjustment and normalization (Line 1-3 in Algorithm 7), the posterior estimate can be derived as follows (Line 4 in Algorithm 7):

$$\hat{x}_k = \sum_{i=1}^N \pi_k^i x_k^i \ . \tag{3.27}$$

As implied by the SIR particle filtering method, resampling is applied at the end of the *Correction* step (Line 5 in Algorithm 7).

The initialization of particles $\{x_0^i, \pi_0^i\}_{i=1}^N$ is non-trivial, since the distribution of the initial state is unlikely to be available in many real-time applications. Therefore, we skip the estimation steps, i.e. Equation 3.26 and eq:pfcorrect, at time 0 and release the noisy measurement z_0. Particles $\{x_0^i\}_{i=1}^N$ are then uniformly drawn from the vicinity of z_0 and $\{\pi_0^i\}_{i=1}^N$ are initialized as $1/N$.

Accuracy. We refer the readers to [21] for the accuracy and convergence results of the SIR algorithm. Intuitively, a larger number of particles implies a more accurate distribution estimation. On the other hand, a larger number of particles requires more computation time, which is crucial to real-time monitoring applications. As a matter of fact, the complexity of Algorithm 6 and 7 is $O(N)$ per time stamp. We will examine the trade-off between accuracy and run time of Algorithm 6 and 7 in the experiment section.

Estimation with Sampling. Combined with sampling in the overall solution, the particle filtering based algorithm adopts the same strategy as the Kalman filter: it releases posterior estimates at sampling points and prior estimates at non-sampling points. The utility of the particle filtering based algorithm will be evaluated against other methods in Section 5.

3.2.2 Sampling

Since each noisy observation from Laplace mechanism comes with a cost (privacy budget spent), we are motivated to sample data values through the differential privacy interface only when needed in our overall solution. Below we propose two sampling strategies: one is to sample the series with a fixed interval, while the other is to dynamically adapt the sampling rate based on feedback control.

Fixed Rate Sampling. Given a pre-defined interval I, the fixed-rate algorithm samples the time series periodically and releases the posterior estimate per I time stamps. As for the time points between two adjacent samples, a predicted value/prior estimate is released. Privacy budget $\alpha/(\frac{T}{I})$ will be spent on each sample to guarantee α-differential privacy for the entire series

Algorithm 8 Fixed Rate Sampling

Input: Current time stamp k, fixed interval I

Output: Sampling or not

```
1: if $k\%I == 0$
2:     $k$ is a sampling point
3: else
4:     $k$ is a non-sampling point
```

according to Theorem 1.

Algorithm 8 summarizes the fixed rate sampling algorithm which can be used in FAST framework. The challenge of fixed-rate sampling is to determine the optimal interval I. When increasing the sampling rate, i.e. when I is low, an extreme case of which is to issue a query at each time step as in the baseline solution, the perturbation error introduced at each time stamp is increased. On the other hand, when we decrease the sampling rate, i.e. when I is high, the perturbation at each sampling point will drop, but the published series will not reflect up-to-date data values, resulting large prediction error. We analyze the posterior error of fixed-rate sampling and find that it is very challenging to quantify and minimize the sum of error *a priori*. Detailed discussion is in Appendix B. Therefore, the fixed-rate sampling method may not be optimal in our problem setting.

Adaptive Sampling. With no *a priori* knowledge of the time series, it is desirable to detect data dynamics and to adjust the sampling rate on-the-fly. Figure 3.2 illustrates the idea of adaptive sampling. We plot the original *traffic* series as well as the number of queries (samples) issued by the adaptive sampling mechanism during each corresponding time unit. As is shown, the adaptive sampling mechanism increases sampling rate between day 20 and day 100, when the traffic count exhibits significant fluctuations,

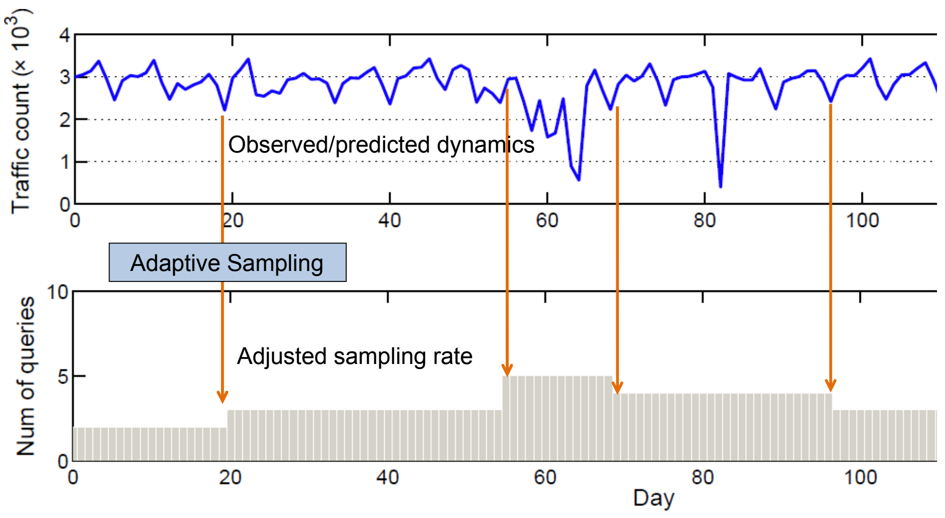


Figure 3.2: Adaptive Sampling with Traffic Data

and decreases sampling rate beyond day 100, when there's little variation among data values.

In FAST framework, we propose an adaptive sampling algorithm with feedback control. The feedback is the error between the posterior and the prior estimates from the filtering module, which is defined below.

Definition 3.7 (Feedback Error)**.** At time step k_n $(0 \leq k_n < T)$, where the subscript n indicates the n-th sampling point $(0 \leq n < M)$, we define the feedback error E_{k_n} as follows:

$$E_{k_n} = |\hat{x}_{k_n} - \hat{x}^-_{k_n}| / \max\{\hat{x}_{k_n}, \delta\} \ . \tag{3.28}$$

Note that no error is defined at a non-sampling point.

The feedback error measures how well the internal state model describes the current data dynamics, assuming $\hat{x}_{k_n}$ is close to the true state. Since $\hat{x}^-_{k_n}$ is given by a constant state model, we may infer that data is going through rapid changes when the error E_{k_n} increases. In response, the controller in our system will detect the error change and increase the sampling rate accordingly.

FAST adopts a PID controller, the most common form of feedback control [46], to measure the performance of sampling over time. We re-define the three PID components, *Proportional, Integral*, and *Derivative*, with the feedback error defined in Equation 3.28.

- **Proportional** error is to keep the controller output (Δ) in proportion to the current error E_{k_n} with k_n being the current time step and subscript n being the sampling point index. It is given by $\Delta_p = C_p E_{k_n}$ where C_p denotes the *proportional* gain which amplifies the current error.

- **Integral** error is to eliminate offset by making the change rate of control output proportional to the error. With similar terms, we define the integral control as $\Delta_i = \frac{C_i}{T_i} \sum_{j=n-T_i+1}^{n} E_{k_j}$ where C_i denotes *integral* gain amplifying the integral error and T_i represents the integral time window indicating how many recent errors are taken.

- **Derivative** error attempts to prevent large errors in the future by changing the output in proportion to the change rate of error. It is defined as $\Delta_d = C_d \frac{E_{k_n} - E_{k_{n-1}}}{k_n - k_{n-1}}$ where C_d is *derivative* gain amplifying the derivative error.

The full PID algorithm is thus

$$\Delta = C_p E_{k_n} + \frac{C_i}{T_i} \sum_{j=n-T_i+1}^{n} E_{k_j} + C_d \frac{E_{k_n} - E_{k_{n-1}}}{k_n - k_{n-1}} \ . \tag{3.29}$$

Control gains C_p, C_i, and C_d denote how much each of the *proportional*, *integral*, and *derivative* counts for the final calibrated PID error. In FAST,

Algorithm 9 Adaptive Sampling

```
Input: Current time stamp $k$, next sampling point $ns$
Output: Sampling or not
1: if $k == ns$
2:     $k$ is a sampling point
3:     afterwards, obtain feedback from filtering
4:     update $\Delta$ according to (29)
5:     calculate $I'$ according to (32)
6:     $ns \leftarrow ns + I'$, $I \leftarrow I'$
7: else
8:     $k$ is a non-sampling point
```

they are constrained by:

$$C_p, C_i, C_d \geq 0 \tag{3.30}$$

$$C_p + C_i + C_d = 1\ . \tag{3.31}$$

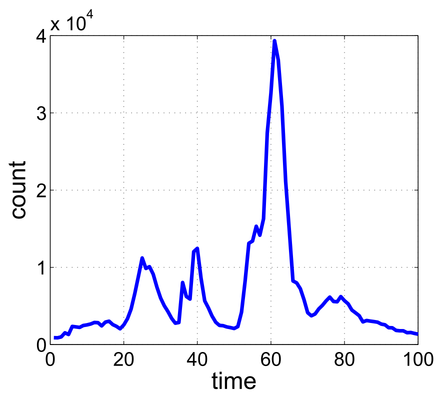
Note that setting $C_i > 0$ requires T_i previous samples in order to evaluate the integral error, which can be implemented as a straight-forward initialization prior to adaptive adjustment of the sampling rate.

Given the PID error Δ, a new sampling interval I' can be determined:

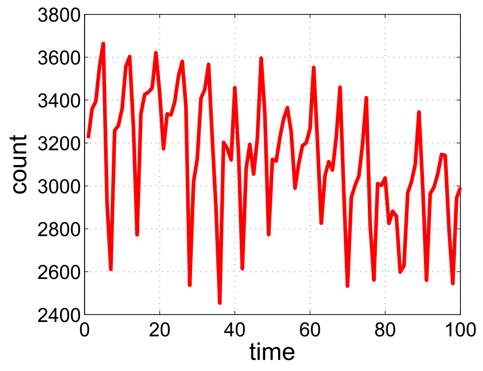
$$I' = max\{1, I + \theta(1 - e^{\frac{\Delta - \xi}{\xi}})\} \tag{3.32}$$

where θ and ξ are user-specified parameters. By default, the smallest sampling interval is set to 1. Parameter θ determines the magnitude of change and ξ is the set point for the sampling process. We assume ξ is 10% in our empirical studies, i.e. the maximum tolerance for PID error is 10%. It can be determined by FAST users according to specific applications.

Algorithm 9 summarizes the adaptive sampling algorithm used in FAST framework. It maintains and updates the variable ns indicating the next



(a) *flu* data



(b) *traffic* data

Figure 3.3: Snapshots of Datasets with Various Dynamics

sampling point. If the current time stamp is determined to be a sampling point, a feedback error can be obtained from the filtering component (Line 3) after correction. The current PID error Δ can then be evaluated (Line 4) as well as a new sampling interval I' (Line 5). A future sampling point, i.e. updated ns, is derived by applying the new sampling interval I' (Line 6). When k is a non-sampling point, the algorithm receives no feedback since only the prediction step is run in the filtering component. We will study the parameters as well as the performance of both sampling algorithms in the next section.

3.3 Experimental Results

We have implemented FAST in Java with JSC (Java Statistical Classes[1]) for simulating the statistical distributions. Our study has been conducted with synthetic as well as real-world data sets. The synthetic data sets are 1000 time stamps long and generated with $Q = 10^5$ to incorporate data value fluctuations.

[1]http://www.jsc.nildram.co.uk

- **Linear** is a synthetic series generated according to the process model in Equation 3.4.
- **Logistic** is a synthetic series generated by the logistic model $x_k = A(1+e^{-k})^{-1}$ with $A = 5000$.
- **Sinusoidal** is a synthetic series generated by a sinusoid $x_k = A\sin(bk+c)$ with $A = 5000, b = \pi/6, c = \pi/2$.

The real-world data sets are of variable length.

- **Flu** is the weekly surveillance data of Influenza-like illness provided by the Influenza Division of the Centers for Disease Control and Prevention[2]. We collected the weekly outpatient count of the age group [5-24] from 2006 to 2010. This time-series consists of 209 data points.
- **Traffic** is a daily traffic count data set for Seattle-area highway traffic monitoring and control provided by the Intelligent Transportation Systems Research Program at University of Washington[3]. We chose the traffic count at location I-5 143.62 southbound from April 2003 till October 2004. This time-series consists of 540 data points.
- **Unemploy** is the monthly unemployment level of African American women of age group [16-19] from ST. Louis Federal Reserve Bank[4]. This data set contains observations from January 1972 to October 2011 with 478 data points.

Figure 3.3 illustrates the different dynamics of the data sets. For instance, the *flu* data set has a relatively smooth curve and reflects significant changes in data value (*unemploy* data shows similar characteristics), while the *traffic*

[2]http://www.cdc.gov/flu/
[3]http://www.its.washington.edu/
[4]http://research.stlouisfed.org/

Symbol	**Description**	**Default Value**
α	Total Privacy Budget	1
Q	Process Noise	10^5
R	Gaussian Measurement Noise	10^6
N	Number of Particles	10^3
(C_p, C_i, C_d)	Control Gains	$(0.9, 0.1, 0)$
T_i	Integral Time Window	5
(θ, ξ)	Interval Adjustment Params	$(10, 0.1)$

Table 3.1: FAST Parameters

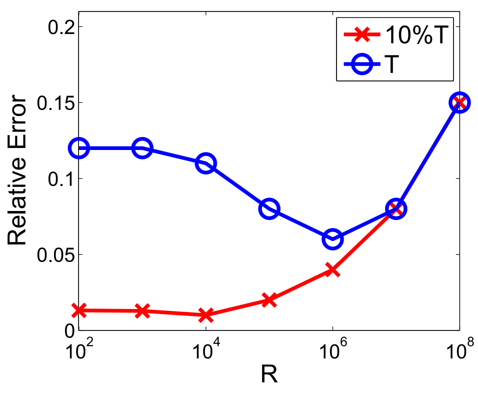
data has a less smooth curve but fluctuates around a rather stable average value.

To show the impact of FAST parameters, we will only present empirical results with the *Linear* data set for brevity. The default parameter setting, unless otherwise noted, is summarized in Table 3.3.

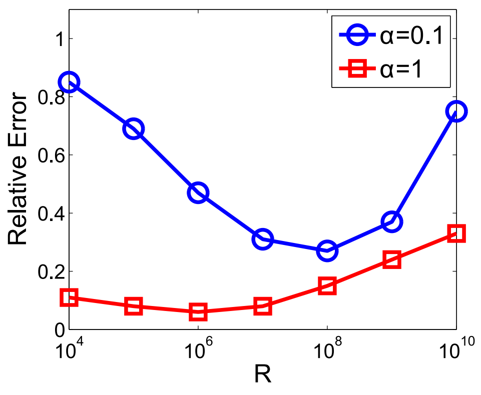
3.3.1 Effects of Filtering

Here we study the impact of parameters on filtering performance alone. Therefore, no sampling is applied in the experiments of this section, hence a posterior estimate can be derived for each time stamp.

Choice of R in the Kalman Filter. Since the process noise Q is intrinsic to the time-series data of interest, it can be determined by domain users who have good understanding about the process to monitor or have access to historic data. What is not straight-forward is the choice of R, the Gaussian measurement noise we have proposed to approximate the Laplace perturbation noise. Figure 3.4(a) plots the utility of the released time-series with various R values when using first 10% of the data series versus using the entire series. Figure 3.4(b) plots the utility with various R values when α set



(a) R vs. T

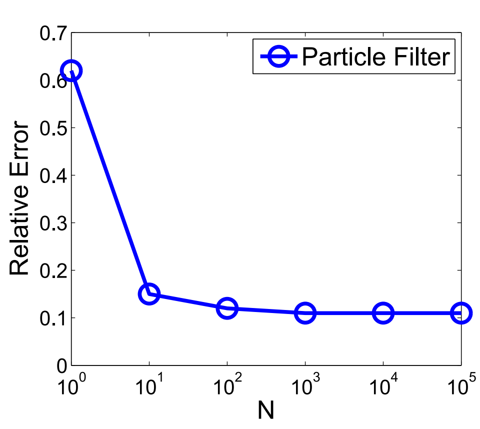


(b) R vs. α

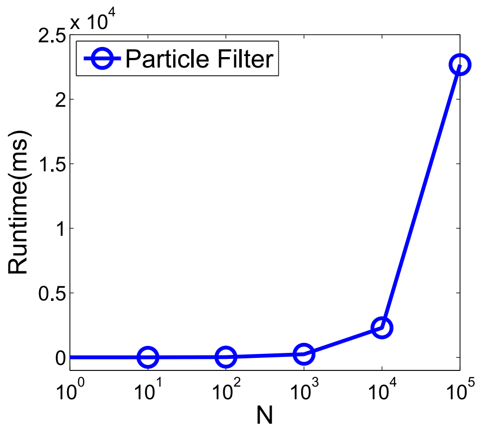
Figure 3.4: Impact of R in the Kalman Filter

to 0.1 versus 1. As can be seen in Figure 3.4(a), when using 10% of the data, the optimal R value is 10^4, as opposed to 10^6 when using the entire series. Similarly in Figure 3.4(b), when $\alpha = 1$, the optimal R value is 10^6, which is a hundred times less than 10^8, the optimal R for $\alpha = 1$. Both these results confirm our finding in Theorem 3 that the optimal R value is proportional to T^2/α^2.

Choice of N in Particle Filter. Due to the Monte Carlo nature of the particle filtering method, a larger number of particles implies more accurate estimation of the posterior distribution and more computation time. Therefore, N cannot be infinitely large for real-time applications where fast response is required. Here we examine the trade-off between accuracy and runtime of the particle filter. Figure 3.5(a) plots the utility of particle filtering with different N values. As we expect, the relative error goes down as the number of particles increases and we observe no significant boost in accuracy when N is greater than 10^3. On the other hand, a larger number of particles requires more processing time, as in Figure 3.5(b). Based on these results, we choose $N = 10^3$ as the default value as it provides a good balance between accuracy and computation efficiency.



(a) Accuracy



(b) Runtime

Figure 3.5: Impact of N in Particle Filter

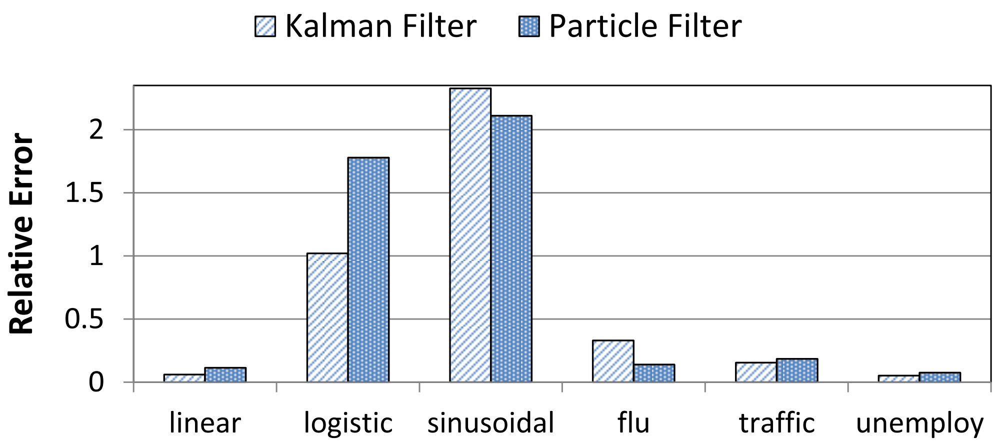
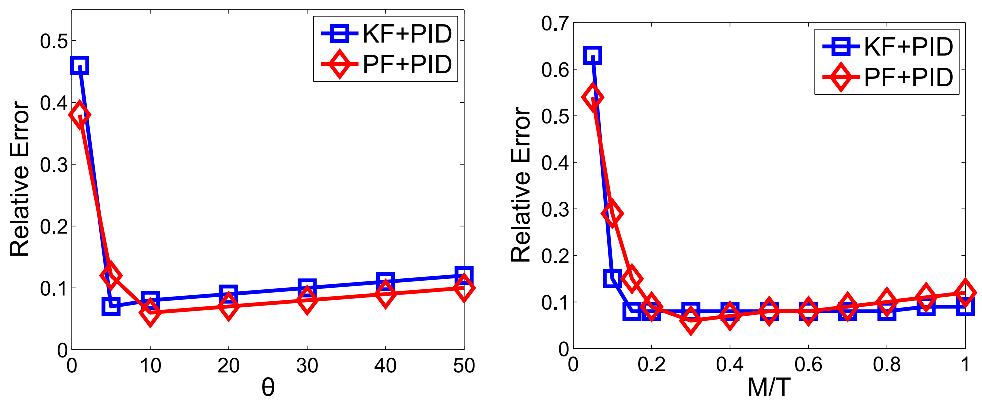


Figure 3.6: Comparison of Two Filtering Algorithms



(a) Impact of θ

(b) Impact of M

Figure 3.7: Impact of Adaptive Sampling Params

Kalman Filter vs. Particle Filter. Recall that the number of computations per time stamp for the Kalman filter is $O(1)$ and that of the particle filter is $O(N)$. Here we compare the utility of the two methods and summarize our findings in Figure 3.6. For *logistic* and *sinusoidal* data, both the Kalman filter and particle filter result in high relative error due to the non-linearity in the data. For the rest data sets, both filtering algorithms achieve high accuracy and their utility results are comparable. We conclude that particle filter requires more time and is more robust since it relies on only one parameter N which is independent of data sets, while the Kalman filter is more efficient and provides comparable accuracy on condition that the Gaussian variance R is wisely chosen.

3.3.2 Effects of Sampling

In the following studies, we apply sampling techniques on top of filtering and examine the advantage of adaptive sampling.

Parameters for Adaptive Sampling. We first study the settings of adaptive sampling parameters θ and M. Recall θ represents the magnitude of sampling interval adaptation and M represents the maximum number of samples allowed for each application. Figure 3.7(a) shows the impact of θ. Both the Kalman filter and particle filter show similar utility results and trends as θ varies. We observe that the error is prohibitive when $\theta = 1$, due to insufficient interval adjustment. In that case, the application quickly exhausts the given privacy budget. The optimal θ value for the Kalman filtering method is 5, while that of the particle filtering method is observed at $\theta = 10$. Both filtering methods result in slightly increased error as θ increases beyond the optimal point, due to enlarged sampling interval and hence a higher prediction error between two adjacent samples. However, the increase is insignificant compared to the extreme case where $\theta = 1$. Therefore, we conclude that FAST algorithms are robust to θ as we avoid apparent, extremely small values. Similarly, we state the same conclusion for the maximum number of samples M. As shown in Figure 3.7(b), we observe robust performance of FAST to the ratio M/T as long as it is not deliberately set to be $M/T < 0.1$. The optimal performance of FAST with the Kalman filter in this setting is achieved at $M = 15\%T$ and with particle filter the optimal performance is at $M = 25\%T$. We record the findings above and use them for our other empirical studies.

We also study the impact of the control gains (C_p, C_i, C_d) as well as the integral time T_i. We find that as long as the control gains are set according to the common practice: *proportional* > *integral* > *derivative*, the error variation between different settings is insignificant and is likely to be introduced by randomness. Hence we omit the detailed results and conclude that there's no extra "rule of thumb" beside the common practice for tuning the controller gains. Similarly, as the integral time increases, the resulting error shows no clear trend. We consider the above control parameters as non-influential in our system.

Fixed-Rate vs. Adaptive Sampling. We now compare the performance

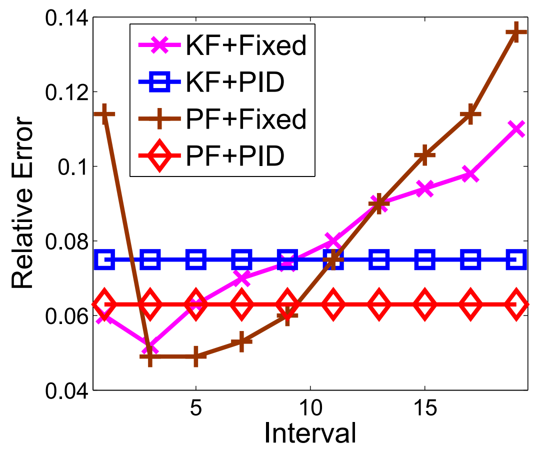
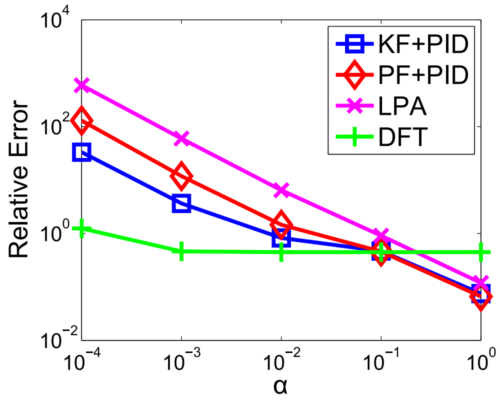


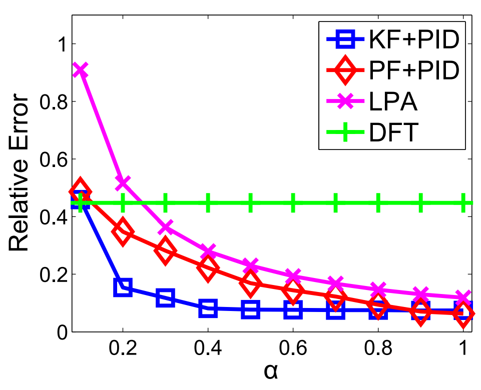
Figure 3.8: Fixed-Rate Sampling vs. Adaptive Sampling

of adaptive sampling, denoted as PID in Figure 3.8, with fixed-rate sampling, while varying the sampling interval for fixed-rate algorithm from 1 to 20. For our adaptive approach, we use the optimal M setting for each filtering method. However, there's a wide range of M to choose from, which provides equivalent level of utility, according to our previous study.

The result is shown in Figure 3.8. As the sampling interval increases, i.e. from 1 to 3, fixed-rate sampling shows reduced average relative error of various scales. This phenomenon can be interpreted by the reduction of perturbation error resulting from less frequent queries. As the interval further increases, from 5 to 20, the error starts to rise, which can be explained by the accumulation of prediction error due to longer intervals between adjacent samples. The optimal sampling interval, which is 3 and 5 for *linear* data set, may not be known *a priori* and may differ from dataset to dataset. We found that the performance of adaptive sampling with no prior knowledge is comparable to the optimal fixed-rate despite the filtering method in use, which confirms once again the advantage of FAST adaptive framework.



(a) Overall Performance

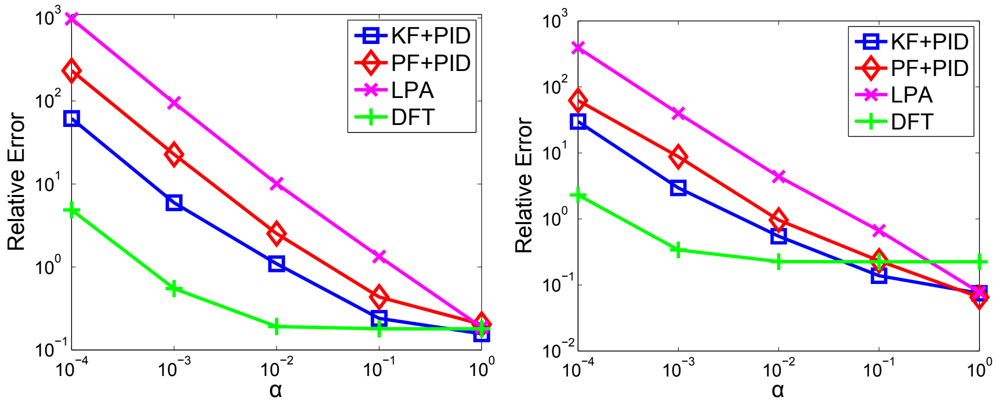


(b) Performance with Larger Budget

Figure 3.9: Utility vs. Privacy with Linear Data

3.3.3 Utility vs. Privacy

We examine the trade-off between utility and privacy in FAST, in comparison to the baseline LPA algorithm and the DFT algorithm. We note that DFT algorithm can be only applied offline and was run using the entire series in our experiments while FAST was run real-time. Figure 3.9 shows the empirical study conducted with the *linear* data set. Figure 3.9(a) plots the relative error against a wide range of privacy budget values, from 10^{-4} to 1. As we relax the privacy level and increase the privacy budget α, all four methods show reduced relative error, to different extents. We observe that the DFT algorithm with off-line processing provides highest utility when $\alpha \in [10^{-4}, 10^{-1}]$. However, no significant utility improvement can be seen when $\alpha \geq 10^{-3}$ due to its dominant reconstruction error. On the other hand, FAST algorithms, i.e. KF+PID and PF+PID, consistently outperform LPA with reduced relative error. When compared with DFT, FAST algorithms provide comparable utility and even outperform DFT when $\alpha \in [10^{-1}, 1]$. Figure 3.9(b) presents a closer look at this privacy budget interval, where FAST algorithms outperform both existing methods, providing high utility



(a) Traffic Data (b) Unemploy Data

Figure 3.10: Utility vs. Privacy with Real-World Datasets

without compromising the privacy guarantee.

In addition, we conducted the same trade-off analysis with real-world data sets in order to study the robustness of FAST framework. Figure 3.10 summarizes our findings with *traffic* and *unemploy* data. For both data sets, FAST methods greatly outperform the LPA algorithm, especially under small privacy budget, i.e. $\alpha \leq 0.1$. With larger privacy budget, i.e. $\alpha \geq 0.1$, our methods are comparable to the off-line DFT algorithm and even outperform DFT with the *unemploy* data. We observe that the result with *flu* data is similar to Figure 3.10(b). With non-linear synthetic data sets, FAST methods result in higher relative error due to model misfit but show similar overall trends as in Figure 3.10(a) in comparison to LPA and DFT. Therefore the detailed figures are omitted here for brevity. We conclude that FAST algorithms greatly improve the utility of released series over the baseline LPA method especially under small privacy cost and provide robust performance despite different data dynamics.

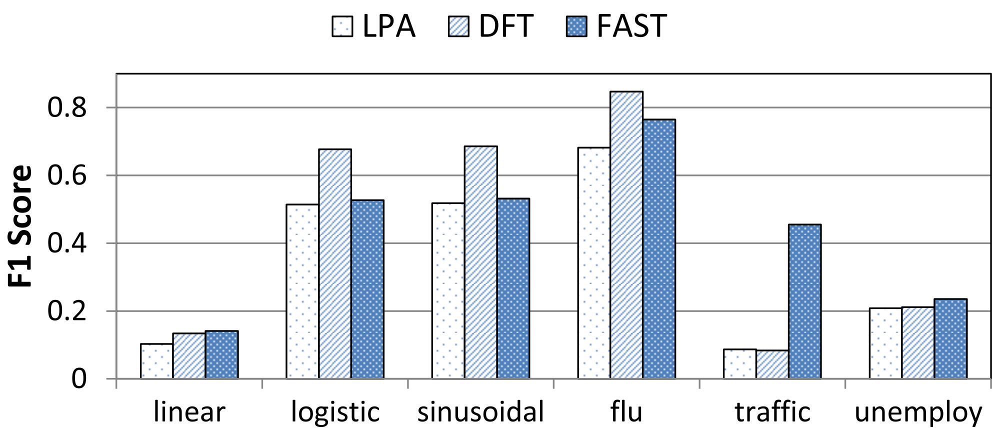


Figure 3.11: F1 Metric of Outbreak Detection

3.3.4 Detection and Correlation

In this study, we explore FAST performance with respect to utility metrics besides the standard relative error. In particular, we studied the F1 metric for outbreak detection with the released series and also performed correlation analysis between the released and original aggregate series.

There have been extensive studies on effective methods in epidemic outbreak detection and various definitions of an outbreak/signal period have been adopted [47, 57, 63]. We take a simplified interpretation of outbreak similar to Pelecanos et al. [57] and define a target event/signal as a significant increase between two adjacent aggregate values. Usually the threshold of increase can be given by users according to the application. In our empirical study, we set this threshold to be 5% of the median value in the original aggregate series, in order to mitigate the effects of extremely small or large values. Figure 3.11 compares FAST solution against existing methods and shows the F1 metric of event/signal detection in the released series across multiple data sets. We observe that FAST consistently outperforms LPA algorithm and provides comparable utility to DFT algorithm. Especially with *traffic* data, our approach provides 30% improvement over existing methods

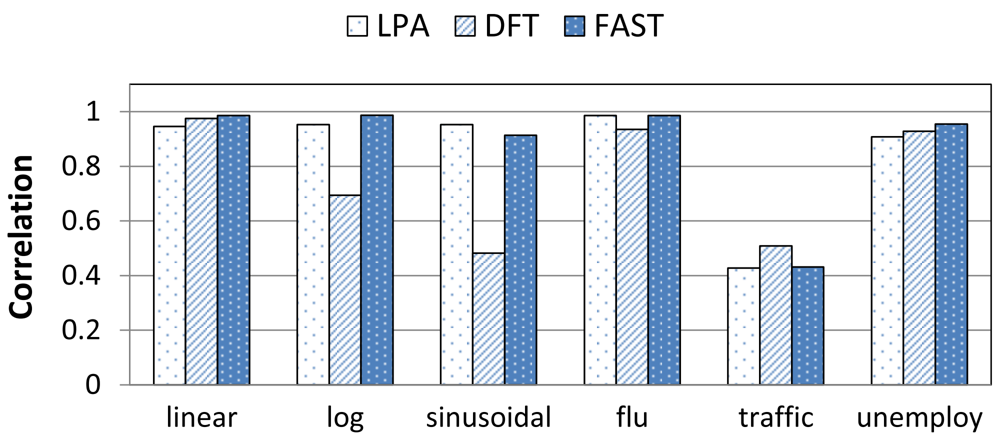


Figure 3.12: Spearman's Rank Correlation between Original Series and Released Series

due to reduced false positives.

As for correlation analysis, we measure the similarity between the released series and the original series with Spearman's rank correlation. Figure 3.12 summarizes the comparative study on FAST and existing methods. We observe that the DFT algorithm doesn't preserve the ranking order for *log* or *sinusoidal* data. In contrast, FAST algorithm provides robust performance across all data sets, even when DFT fails to produce correlated release.

To summarize, FAST yields high utility in both outbreak detection and correlation analysis compared to existing methods. In many cases, our solution outperforms both existing methods, e.g. with *traffic* data in Figure 3.11. Furthermore, FAST releases differentially private aggregates in real-time , in contrast to the offline DFT algorithm. We believe that FAST will enable a wide range of monitoring applications with the real-time feature and the adaptive strategies.

Chapter 4

Analyzing Individual Privacy Risk for Releasing Long Aggregate Series

In this chapter, we analyze individual privacy risk when FAST is applied to monitoring tasks over a long time period and demonstrate through a case study on epidemic outbreak detection. We show that by doing so we can lower the privacy perturbation while retaining the majority of individual data.

4.1 Differentially Private Anomaly Detection

In this section, we apply FAST framework to anomaly detection tasks and provide the technical details of each component through a case study of epidemic outbreak detection for syndrome surveillance.

4.1.1 Privacy Risk Definition

Dwork et al. [22] show that given function $f : D \to \mathbb{R}^d$, α-differential privacy can be achieved as follows:

$$\tilde{f}(D) = f(D) + (\nu_1, ..., \nu_d) \ . \tag{4.1}$$

ν_k are independent draws from a Laplace distribution $Lap(\frac{\Delta f}{\alpha})$ with 0 mean and the following probability density function:

$$p(x) = \frac{1}{2\frac{\Delta f}{\alpha}} e^{-|x|/\frac{\Delta f}{\alpha}} \ . \tag{4.2}$$

Δf represents the *global sensitivity* [22] of the function f. Given function $f : D \to \mathbb{R}^d$, the global sensitivity of f is defined as:

$$\Delta f = \max_{D,D'} ||f(D) - f(D')||_1 \tag{4.3}$$

where the maximum is taken over all pairs of neighboring databases.

Example 4.1 (count *Sensitivity)*. *For a count query, adding or removing an individual user from database D would change the output by at most* 1*, i.e.* $\Delta count = 1$.

In the example anomaly detection tasks, it is required to output a series of *count* queries, i.e. $f(D) = \{x_1, ..., x_T\}$, where x_k represents the number of events/occurrences during the kth time interval. The global sensitivity of f can be bounded by $\Delta f \leq T$, since adding or removing an individual would change x_k by 1 for each k, $k \in \{1, ..., T\}$.

An overview of our proposed framework for privacy-preserving anomaly detection is shown in Figure 4.1. The input to the framework is a time series of aggregates that are collected from individual users by a server. Since the server is trustworthy, such as an Emergency Department or Internet Service Provider, we assume no privacy risk at the aggregation step. At every time stamp, the raw aggregate, e.g. the number of Influenza cases or the number of incoming connections, is perturbed by the *Laplace Perturbation* module to enforce the pre-defined level of privacy guarantee. The perturbed aggregate is then received by the *Filtering* module to produce a posterior estimate. There are two internal procedures, i.e. *prediction* and *correction*, that are performed recursively at every time stamp. The posterior estimate, which

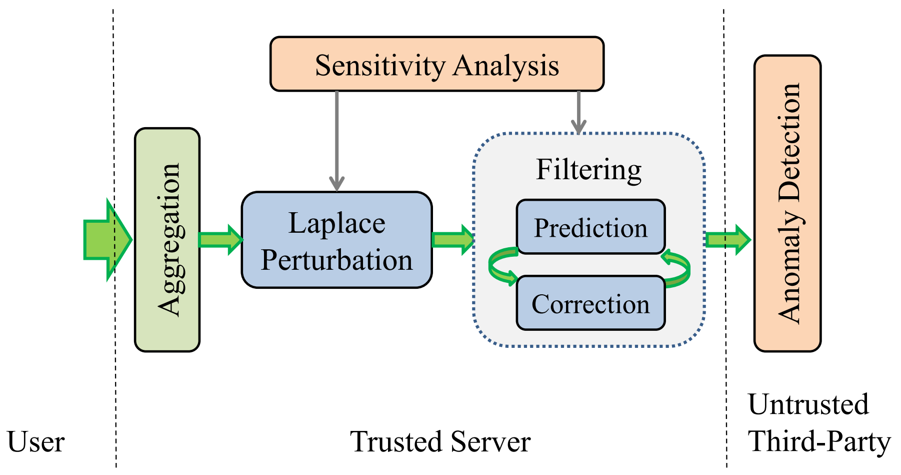


Figure 4.1: Proposed Solution for Differentially Private Anomaly Detection

is less noisy than the purely perturbed aggregate, can then be used by an anomaly detection algorithm.

On the other hand, given a real-time anomaly detection task, such as epidemic outbreak detection or network intrusion detection, a couple task-specific aspects need to be considered to set up the privacy preserving framework. Firstly, what aggregate is required at each time stamp, e.g. the *average* value of a certain attribute or a *count* of records that satisfy certain predicate? Secondly, what is the privacy principal in the application domain, an *event* or a *user*? With these questions settled, sensitivity analysis can be performed to understand the privacy risk of releasing continual aggregates given the anomaly detection task. We show a practical estimate of the global sensitivity can be derived with domain specific knowledge/statistics. The Laplace perturbation model as well as the internal models for filtering can be established correspondingly with respect to the global sensitivity. We will introduce a case study on epidemic outbreak detection and discuss each component in details in the following subsections.

4.1.2 Epidemic Outbreak Detection

The early detection of disease outbreaks has long been a concern of public health because of the potential to reduce morbidity and mortality. Patient data collected from Emergency Department/urgent-care and ambulatory-care can be aggregated, e.g. daily case counts, and shared continuously as the input to regional or national syndromic surveillance systems, such as CDC BioSense [5], EARS [33], and RODS [62]. Below we provide technical details of each component with an application to real-time epidemic outbreak detection. Note that any anomaly detection task on continual user statistics can be enabled by our proposed framework.

Most outbreak detection methods compare the current case count with a baseline period to determine whether there is an outbreak or not. We choose three extensively used, EARS [33] algorithms C1,C2, and C3 [42] for outbreak detection task. EARS algorithms were developed based on a one-sided positive CUSUM (cumulative sum) calculation. They were named according to their degree of detection sensitivity, with C1 being the least sensitive and C3 the most sensitive. The advantage of EARS algorithms is that they require limited baseline data, e.g. case counts from the previous week, compared to other methods that require that of previous years.

For C1 and C2, the test statistics on day k was calculated as

$$C_i(k) = \frac{r_k - \mu_k}{\sigma_k}, i \in \{1, 2\} \tag{4.4}$$

where r_k is the disease case count on day k, and μ_k and σ_k are the mean and standard deviation of the counts from the baseline period. For C1, the baseline data is $\{r_{k-7}, ..., r_{k-1}\}$, i.e. one week's data immediately prior the current day. For C2, the baseline is $\{r_{k-9}, ..., r_{k-3}\}$, i.e. one week's data with a two-day gap from the current day. In EARS [33], an *outbreak* is detected when $C_i(k) > 3, i \in \{1, 2\}$.

C3 algorithm uses the C2 statistics from day k and the previous two days.

The test statistic for C3 is calculated as

$$C_3(k) = \sum_{j=k}^{k-2} \max(0, C_2(j) - 1). \tag{4.5}$$

An *outbreak* is detected when $C_3(k) > 2$.

4.1.3 Sensitivity Analysis.

For outbreak detection tasks, the daily case count of a certain illness is usually the data of interest. In most existing syndrome surveillance systems, this data is collected daily from the Emergency Department for early detection. Below we analyze the global sensitivity for monitoring the daily count of Influenza cases over T days, where T is a pre-defined project time line, e.g. the number of days over 10 years. Note that for other anomaly detection applications, the sensitivity analysis can be conducted in a similar way with domain-specific statistics/knowledge.

Let D be the patient database and $f(D)$ outputs a sequence of counts $\{x_1, ..., x_T\}$, where x_k represents the number of Influenza cases on day k. As discussed in Section III, the global sensitivity of $f(D)$ can be set $\Delta f = T$, which is the length of the surveillance period, roughly 3650 for 10 years' time line. Although theoretically sound, there are two drawbacks to adopting the upper bound of Δf. First of all, it is an impractical estimate and will almost never happen in reality. Recall that the global sensitivity defines the maximal contribution of any individual to the function output. It is very rare that anyone would visit the emergency room and be diagnosed with Influenza for every day in 10 year's period. Secondly, when T is large, the Laplace perturbation error introduced at every time stamp has a variance proportional to T^2, according to Equation 4.2. The released data with such high perturbation error would be practically useless for detection purposes.

Below we quantify the rareness of $\Delta f = T$ with publicly available survey statistics and provide a practical, smaller valued estimate of Δf. Again, the value of Δf represents the number of times that an individual visits emergency room and is diagnosed with Influenza in 10 years' period. According to the National Hospital Ambulatory Medical Care Survey: 2010 Emergency Department Summary Tables[1], the number of emergency room visits per 100 persons in 2010 is 42.8 and 3.2% of visits are diagnosed with "Acute upper respiratory infections, excluding pharyngitis", which includes Influenza. We note that young children, under 1 years old, have a higher rate of emergency room visit and other age groups have similar rates. Therefore, we derive the probability of a patient visiting emergency room and being diagnosed with Influenza every year, p, as follows:

$$p \approx 42.8\% \times 3.2\% = 1.4\% \ . \tag{4.6}$$

Let n denote the number of a patient visiting emergency room and being diagnosed with Influenza in 10 years' period. Given Equation 4.6, we obtain the following:

$$P(n \leq 2) = \sum_{k=0}^{2} P(n = k) = \sum_{k=0}^{2} \binom{10}{k} p^k (1-p)^{10-k} = 99.9\% \tag{4.7}$$

which indicates that with 99.9% confidence, any individual patient will contribute to $f(D)$ at most twice in every 10 years.

4.1.4 Laplace Perturbation

The Laplace Perturbation component ensures differential privacy by perturbing the input aggregate at every time stamp. Let $f(D)$ outputs a sequence of counts $\{x_1, ..., x_T\}$ on data set D, where x_k represents the number of events of interest during time interval k. Given the global sensitivity Δf and the

[1]http://www.cdc.gov/nchs/ahcd/web_tables.htm

desired level of privacy α, we can derive a private count sequence $\tilde{f}(D)$ that satisfies α-differential privacy as follows:

$$\tilde{f}(D) = \{z_k = x_k + \nu_k | k = 1, ..., T\} \tag{4.8}$$

where ν_k are independent draws from $Lap(\frac{\Delta f}{\alpha})$. The default value for Δf is T. When a practical estimate can be derived from domain-specific statistics, we can set Δf with a much smaller value, i.e. $\Delta f = 2$ as in the analysis above for Influenza outbreak detection. This perturbation model in Equation 4.8 is used in our proposed framework.

A natural concern arises when setting $\Delta f < T$: how about those individuals that are counted more than Δf times in the released data? As the sensitivity analysis above shows, only few patients, i.e. 0.1%, will contribute to the time series for more than two times in 10 years. In reality, those are patients who are more susceptible to Influenza or who have more frequent emergency room visit history, and thus can choose to opt out at the data collection stage without significantly affecting the quality and quantity of collected data. Below we assume that the database consists of the majority patients, i.e. 99.9% patients who contribute to the data series at most 2 times over 10 years, and can be protected by differential privacy.

4.1.5 Filtering

The Filtering component in our framework utilizes time series modeling and estimation algorithms to improve the accuracy of released aggregates. The Kalman filter based estimation algorithm is adopted and is shown to be computationally efficient [29]. We briefly show the state-space model for time series data as well as the filtering algorithms used in our framework.

For a time series of *count* queries, i.e. $\{x_k\}$, we establish the following models:

$$x_k = x_{k-1} + \omega \ , \tag{4.9}$$

$$\omega \sim \mathcal{N}(0, Q) \ . \tag{4.10}$$

The noisy observation, which is obtained from the Laplace Perturbation mechanism, can be represented as follows:

$$z_k = x_k + \nu_k \ , \tag{4.11}$$

$$\nu_k \sim Lap(\Delta f/\alpha) \tag{4.12}$$

We adopt the following Gaussian approximation:

$$\nu_k \sim \mathcal{N}(0, R) \ , \quad R \propto (\Delta f)^2/\alpha^2 \tag{4.13}$$

which has been shown in [29] to be computationally efficient and to minimize posterior estimation error.

We outline the Filtering procedure in Algorithm 10. It consists of two recursive operations: *prediction* and *correction*. The detailed definitions and derivations as well as *Prediction* and *Correction* implementations are omitted here for brevity, as they can be found in [28, 29].

It is shown in [28] that releasing $\{\hat{x}_k\}$ rather than $\{z_k\}$ greatly improves the accuracy of the shared aggregates. We will empirically evaluate the utility of shared data in the context of outbreak detection in the next section.

4.2 Experimental Results

We implemented the proposed framework as well as three outbreak detection algorithms, i.e. C1, C2, and C3, in Java. All experiments were conducted using a 2.90 GHz Intel Core i7 PC with 8GB RAM. We used simulated data sets provided by CDC EARS[2], all simulating 6 years of daily count data. We

[2]http://www.bt.cdc.gov/surveillance/ears/datasets.asp

Algorithm 10 Filtering

Input: Noisy measurements **Z**= $\{z_k\}$

Output: Released series **R**= $\{r_k\}$

```
1: for each $k$ do
2:     Prediction: $\hat{x}_k^- = r_{k-1}$
3:     $z_k \leftarrow$ Laplace Perturbation
4:     Correction: $\hat{x}_k = \hat{x}_k^- + K_k(z_k - \hat{x}_k^-)$
5:     $r_k \leftarrow \hat{x}_k$
```

Set	**Mean**	**Standard Dev.**	**Trend**	**Seasonality**
$s01$	90.2	33.3	Yes	Mild-None
$s02$	29.9	5.6	No	Medium
$s03$	1.19	5.76	No	Mild-None
$s04$	6	4.3	Yes	Very
$s07$	150	26.635	No	Mild-None
$s11$	301.1	78.8	Yes	Medium

Table 4.1: Simulated Outbreak Detection Data Sets

chose 6 data sets, i.e. `s01-04`, `s07`, and `s11`, with different characteristics to evaluate our proposed solution. The descriptions of the 6 data sets are listed in Table 4.2. From each data set, 10 iterations were used in the evaluation and the average utility results are reported.

Unless specified, the default parameter settings are as follows: $\alpha = 1$, $R \propto (\Delta f)^2/\alpha^2$, $Q[s01] = 100$, $Q[s02] = 100$, $Q[s03] = 0.001$, $Q[s04] = 1$, $Q[s07] = 100$, $Q[s11] = 100$. Note that the Q values are commonly derived by offline tuning and our settings may not be optimal.

4.2.1 Accuracy of Released Data

We first study the trade-off between privacy and accuracy in our proposed framework, in comparison to the baseline Laplace Perturbation algorithm (LPA) with default sensitivity $\Delta f = T$ and practical sensitivity $\Delta f = 2$. The LPA algorithms apply perturbation at every time stamp, and release perturbed values, i.e. $\{z_k\}$ as in Equation 4.11. In contrast, our solution adopts a tighter sensitivity bound, i.e. $\Delta f = 2$, and releases the posterior estimates, i.e. $\{\hat{x}_k\}$ as in Algorithm 10. We measure the accuracy of the released time series by average relative error, as proposed in [29]. We plotted the trade-off curves with data set `s04` in Figure 4.2 and note that other data sets have similar trends.

As we increase the privacy budget α from 0.001 to 1, all methods show improvement in accuracy of released data series, due to the reduced perturbation noise. As can be seen, methods that adopt practical sensitivity outperform baseline `LPA` with default sensitivity, reducing the relative error by several orders of magnitude. Moreover, our proposed solution constantly outperforms LPA($\Delta f = 2$). We conclude that FAST filtering techniques can improve the accuracy of released data on top of already reduced perturbation error.

We further examine the private, released series by our solution and compare it with the original `s04` data set. The released data series was generated with $\alpha = 1$. As is shown in Figure 4.3, the data series released by our privacy-preserving framework closely follows the original data line at all time stamps. This shows that our framework provides highly accurate data release while providing differential privacy guarantee.

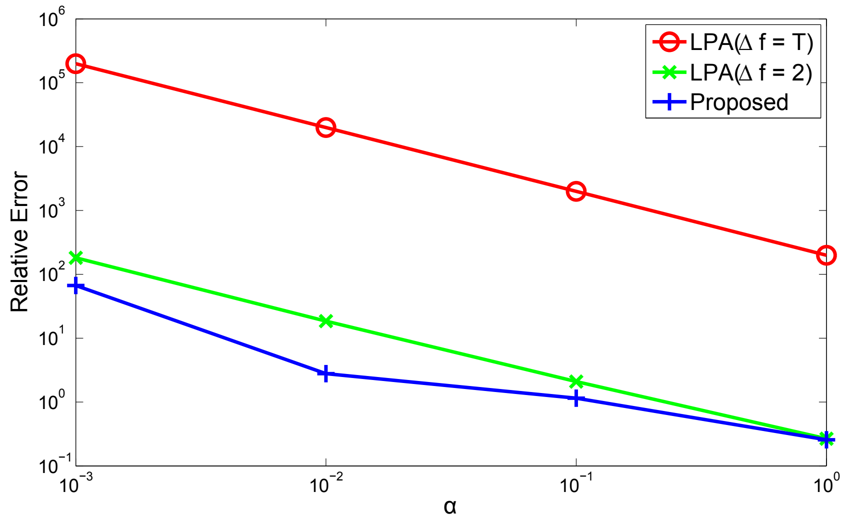


Figure 4.2: Privacy vs. Accuracy with `s04` Data Set

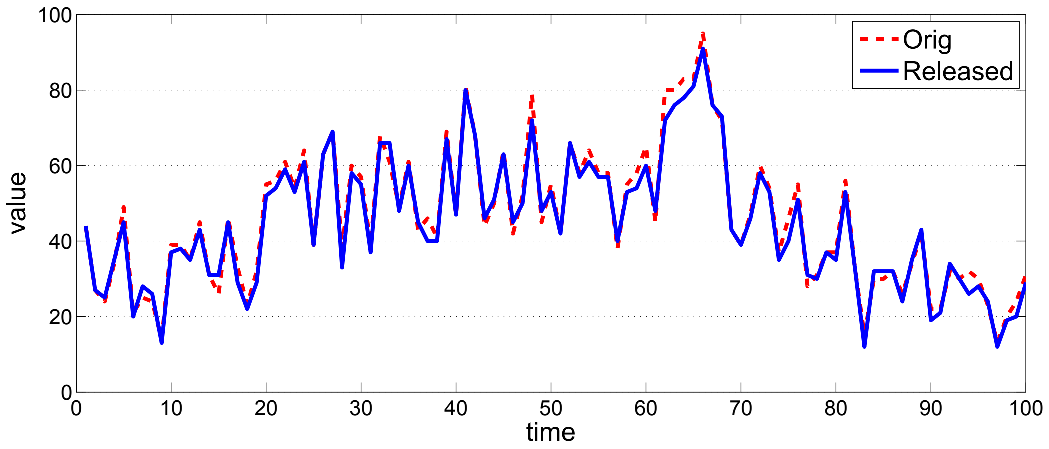


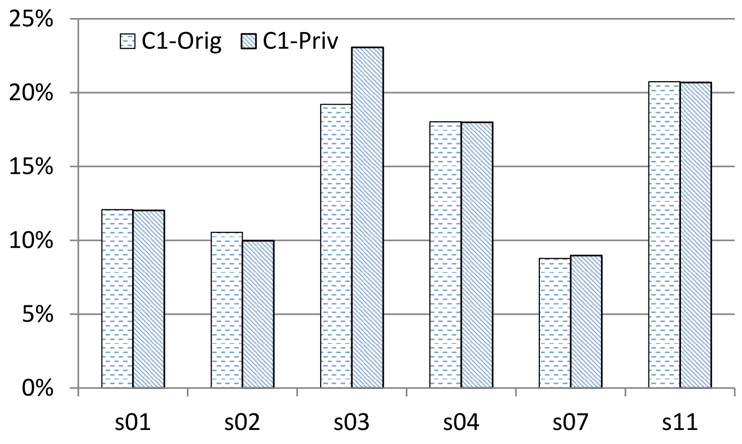
Figure 4.3: Original vs. Released Series with `s04` Data Set

4.2.2 Outbreak Detection Evaluation

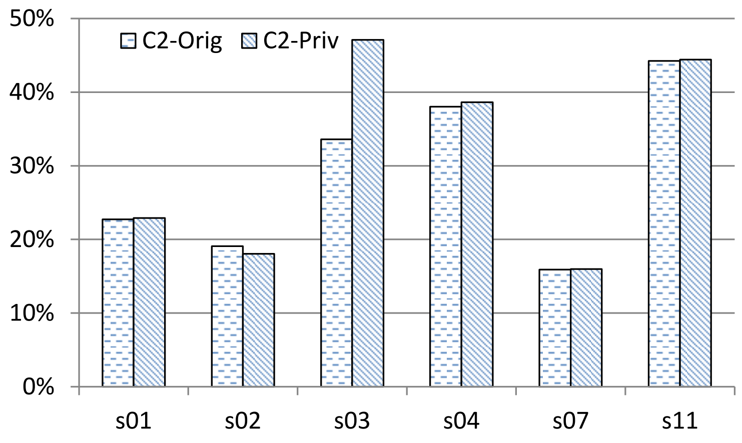
We study the usefulness of the private, released data series provided by our solution with EARS outbreak detection algorithms. We generated released data from all six data sets with the privacy parameter $\alpha = 1$ and ran C1, C2, and C3 on both original data sets and private, released data sets. For C1, C2, and C3 algorithms, we set $r_k = x_k$ if detection is performed on original data with no privacy preservation and $r_k = \hat{x}_k$ if using our privacy-preserving framework. Same rules apply to the baseline counts.

Two common data mining metrics, i.e. *Sensitivity* and *Specificity*, are studied. Sensitivity, also referred to as recall, measures an algorithm's ability to identify positive results, while specificity relates to an algorithm's ability to identify negative results. A *true positive* is a correctly identified outbreak day while a *true negative* is a correctly identified non-outbreak day. Note that the ground truth is provided along with the simulated data sets. We summarize the utility results in Figure 4.4 and Figure 4.5.

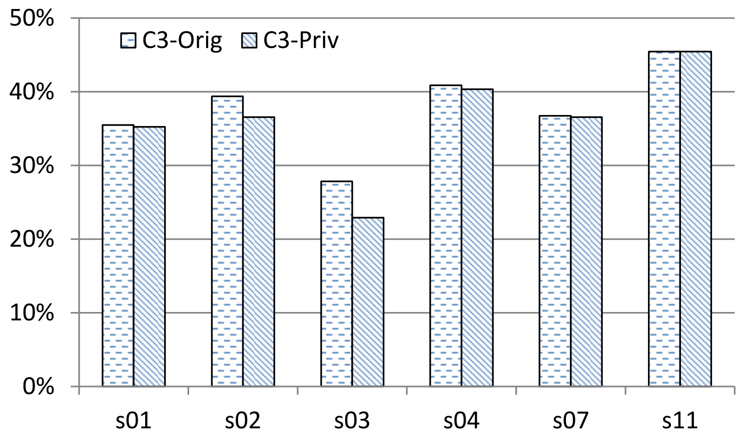
As can be seen, using the data released by our privacy-preserving solution does not significantly degrade the performance of any outbreak detection algorithm. For instance in Figure 4.4(c), C3 achieves 35.48% sensitivity with the original s01 data set, and 35.25% with the private released data. We notice that with data set s03, where the original data values exhibit relatively high variation and no clear trend, using the private released data incurs more detected positives, i.e. higher sensitivity and lower specificity. We interpret this phenomenon as a result of model misfit, since the process model for the underlying time series is constant. As a future work direction, in-depth study about the dynamics of real data sets can be performed in order to establish an accurate internal model in the Filtering component.



(a) C1 Algorithm

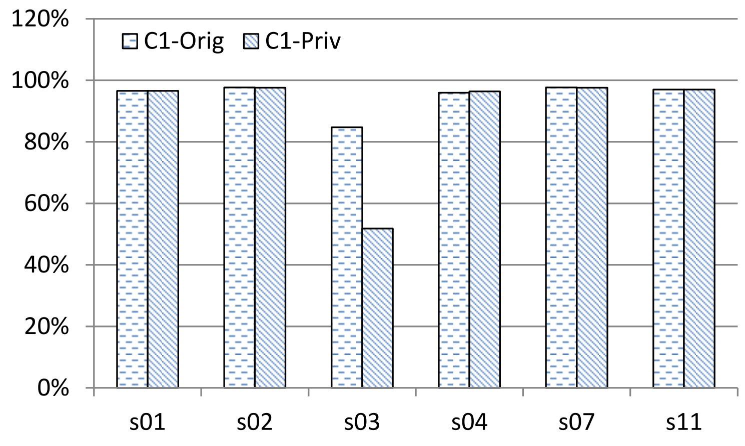


(b) C2 Algorithm

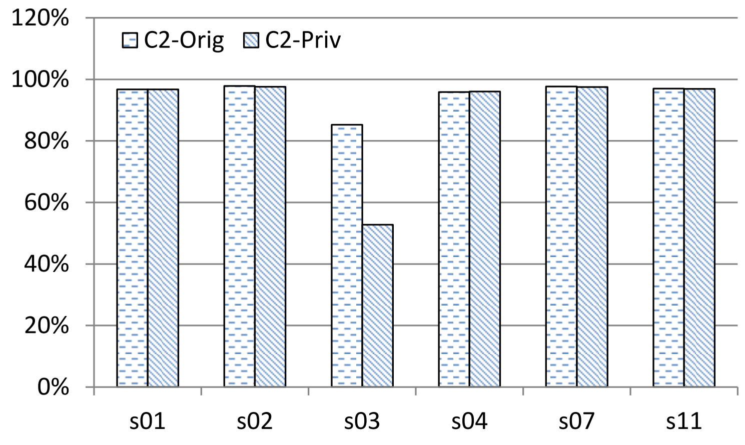


(c) C3 Algorithm

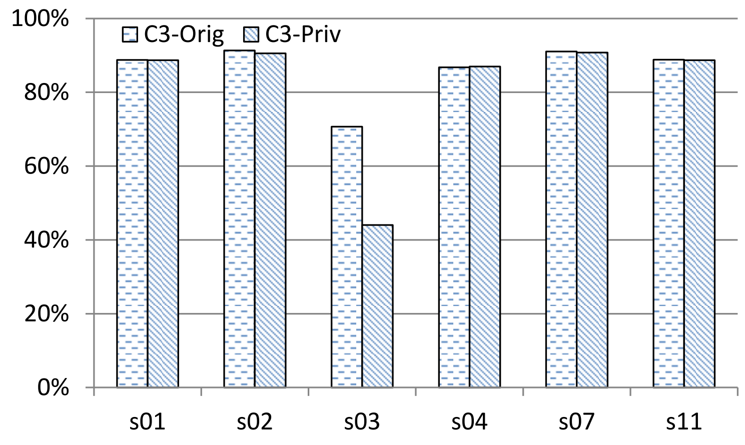
Figure 4.4: Sensitivity Performance with Original vs. Private Data



(a) C1 Algorithm



(b) C2 Algorithm



(c) C3 Algorithm

Figure 4.5: Specificity Performance with Original vs. Private Data

Chapter 5

Modeling Spatio-Temporal Correlation of Multi-Variate Aggregate Time Series

In this chapter, we address the problem of modeling spatio-temporal correlation between multiple aggregates when releasing multi-variate aggregate time series. We first explore the possibility of incorporating individual behavioral patterns into FAST process model for web mining tasks. We then investigate complex spatio-temporal correlations, i.e. traffic histogram over a road network, and propose to model the spatial correlation based on the background road network.

5.1 Differentially Private Web Monitoring

5.1.1 Problem Statement

Here we formally define the problem of monitoring web browsing activities with differential privacy. A browsing session is defined as a sequence of web pages browsed at consecutive, discrete time stamps, which can be obtained from the log file of page requests [1]. We consider all browsing sessions are es-

[1]Between two successive requests, the user is assumed to browse the previous page

Session	t_1	t_2	t_3	t_4	t_5	...
s_1	fp	fp				
s_2		fp	news	sports news		
s_3		tech	news	news	local	...
s_4			on-air	health	local	...
...	...	...	...	...	...	...

Table 5.1: Examples of Individual browsing sessions

tablished on a single server, where no data integration is needed across different servers. Furthermore, the sessions are dynamically established, possibly starting at different time points with variable lengths.

Let D denote the database of browsing sessions (illustrated in Table 5.1) and T denote the expected length of monitoring period. Without loss of generality, we assume that there are m web pages, i.e. $page_1, page_2, \ldots, page_m$, hosted on the server. An example of a browsing session of length 3 is s_2 in Table 5.1. In this session, the user starts from the front page ("fp" for short) at time t_2, then successively navigates to the news page at time t_3, and finally at time t_4 browses the sports news page and the session terminates there. The goal of our work is to release the number of sessions in D browsing $page_i$ at time k for each i and k, without disclosing the presence or absence of any private session. A formal definition of our problem is provided below.

Problem 5.1 (Private Web Monitoring)**.** *Let x_k^i denote the number of sessions in D that browse the i-th page at time stamp k, $1 \leq k \leq T$. For every time stamp k, we are to release a sanitized count r_k^i for every i, such that the series of private releases $\{\{r_k^i\}_{i=1}^m,$ for $k = 1, \ldots, T\}$ satisfies α-differential privacy.*

We further limit user browsing activities by introducing an additional parameter l_{max}, which indicates the maximum session length allowed in our problem setting. We assume $l_{max} < T$. Our consideration is three-fold: **1**) In practice, a typical browsing session would not contain unlimited number of web pages. For example, the average session length is 4.7 from data collected on `msnbc.com`, shown in Table 5.3. **2**) It is very common for web applications to specify a fixed time-out period such that the session automatically ends if the user does not refresh or request a page within that period. **3**) From privacy preservation point of view, if a session could contain an unbounded number of web pages, it would have an unlimited influence on the released aggregate values. As a consequence, the differential privacy mechanism, described below, would have to inject a large perturbation in order to mitigate such an influence. Based on these considerations, we set $l_{max} = 20$ later in our empirical studies.

Privacy Definition. The following lemma establishes the sensitivity of counting web page visits in data set D for T time stamps, in order to protect the privacy of each individual session.

Lemma 5.2 (Sensitivity for Counting Page Visits)**.** *Let D be the session database with maximum session length l_{max}. Then for a query $q(D) = \{\{x_k^i\}_{i=1}^m$, for $k = 1, \ldots, T\}$ which computes the number of sessions in D browsing $page_i$ at time k for every i and k, the sensitivity $GS(q)$ of q is at most l_{max}.*

Proof. For any session s in D, at most one page $page_i$ is visited by s at time k. Furthermore due to the session length constraint, over the entire time frame T, the session s can visit at most l_{max} pages (either the same page for multiple times or different web pages). Hence, by adding or removing any session s from the dataset D, we would change at most l_{max} counts of $q(D)$ output by one. Therefore, it follows that the sensitivity $GD(q)$ of q is at most l_{max}. $\square$

5.2 Proposed Solution

The goal of our work is to enable the data holders to share useful aggregates for web monitoring applications, while preserving privacy. There are three key components to achieving this goal. First of all, the privacy of individual sessions should be protected with strong guarantee. Our solutions adopt differential privacy to protect every session, despite possible background knowledge known to the adversary. Secondly, the differential privacy mechanism injects perturbation noise into every released aggregate. Therefore, a post-processing, i.e. estimation, component is used in our system in order to recover the perturbed values and to improve the utility of released aggregates. We propose two approaches to estimate the aggregates for each web page separately or all web pages simultaneously. Thirdly, the dynamics of the underlying aggregate time series should be utilized in order to perform accurate estimation. Therefore, we propose to learn the state-space models with publicly available data, which can come from historical data or users who consent disclosure. We will demonstrate that our solutions learn accurate models and achieve high utility in the released data, even with a small training set.

The system framework of monitoring web browsing activities with differential privacy is depicted in Figure 5.6. At every time stamp, the statistics of browsing sessions can be obtained by aggregating the **private user data**. The raw aggregates will then be perturbed by the **Laplace Perturbation** mechanism to ensure the pre-defined level of differential privacy. The perturbed aggregates are then used by the **Estimation** module to derive posterior estimates of the original aggregates. Based on which approach is in use, the posterior estimate can be derived for each web page separately or for all web pages simultaneously. The state-space model used in posterior estimation can be learned through the **Training** component with publicly

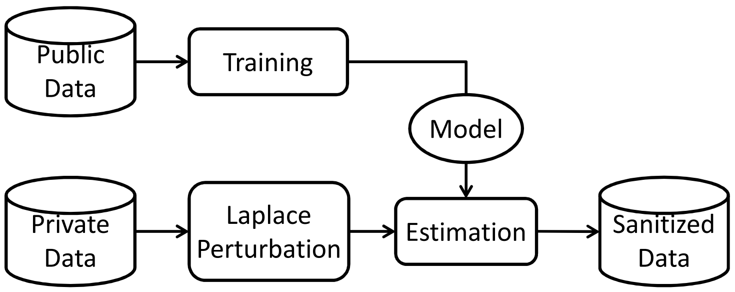


Figure 5.1: Proposed Solution for Differentially Private Web Monitoring

available data. Based on the method in use, we can learn a state-space model for each activity separately or an all-in-one state-space **model** for all web pages. The **sanitized output data** is the posterior estimates released in real-time, which are statistically more accurate than the purely perturbed values. The detailed algorithms are described below.

5.2.1 Univariate Time Series Approach

The first solution to monitoring web activities with differential privacy is to establish a univariate state-space model for the *count* series of each web page and estimate the true states from the perturbed values. We adapt our previously proposed FAST framework [28, 29] to provide session-level privacy and apply the Kalman filter based estimation algorithm to each univariate *count* series.

For each web page i, we establish the following process model for the time series $\{x_k^i, \text{for } k = 1, \ldots, T\}$:

$$x_{k+1}^i = x_k^i + \omega_k^i \tag{5.1}$$

$$\omega_k^i \sim \mathbb{N}(0, Q^i) \tag{5.2}$$

Similarly, the measurement model for the Laplace perturbed value z_k^i is es-

Algorithm 11 Univariate Time-Series Algorithm(k)

Input: Raw counts $\{x_k^i, \text{for } i = 1, \ldots, m\}$, privacy budget α

Output: Private, released counts $\{r_k^i, \text{for } i = 1, \ldots, m\}$

```
1: for $i = 1, \ldots, m$, do
2:     $prior \leftarrow$ Prediction($i, k$)
3:     $z_k^i \leftarrow$ perturb $x_k^i$ by $Lap(\frac{l_{max}}{\alpha})$
4:     $posterior \leftarrow$ Correction($i, k$)
5:     $r_k^i \leftarrow posterior$
```

tablished below:

$$z_k^i = x_k^i + \nu_k^i \tag{5.3}$$

$$\nu_k^i \sim Lap(0, \frac{l_{max}}{\alpha}) \tag{5.4}$$

Note that since ν_k^i is added to each page i in parallel at every time stamp k, it follows the same distribution regardless of i and k.

In this work, we adopt the approximation in Equation 5.5 in order to utilize computational attractive structure of the Kalman filter for posterior estimation.

$$\nu_k^i \sim \mathbb{N}(0, R) \tag{5.5}$$

Estimation algorithm. Below we outline the recursive Kalman filter mechanism for our proposed solution based on univariate time series models. The detailed procedures for prediction and correction can be found in [26].

The overall solution based on univariate time series state-space model is summarized in Algorithm 11. At every time stamp k, for each web page i, a prior estimate is derived from the prediction procedure according to the process model, while a posterior estimate is derived by combining the prediction and the noisy observation in the correction procedure. At the first time stamp, i.e. $k = 1$, the perturbed value z_1^i is released for initialization. The advantage of the univariate solution is its efficiency in computing the minimum variance estimate for linear Gaussian problems. As can be seen, the required computation time is linear of the number of web pages, i.e. $\mathcal{O}(m)$, per time stamp. The process can be easily parallelized, since each web page is modeled and processed separately. We will further study the run time performance of this approach in the experiment section.

5.2.2 Multivariate Time Series Approach

As web browsing sequences exhibit strong navigation patterns between adjacent web page requests, it has been shown that user navigation patterns can be well captured by first-order Markov chain [10]. In order to incorporate this rich spatio-temporal correlation, we propose the following method which establishes a multi-variate time series model and at each time stamp releases the counting histograms of all web pages at once.

The multivariate time series process model which utilizes the navigation Markov chain is described below:

$$\mathbf{X_{k+1}} = \mathbf{M}\mathbf{X_k} + \omega_\mathbf{k} \tag{5.6}$$

$$\omega_\mathbf{k} \sim \mathbb{N}(\mathbf{0}, \mathbf{Q}) \tag{5.7}$$

Each state in the multivariate time series is a vector, i.e. $\mathbf{X_k} = (x_k^1, \ldots, x_k^m)^\intercal$. The linear coefficient $\mathbf{M}$ is represented as a m-by-m Markov transition matrix and is defined as follows:

$$\mathbf{M} = \begin{pmatrix} p_{1,1} & p_{1,2} & \cdots \\ p_{2,1} & p_{2,2} & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix} \tag{5.8}$$

where each element $p_{i,j}$ represents the probability of transitioning to the *ith* page from the *jth* page.

The process noise is also a vector, i.e. $\omega_{\mathbf{k}} = (\omega_k^1, \ldots, \omega_k^m)^\intercal$, where ω_k^i represents the process noise of the ith page at time stamp k. By assuming each ω_k^i follows a white, time-invariant Gaussian distribution and all ω_k^i's are mutually independent, we can derive that $\omega_{\mathbf{k}}$ is also white Gaussian and its covariance matrix $\mathbf{Q}$ is the diagonal, i.e.

$$\mathbf{Q} = \begin{pmatrix} Q^{1,1} & 0 & \ldots & 0 \\ 0 & Q^{2,2} & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & Q^{m,m} \end{pmatrix} \tag{5.9}$$

where each $Q^{i,i}$ is a positive scalar value that represents the variance of process noise ω_k^i in the multivariate model. Note that $Q^{i,i}$ might be different from Q^i as in the univariate model, since the multivariate model captures the spatio-temporal correlation of multiple aggregates.

Similarly, the multivariate measurement model is established as follows:

$$\mathbf{Z_k} = \mathbf{X_k} + \nu_{\mathbf{k}} \tag{5.10}$$

$$\nu_{\mathbf{k}} \sim \mathbb{N}(\mathbf{0}, \mathbf{R}) \tag{5.11}$$

where $\mathbf{Z_k}$ is the vector of perturbed values at time k, i.e. $\mathbf{Z_k} = (z_k^1, \ldots, z_k^m)^\intercal$. $\nu_{\mathbf{k}}$ stands for the vector of independent perturbation noise, i.e. $\nu_{\mathbf{k}} = (\nu_k^1, \ldots, \nu_k^m)^\intercal$, where each noise follows the time-invariant Laplace distribution $Lap(0, \frac{l_{max}}{\alpha})$. For efficiency, we approximately model $\nu_{\mathbf{k}}$ as a white Gaussian noise with covariance matrix $\mathbf{R}$.

The perturbation noises are added independently to web pages. Therefore, the covariance matrix $\mathbf{R}$ is also diagonal:

$$\mathbf{R} = \begin{pmatrix} R^{1,1} & 0 & \ldots & 0 \\ 0 & R^{2,2} & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & R^{m,m} \end{pmatrix} \tag{5.12}$$

Algorithm 12 Multivariate Time-Series Algorithm(k)

Input: Raw counts $\{x_k^i, \text{for } i = 1, \ldots, m\}$, privacy budget α

Output: Private, released counts $\{r_k^i, \text{for } i = 1, \ldots, m\}$

```
## Prediction
1: $\mathbf{\hat{X}_k^-} = \mathbf{M\hat{X}_{k-1}}$
2: $\mathbf{P_k^-} = \mathbf{MP_{k-1}M^\intercal} + \mathbf{Q}$
## Perturbation
3: $\mathbf{Z_k} \leftarrow$ perturb $\mathbf{X_k}$ by $Lap(\frac{l_{max}}{\alpha})^m$
## Correction
4: $\mathbf{K_k} = \mathbf{P_k^-}(\mathbf{P_k^-} + \mathbf{R})^{-1}$
5: $\mathbf{\hat{X}_k} = \mathbf{\hat{X}_k^-} + \mathbf{K_k}(\mathbf{Z_k} - \mathbf{\hat{X}_k^-})$
6: $\mathbf{P_k} = (\mathbf{I} - \mathbf{K_k})\mathbf{P_k^-}$
7: $\{r_k^i\} \leftarrow \mathbf{\hat{X}_k}$
```

where each $R^{i,i}$ is a positive scalar value that represents the variance of measurement noise ν_k^i in the multivariate model.

Estimation algorithm. The estimation algorithm based on multivariate time-series model is summarized in Algorithm 12. As can been seen, similar computation structure based on the Kalman filter is adopted. At every time stamp k, a posterior estimate $\mathbf{\hat{X}_k}$ is released for monitoring applications, which contains the estimated counts at time k for each web page.

From the computational aspect, the multivariate time series approach includes four matrix-matrix multiplications as well as one matrix inversion at every time stamp. The computation complexity at every time stamp is therefore $\mathcal{O}(m^3)$, due to the matrix computations. We will empirically study its runtime performance in the experiment section.

Parameters. In the Markov matrix $\mathbf{M}$, each entry $p_{i,j}$ represents the probability of transitioning to the ith page from the jth page. We propose to estimate each $p_{i,j}$ as follows:

$$p_{i,j} = \frac{\#(page_j, page_i)}{\#page_j} \tag{5.13}$$

where $\#page_j$ is the number of occurrences of page j and $\#(page_j, page_i)$ is the number of occurrences of page i immediately following page j. The number of occurrences can be obtained by counting support from the browsing sequences in the training set.

The noise covariance matrix $\mathbf{Q}$ can be also learned by off-line tuning. Unlike the univariate approach, the tuning of covariance matrix $\mathbf{Q}$ has an exponential search space, since we need to simultaneously set all diagonal elements. We adopt a similar approach to Yan et. al [69] which utilizes the computation structure of genetic algorithm (GA) for parameter optimization. The details are provided in the experiment section.

As for the approximate measurement noise $\mathbf{R}$, since the actual measurement noise is only determined by the Laplace perturbation mechanism, we set $R^{i,i} = R$ for every i as in the univariate model in our experiments.

Privacy. The privacy guarantee of multivariate algorithm is stated in the following theorem.

Theorem 5.3 (Privacy Guarantee)**.** *Algorithm 12 satisfies α-differential privacy.*

Proof. Similar to the proof of Theorem 2. Omitted. □

5.3 Experimental Results

Here we present a set of empirical studies conducted a simulation of dynamic web browsing behavior generated from real-world data. In each study, we compare the following four methods: 1) `U-KF`, i.e. our univariate Kalman filter approach as an extension of the FAST framework [28], 2) `M-KF`, i.e.

Symbol	Description	Default Value
α	Total Privacy Budget	1
R	Gaussian Noise Variance	$40,000$
m	Number of Web Pages	18
T	Monitoring Time	100
l_{max}	Max Session Length	20

Table 5.2: Parameters for Multi-Variate Aggregate Series Release

our multivariate Kalman filter approach which incorporates the rich spatio-temporal correlation in the process model, 3) `LPA`, i.e. the baseline method that releases the Laplace perturbed value at every time stamp, which is applied to each univariate times series, and 4) `DFT`, i.e. the Fourier transformation based algorithm [59], applied to each univariate time series separately in an off-line manner.

The default settings of parameters required by the above methods are shown in Table 5.3, except for the process noise parameters Q^i and $Q^{i,i}$ for every i and the markov transition matrix $\mathbf{M}$, which can be learned from the training data. Note that the number of web pages m is 18, which includes all the actual web pages from `MSNBC` data set and an inactive status "\$" introduced by us on purpose. We preserve $d = 20$ Fourier coefficients for the `DFT` method, as suggested by the authors [59].

5.3.1 Simulating Dynamic Browsing Sessions

We believe that empirical evaluations should be conducted in a practical setting in order to demonstrate the usability of proposed methods in solving real problems. In the absence of raw log files with the finest page and time granularity, we propose to simulate the dynamic browsing behavior with the Poisson process and anonymous, real-world session data.

Dataset	MSNBC
sessions	989,818
categories	17
longest session length	14,975
average session length	4.7

Table 5.3: MSNBC Dataset Characteristics

As the data pool for our simulation, we consider the MSNBC anonymous web dataset at the UCI Machine Learning Repository [2]. The MSNBC data, summarized in Table 5.3, contains nearly 1 million anonymous browsing sessions collected over a period of twenty-four hours on the msnbc.com domain. All web pages were classified into 17 categories, and each session in MSNBC data set records a sequence of category requests with variable length.

In our simulated data set, we consider a time frame of $T = 100$ time stamps, where at each time stamp a number of new sessions start and some existing ones may end. At time $t = 1$ we start by randomly sampling $S_{start} = 100000$ sessions from the candidate set. Successively at every new time stamp, we randomly sampled S_{new} sessions from the candidate set, where S_{new} is a random variable drawn from a Poisson distribution with mean 10000. This choice is motivated by the fact that the Poisson process has been commonly used in modeling user page request rate for web browsing [15, 50] and for video-on-demand systems [70]. In particular, we use the same methodology as in Yu et al. [70], where S_{new} is upper-bounded by $N = 20000$ which represents the maximum number of new sessions that the server can handle at any time. For every session drawn from the candidate set, prior to adding to our simulated data set, it is first truncated if needed to contain up to l_{max} web pages. Then it is padded with a special symbol "$", which indicates being inactive, at the beginning as well as in the end, such that the total number of

[2]http://archive.ics.uci.edu/ml/datasets/MSNBC.com+Anonymous+Web+Data

symbols is T. For each session starting at time stamp k, $k-1$ \$'s are added at the beginning. A proper number of \$'s are added to the end of each session based on the actual session length. As a result, we generated $1,089,281$ dynamic browsing sessions, where the start of each session is indicated by the first non-\$ symbol in the sequence.

To estimate model parameters for our proposed methods, we created a training data set with a small percentage, i.e. 5%, randomly sampled from the simulated data set. For evaluation purpose, we randomly sample 100 test data sets $\{D_1, D_2, ..., D_{100}\}$, each containing 10% of the simulated data set. Average results obtained from $\{D_1, D_2, ..., D_{100}\}$ are reported in our evaluations.

5.3.2 Learning Models

Below we describe how to estimate the model parameters from the training data. As we introduced \$ in simulating the browsing behavior to indicate the inactive states, we treat \$ as a web page and learn the process noise for its count series as well as the transition probabilities for our proposed methods.

For the univariate approach, we can learn the dynamics of the count series for each web page i, i.e. the process noise variance Q^i, from the training data set. For each web page i, we aggregated its count series from the training set and tuned Q^i to minimize the posterior estimate error. Since each Q^i is a real value and implies a large search space, we specified a search domain comparing only different orders of magnitude, i.e. $\{10^{-4}, 10^{-3}, \ldots, 10^{9}\}$, in order to speed up the training process. For every i and each setting of Q^i, we ran the univariate approach 50 times to overcome the randomness of the perturbation noise and the value which resulted in minimum average relative error, as defined in Equation 5.14, for posterior estimates was preserved for real-time monitoring.

For the multivariate approach, we can learn the transition probabilities $p_{i,j}$ for any web page pair i and j from the training set, as in Equation 5.13. Note that we can also learn the transition probability from the inactive status \$ to any web page, which indicates the likelihood of starting a new session and which web page it starts from. As for the noise covariance matrix $\mathbf{Q}$, the same search domain as above was specified for each element $Q^{i,i}$. We implemented the genetic algorithm (GA) with random initial solutions and the population size 50, and ran it for 20 iterations. The fitness value is defined as the average relative error (over 50 runs) of the posterior estimates generated by the multivariate approach, compared to the raw count series aggregated from the training set. The best solution generated by GA algorithm was preserved for real-time monitoring.

5.3.3 Utility Evaluation

The goal of our work is to share useful statistics of on-line browsing behavior in order to perform monitoring tasks. We compare the utility of data released by our proposed methods, i.e. `U-KF` and `M-KF`, against existing approaches, i.e. `LPA` and `DFT`. Note that the `DFT` is an off-line method and therefore cannot be applied to real-time monitoring tasks. It is only included in our experiments for reference and comparison.

We adopt three different utility metrics in the following studies, including both generic metrics as well as application-specific metrics. For each set of evaluations, we further study the trade-off between utility and privacy for each method by varying the privacy budget, i.e α value. The usual range of α adopted by most other works in differential privacy is between 0.1 an 1. However, we choose a larger range in our experiments including smaller α values, i.e. 0.01 and 0.05, to demonstrate the applicability of all four methods when the privacy requirement is high.

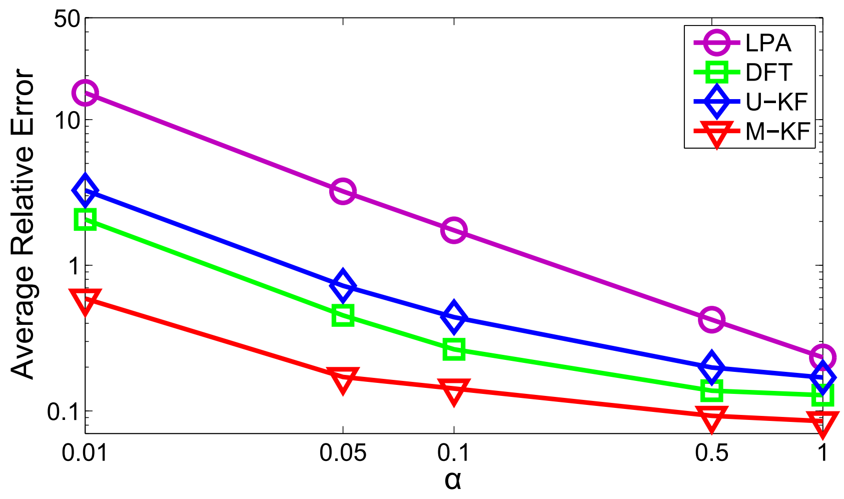


Figure 5.2: Privacy vs. Average Relative Error

Average Relative Error

In the first set of empirical evaluations, we consider to measure the Average Relative Error (ARE) between the released count series and the original count series. In short, ARE is a widely used metric which measures how well the released time series $\{r_k^i\}$ follows the original series $\{x_k^i\}$ for every $i = 1, \ldots, m$. It is a generic metric to evaluate data accuracy, disregarding the actual, domain-specific applications. More formally, we define the ARE error as follows:

$$ARE = \frac{1}{mT} \sum_{i=1}^{m} \sum_{k=1}^{T} \frac{|r_k^i - x_k^i|}{\max\{x_k^i, \delta\}} \tag{5.14}$$

where $\delta = 1$ by default, in order to handle the special case when x_k^i is zero.

As in the above definition, the ARE value provides an indication about the quality of the overall released time series, where smaller values of ARE imply higher similarity between the released and the original series, hence higher utility. We ran all four methods under different privacy budgets and the corresponding ARE values are reported in Figure 5.2. We observe that for every approach the ARE drops as the privacy budget increases. This is due to reduced perturbation error introduced by the differential privacy mechanism.

The baseline approach `LPA` which directly releases perturbed values results in the highest ARE error in every privacy setting, due to the perturbation noise. With relatively strong privacy requirement, i.e. $\alpha = 0.01$, the `LPA` algorithm results in large relative error which is more than 10 times higher than that of our proposed method `M-KF`. The `U-KF` method and the `DFT` method show similar results for all privacy settings. However, the former releases aggregates in real-time, while the latter requires an off-line processing of the time series due to the Fourier transform. Our proposed algorithm `M-KF` turns out to be superior and constantly outperforms all other methods, resulting in the lowest ARE error with real-time release of private data. `M-KF` yields to 59% error when $\alpha = 0.01$ and 8% error when $\alpha = 1$, while `DFT` results in more than 200% error when $\alpha = 0.01$ and 13% error when $\alpha = 1$.

Top-K Mining

A fundamental application of monitoring web browsing behavior is top-K mining, which aims to find the K most popular web pages visited at every time stamp. Therefore, the ability to preserve the most popular pages in the private, released data values is an important indicator of the solution usability. In the next set of experiments, we perform top-K mining at every time stamp on the released data by all the methods and report the Average Precision (AP) over the entire monitoring time period. We define the average precision as follows:

$$AP = \frac{1}{T}\sum_{k=1}^{T} TPR_k \tag{5.15}$$

where TPR_k represents the true positive rate of the top-K pages discovered from the private, released data at time k. Apparently, a higher value of

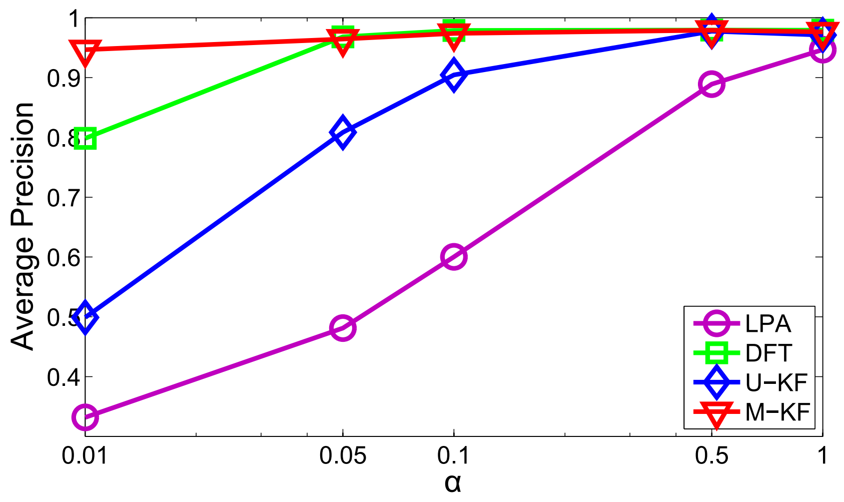


Figure 5.3: Privacy vs. Precision for Top-K Mining

AP indicates higher utility, since it reflects the capability for more accurate discovery of most visited web pages at any time stamp. We ran all four methods with different privacy budget values and plot the average precision for mining top-5 web pages in Figure 5.3. Similar trends can be observed when experimenting with different K values. Thus we omit those results here for brevity.

For all four methods, the average precision is raised as the privacy budget α increases. The baseline `LPA` again offers the worst mining utility in the shared data in every privacy setting, preserving only 33% of top-5 web pages when α is small, i.e.$\alpha = 0.01$, due to the random perturbation. Our univariate approach `U-KF` falls behind the off-line method `DFT` and the multivariate approach `M-KF` until the privacy budget is large enough, i.e. $\alpha \geq 0.5$, due to the individual state-space modeling for each web page. However, we observe that `U-KF` is still applicable and it yields 80% precision with $\alpha = 0.05$. The off-line `DFT` method yields 80% precision when $\alpha = 0.01$ and provides comparable utility to our multivariate approach `M-KF` when $\alpha \geq 0.05$. Again the `M-KF` method is proved to be superior to all the other methods, providing

95% precision in top-K mining even under very small privacy budget, i.e. $\alpha = 0.01$.

Distributional Similarity

In this set of experiments, we consider the count values at every time stamp, i.e. $\{x_k^i, \text{for } i = 1, \ldots, m\}$ and $\{r_k^i, \text{for } i = 1, \ldots, m\}$, as distributions of user sessions over the domain of web pages and propose to evaluate the distributional similarity between the released counts and the original counts at all time stamps. The intuition behind is that the session distribution over the domain of web pages enables understanding of the relative popularity of any web page at any time stamp. Thus measuring the distributional similarity would provide a comprehensive view of the utility of data released over the entire web page domain.

A common metric widely used to measure the distance between two probability distributions is the KL-divergence. It is a non-symmetric measure that computes the information lost when a proposed distribution is used to estimate a true distribution. In our scenario, the estimate distribution at time stamp k comes from the released data values $\{r_k^i, \text{for } i = 1, \ldots, m\}$, while the true distribution comes from the original count values $\{x_k^i, \text{for } i = 1, \ldots, m\}$. We normalized the data values at every time stamp and denote the corresponding distributions as $\{\tilde{x}_k^i\}$ and $\{\tilde{r}_k^i\}$. Therefore, the average KL-divergence of the released time series $\mathbf{R}$ with respect to the original data series $\mathbf{X}$ can be defined as follows:

$$D_{KL}(\mathbf{X} \| \mathbf{R}) = \frac{1}{T} \sum_{k=1}^{T} \sum_{i=1}^{m} \ln\left(\frac{\tilde{x}_k^i}{\tilde{r}_k^i}\right) \tilde{x}_k^i \tag{5.16}$$

It reports the average KL-divergence of the released distributions $\{\tilde{r}_k^i\}$ with respect to the true distribution $\{\tilde{x}_k^i\}$ at all time stamps. Intuitively, the smaller the average KL-divergence is, the more similar the released distributions are to the original distributions.

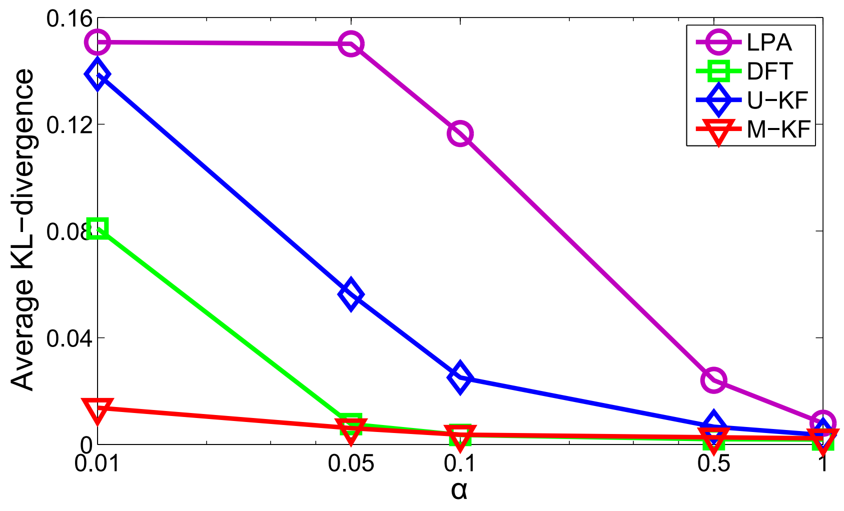


Figure 5.4: Privacy vs. Distributional Similarity

All four methods were run under different privacy settings and the average KL-divergence results are shown in Figure 5.4. The baseline `LPA` yields highest KL-divergence among all methods in every privacy setting. Due to the randomness of the perturbation noise, the data released by `LPA` fails to preserve the distributional similarity with respect to the original data. Our univariate approach `U-KF` does not show clear advantage over `LPA` when $\alpha = 0.01$, due to the separate modeling of each web page. However, it can be seen that `U-KF` quickly catches up with `DFT` and the multivariate approach `M-KF` when $\alpha \geq 0.1$. Again, the `M-KF` method provides the best utility in every privacy setting, preserving the distributional properties in the private, released data values even when the privacy budget is small, i.e. $\alpha = 0.01$. We can see that `M-KF` yields a four times smaller divergence compared to the off-line `DFT` method when $\alpha = 0.01$, thanks to its accurate, multivariate model.

5.3.4 Runtime

An important aspect of the methods that release data for monitoring tasks is the processing time of shared data. Here we empirically compare all four methods in terms of the total running time for releasing private data values over T time stamps. Note that we exclude the training time for learning the model parameters since it is a one-time cost and usually done off-line. In addition, the processing time for each time stamp can be easily estimated from the total running time.

All four methods were run under the default parameter settings. Their testing time (training time excluded) was recorded and the average runtime in milliseconds over all test data sets is reported in Figure 5.12. As can be seen, the off-line method `DFT`, with overall complexity $\mathcal{O}(mT^2)$ [3], turns out to be the most expensive compared to other real-time methods. We observe that `DFT` takes 23 milliseconds to release aggregated data for $m = 18$ web pages over $T = 100$ time stamps. Our multivariate approach `M-KF`, which requires matrix multiplications, additions, and inversions, has overall complexity $\mathcal{O}(m^3T)$. As is shown, `M-KF` only takes 4 milliseconds, one sixth of `DFT` runtime, to release the same amount of data. We believe `M-KF` is highly applicable in our problem setting, especially when $m \ll T$. Both our univariate approach `U-KF` and the baseline `LPA` have linear complexity, i.e. $\mathcal{O}(mT)$. As in the empirical results, both `LPA` and `U-KF` take in-significant amount of time which is measured as zero. We conclude that compared to the baseline `LPA`, our `U-KF` method achieves good amount of utility improvement with no additional computational cost, while our `M--KF` method greatly improves utility with moderate additional computational cost. We believe that our proposed methods can be applied to sharing private statistics in real-time, without compromising the outcome of web monitoring applications.

[3]In general, the Discrete Fourier Transform requires $\mathcal{O}(T^2)$ complex multiplications and additions for a time series of length T.

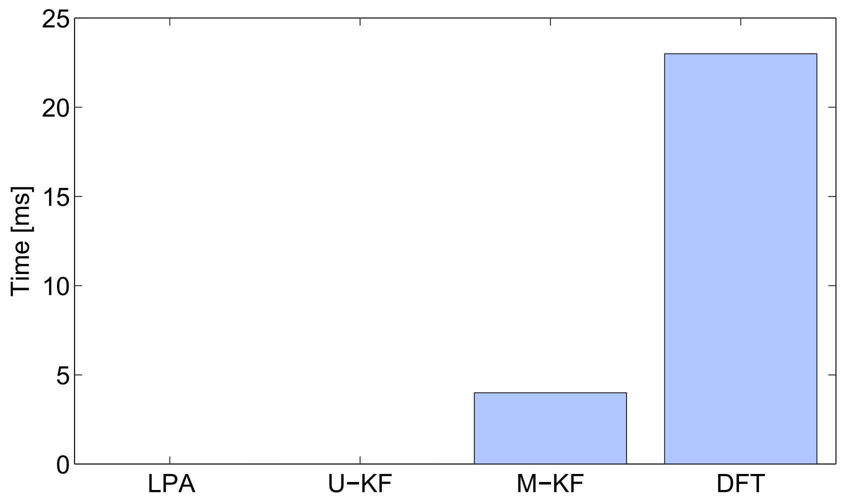


Figure 5.5: Runtime Performances for Web Monitoring

5.4 Differentially Private Traffic Monitoring

5.4.1 Problem Statement

In the traffic monitoring application we consider, a set of objects are moving in a two-dimensional space and a central server is collecting information about their locations over time. We adopt a fine-grained 2D grid that partitions the space G into $w \times w$ cells, where w is a constant number called *resolution*. We further assume the expected collection time span is T and denote k as the discrete time index where $0 \leq k < T$. For each cell c in G, we define the *frequency series* of c as $\mathbf{X}^c = \{x_k^c | \ 0 \leq k < T\}$, where x_k^c represents the number of objects within its extent at time stamp k. A multi-dimensional time series $\mathbf{X}^G$ can be defined as the set of frequency series of every cell c in G, i.e. $\mathbf{X}^G = \{\mathbf{X}^c | \ c \in G\}$. A *snapshot* of the spatio-temporal database $\mathbf{X}_k^G$ is defined as the set of cell frequencies at time k, i.e. $\mathbf{X}_k^G = \{x_k^c | \ c \in G\}$. The same terms for the released data set $\mathbf{R}^G$ can be defined similarly.

Problem 5.4. *Given a multi-dimensional time series* $\mathbf{X}^G$ *where* $G = w \times w$

Algorithm 13 Laplace Perturbation Algorithm(LPA)

Input: Raw data series $\mathbf{X}^G$, privacy budget α

Output: Released data series $\mathbf{R}^G$

```
1: for each cell $c \in G$ do
2:     for each time stamp $k$ do
3:         $r_k^c \leftarrow$ perturb $x_k^c$ by $Lap(0, \frac{T}{\alpha})$;
```

cells, for each snapshot $\mathbf{X}_k^G$*, release in real-time a sanitized version* $\mathbf{R}_k^G$ *such that the overall release* $\mathbf{R}^G$ *satisfies* α*-differential privacy, where* α *is a user-specified privacy level.*

Note that sharing $\mathbf{R}^G$ will enable a variety of data mining tasks. Therefore we use a generic utility metric, i.e. relative error, to measure the usefulness of the released series for each cell c:

Definition 5.5 (Utility Metric)**.** The utility of a published series $\mathbf{R}^c = \{r_k^c\}$ can be measured by the *average relative error*, denoted as E^c, against the original time-series $\mathbf{X}^c = \{x_k^c\}$.

$$E^c = \frac{1}{T}\sum_{k=0}^{T-1} \frac{|r_k^c - x_k^c|}{max\{x_k^c, \delta\}} \tag{5.17}$$

where δ is a user-specified constant (also referred to as *sanitary bound* as in [65]) to mitigate the effect of excessively small query results, e.g. $0's$. Here we set $\delta = 1$ throughout the entire time-series for all cells.

5.4.2 Baseline Solution - LPA Algorithm

A baseline solution to sharing differentially private multi-dimensional time series is to apply the standard Laplace perturbation at each time stamp to

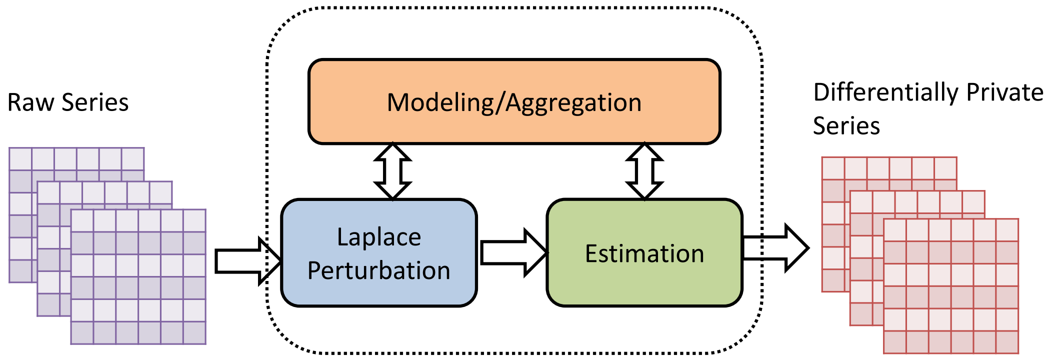


Figure 5.6: Proposed Solution for Differentially Private Traffic Monitoring

every frequency series. For any c, if every released aggregate satisfies α/T-differential privacy, by Theorem 1 the released frequency series guarantees α-differential privacy. We summarize the baseline algorithm in Algorithm 13 and Line 3 represents the Laplace mechanism to guarantee α/T-differential privacy for each released aggregate. Empirical studies of the LPA algorithm against our proposed solutions are included in Section 4.

5.5 Proposed Solutions

In this section, we present our proposed solutions for privacy-preserving traffic monitoring. Figure 5.6 provides a high-level overview of the system framework. At every time stamp, the input multi-dimensional data is perturbed by the *Laplace Perturbation* mechanism to guarantee differential privacy. Then the perturbed data can be post-processed by the *Estimation* module to produce a more accurate, released version. Domain knowledge, such as road network and population density, is utilized by *Modeling/Aggregation*, which in return interacts with the perturbation component as well as the estimation method in use. Below we describe in detail two separate estimation algorithms: one is to perform time-wise estimation for each individual cell, while the other is to perform spatial aggregation and estimation over the entire 2D space.

5.5.1 Temporal Estimation

For each cell c in space G, we can apply FAST framework [28, 29] to the cell frequency series $\mathbf{X}^c$. We model the dynamics of different types of cells according to the domain knowledge on the road networks. Below we briefly show how to model the cell frequency series and refer to our work [28] for further implementation details.

Note that the internal model of cell frequencies depends on many factors, such as location, overall population, road network, etc. Here we simply classify each cell as *sparse* or *dense* based on road network connections and assume the same internal model for cells within each category. For each cell c, its frequency series $\mathbf{X^c}$ can be represented by the following process model:

$$x_{k+1}^c = x_k^c + \omega^c \ , \quad p(\omega_c) \sim \mathbb{N}(0, Q^c) \tag{5.18}$$

Q^c value indicates the level of variation between adjacent time stamps. Intuitively, sparse cells exhibit little variation since very few objects travel within them, therefore we should specify a small Q^c value for such cells. On the other hand, higher Q^c should be assigned for dense cells since they are visited more frequently in reality.

The noisy observation, which is obtained from the Laplace Perturbation mechanism, can be modeled as follows:

$$z_k^c = x_k^c + \nu \ , \quad \nu \sim Lap(0, 1/\alpha_0) \tag{5.19}$$

The differential privacy budget for each traffic count is $\alpha_0 = \alpha/T$, since the overall privacy budget α is uniformly allocated to each time stamp.

Algorithm 14 Temporal Estimation Algorithm

Input: Raw data series $\mathbf{X}^G$, privacy budget α

Output: Released data series $\mathbf{R}^G$

```
1: for each timestamp $k$ do
2:     for each cell $c \in G$ do
3:         $prior \leftarrow c.\mathbf{Predict}(k)$ ;
4:         $z_k^c \leftarrow$ perturb $x_k^c$ by $Lap(0, \frac{T}{\alpha})$;
5:         $posterior \leftarrow c.\mathbf{Correct}(k, prior, z_k^c)$;
6:         $r_k^c \leftarrow posterior$;
```

We adopt the following Gaussian approximation for every cell and use the Kalman filter based filtering technique [28] for posterior estimation.

$$\nu \sim \mathbb{N}(0, R) \; . \tag{5.20}$$

The outline of the temporal estimation algorithm is presented in Algorithm 14. The advantage of temporal estimation approach is that it utilizes the internal time series model and the observations to form an educated guess, which is shown in [28] to greatly improve the accuracy of released data per time stamp. As for complexity, we can see that the computation time requirement is $O(w^2)$ for every time stamp where w is the spatial resolution, since only $O(1)$ operations are performed for each cell.

5.5.2 Spatial Estimation

When every cell is perturbed individually, data sparsity imposes great utility challenge, i.e. high relative error due to perturbation. We thus are motivated to group similar cells to overcome the data sparsity issue. Considering the spatial correlation among cells, it is very likely that neighboring cells are connected by the same roads therefore are more similar to each other. To

Algorithm 15 QuadTreeAgg Algorithm

Input: 2D grid G, depth threshold d

Output: QuadTree index structure QT

```
1: $QT.root \leftarrow G$;
2: $queue$.add($QT.root$) ;
3: while ! $queue$.empty() do
4:     $node \leftarrow queue$.remove() ;
5:     if ! $node$.homogeneous() and $node.depth < d$
6:         $node$.split() ;
7:         $queue$.add($node.children$) ;
```

utilize this heuristic, we propose to aggregate similar cells into partitions according to spatial vicinity and perform estimation within each partition assuming uniformly distributed objects within the partitions.

We propose a top-down space partitioning approach based on Quadtree due to several considerations. One advantage of Quadtree is its efficiency: it recursively partitions a 2D space into 4 quadrants disregard the actual object distribution in the space. Another advantage of Quadtree is that it doesn't incur any extra privacy cost due to its independence from data. In contrast, the kdTree structure proposed by Cormode et al [16] does require extra privacy budget spent on finding the "private median". Since the privacy budget for each time stamp is very limited, we believe that Quadtree is more suitable in the multi-dimensional time series scenario.

We outline the spatial aggregation algorithm based on Quadtree in Algorithm 15. Line 5 checks every node/partition for the splitting condition. Line 6 splits a partition into four equal quadrants. The *node*.homogeneous() method returns true if all the cells within the partition belong to the same category. Again, each cell is pre-classified as *sparse* or *dense* based on domain

Algorithm 16 Spatial Estimation Algorithm

Input: Raw data series $\mathbf{X}^G$, depth threshold d, privacy budget α

Output: Released data series $\mathbf{R}^G$

```
1: $QT \leftarrow$ QuadTreeAgg$(G, d)$; # initialize the quadtree index
2: for each timestamp $k$ do
3:     for each partition $p \in QT$ do
4:         $p_k \leftarrow \sum_{c \in p} x_k^c$ ;
5:         $\tilde{p}_k \leftarrow$ perturb $p_k$ by $Lap(0, \frac{T}{\alpha})$;
6:         $r_k^c \leftarrow \tilde{p}_k / p.size(), c \in p$ ;
```

knowledge. We stop splitting a partition if it is homogeneous. Otherwise, as long as the predefined depth threshold d is not violated, we further split the partition in the hope of reducing the class impurity in each child partition. The value of d represents the aggregation level. Setting $d = 0$ implies that all cells are aggregated in one partition. Since the uniform assumption within the partition does not hold, high estimation error will be incurred. On the other hand, a higher value of d implies that many partitions will be further split to produce homogeneous regions so as to reduce estimation error. However, due to data sparsity, very few moving objects will fall into each partition when it is small. Therefore, the perturbation error will dominate the released data in that case. Clearly the optimal d value depends on the spatial distribution of cells. We will examine the impact of d in the experiment section.

Once the Quadtree index structure of the space G is established, we assume uniform data distribution within each partition and estimate each cell frequency with average partition frequency. The spatial estimate algorithm is described in Algorithm 16. For each time stamp k, a partition count is aggregated from cells for every partition (Line 4). It is then perturbed by the Laplace mechanism to guarantee differential privacy (Line 5) and the average noisy count is used to estimate the frequency of each cell within the partition (Line 6). The intuition is that the cells within each partition have similar density. Therefore by uniformly distributing the noisy partition count to each cell, we reduce the magnitude of perturbation error applied to each cell without compromising the accuracy.

One advantage of the spatial estimation algorithm is that it relies on simple and practical assumptions. The complexity is also $O(w^2)$ for each time stamp since every cell is visited $O(1)$ times. Although it takes extra time to build the spatial index for initialization, we see it as a one-time cost which can be done off-line. The runtime of the spatial estimation is reduced because only one perturbation noise is needed for every partition at every k (Line 5). In contrast, both the baseline LPA algorithm and the temporal estimation algorithm will generate one perturbation noise for each cell at every k. We will study their runtime performance in the next section.

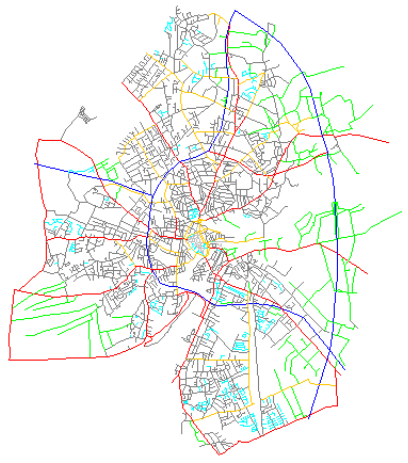
5.6 Experimental Results

We implemented the proposed algorithms as well as alternative methods in Java with JSC[4] for simulating the statistical distributions. All experiments were conducted using a 2.90GHz Intel Core i7 PC with 8GB RAM.

Data Set. We generated synthetic traffic data with the Brinkhoff generator [6]. The input of the generator is the road map of Oldenburg in Germany[5] (Figure 5.7(a)),which contains 6,105 nodes and 7,035 edges, and the output is a set of moving objects on the road network. We created the data set with 100 discrete timestamps, with 500,000 objects at the beginning and 25,000 new objects introduced at every time stamp. The starting positions and

[4]http://www.jsc.nildram.co.uk

[5]http://iapg.jade-hs.de/personen/brinkhoff/generator/


(a) Map of Oldenburg

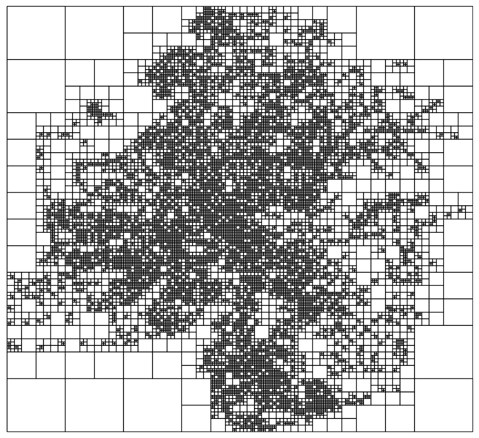

(b) Partitions by Quadtree

Figure 5.7: Road Network and Generated Partitions of Oldenburg Dataset

destinations of the moving objects are selected randomly by the generator (see [6] for detailed network-based techniques). Once an object reaches its destination, it disappears from the map. At the server side, we use a 2D grid with 1024×1024 cells to record the locations of the moving objects, with each cell representing approximately 20×20 square meters' range in reality. We assign each cell a class label, i.e. *sparse* or *dense*, based on the presence of roads within its extent. Roughly 95% cells are labeled *sparse*, indicating only the rest 5% have been visited by the moving objects. Figure 5.7(b) visualizes the partition result achieved by the Quadtree-based algorithm with the depth threshold $d = 8$. It can be seen that spares regions around map edge are contained in larger partitions and densely connected regions in map center are further split into smaller partitions. We will evaluate the set of *sparse* cells and the set of *dense* cells separately since they exhibit very different dynamics over time.

Comparison. We compare our proposed solutions against the state-of-the-art methods which are summarized below:

- **DFT** is the Fourier Perturbation Algorithm recently proposed by Rastogi and Nath [59] for sharing single time series. It first performs the Discrete Fourier Transform on an input time series and retains only the first l DFT coefficients. Those coefficients are then perturbed by the Laplace mechanism to guarantee differential privacy. Finally, the Inverse Discrete Fourier Transform is performed on the perturbed coefficients to produce a released series. The number of coefficients to preserve, i.e. l, is a user-specified parameter. In our empirical study, we set $l = 20$ according to their recommendation [59].
- **kd-hybrid** is proposed by Cormode et al [16] as their best method to achieve differentially private space decomposition with static data. Without the help of a grid, *kd-hybrid* builds a mixture index over the 2D data space that begins with kd-tree and switches to quad-tree at a certain level. They slightly modified the kd-tree algorithm, changing the fanout rate to 4 in order to reduce the privacy budget consumption. According to their studies, *kd-hybrid* is most reliable among several representative differentially private space partitioning methods. They reported the optimal parameter setting empirically with the height set to 8 and the switch level set to 4.

Since the DFT algorithm can be only performed with the complete series, it is not compatible to real-time applications. However, we include it in our evaluation since it serves as a good, off-line reference for utility. As for the *kd-hybrid* algorithm, there are two limitations. One is its high privacy cost since the algorithm iteratively spends budget on finding “private medians” for every data snapshot. The other limitation is its high computation cost: application of the *kd-hybrid* method requires constructing the index structure at every time stamp. Experiments with the author’s provided implementation take hours for each iteration, since the domain size and the

Symbol	**Description**	**Default Value**
α	Total Privacy Budget	1
w	Resolution for Each Dimension	1024
T	Length of Multidimensional Time Series	100
Q_{sparse}	Process Noise for *Sparse* Cells	10^{-2}
Q_{dense}	Process Noise for *Dense* Cells	10^{3}
R	Gaussian Measurement Noise	10^{6}
d	Depth Threshold for Quadtree	8

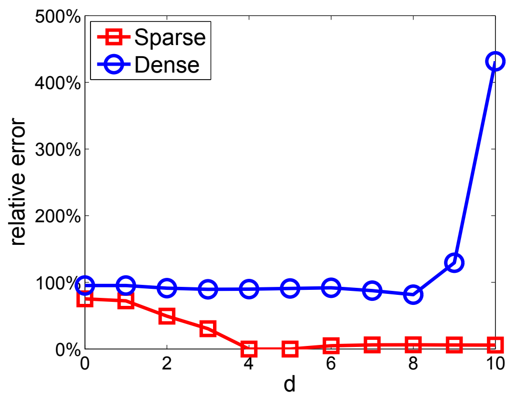
Table 5.4: Parameters for Traffic Histogram Release

number of objects in our data set are extremely large. We conclude that the *kd-hybrid* method is too expensive for the continuous, real-time applications and therefore do not include the results in the remaining section.

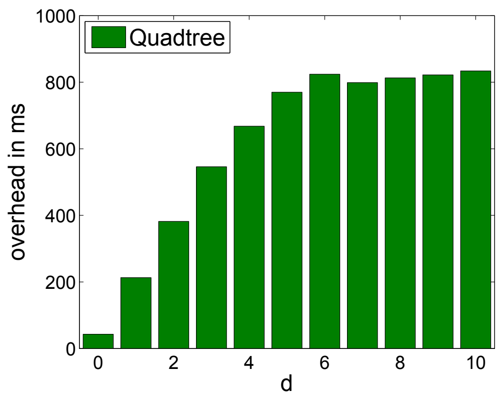
5.6.1 Parameter Impacts

The default parameter setting, unless otherwise noted, is summarized in Table 5.6. Note that Q_{sparse} and Q_{dense}, which correspond to Q^c in Equation 5.18 for *sparse* and *dense* cells, can be chosen by domain users and our default setting may not be optimal. As for R from Equation 5.19, we set its value according to our previous studies [27], which shows that the optimally R is proportional to T^2/α^2.

We study the impact of the depth threshold d used in Algorithm 15 in terms of utility as defined in Equation 5.17 and runtime. Intuitively, the larger value d takes, the finer partitions the algorithm results in, especially along the border of sparse and dense regions. However, it also incurs a higher overhead to construct the index as we can expect. Figure 5.8(a) plots separately the utility of released series for sparse cells and dense cells when varying the depth threshold d. For each class of cells, we plot the median



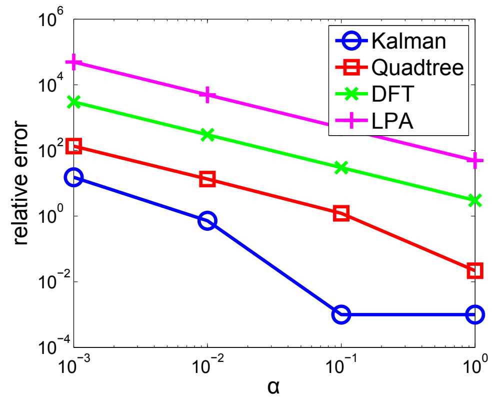
(a) Utility vs. Depth Threshold



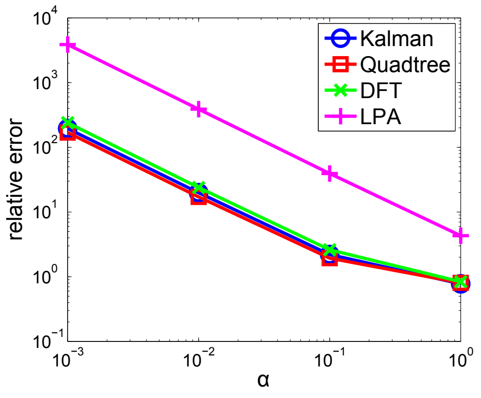
(b) Overhead vs. Depth Threshold

Figure 5.8: Impact of d on Quadtree-based Spatial Estimation

relative error to avoid the extremely small or large values. As we increase the threshold value, the error for sparse cells gradually drops to 0 between $d = 0$ and $d = 4$ and remains stable when d value is further increased. This is due to the fact that majority sparse cells are located together (on map edge) and will not take too many splits to be separate from dense cells (in map center). Increasing the d value can help separating those sparse cells on the boarder line. However, the utility of majority sparse cells is not affected since those on the boarder line only count for a very small percentage. On the other hand, dense cells require more splits to achieve optimal separation ($d = 8$). When further split ($d > 8$), the perturbation noise greatly impacts their utility due to data sparsity. Figure 5.8(b) shows the overhead for constructing the aggregation index when varying the d value. It takes at most 0.9 second and we note that it is a one-time cost. As we expect, a higher depth threshold requires more construction time (from $d = 0$ to $d = 6$). However, when $d > 6$ the overhead does not grow since there are only very few partitions that do not meet the homogeneous requirement at depth 6. As can be seen in Figure 5.7(b), the densely connected areas in the map are split into finer partitions compared to less populous areas on the map edge.



(a) Privacy vs. Accuracy for *Sparse* Cells

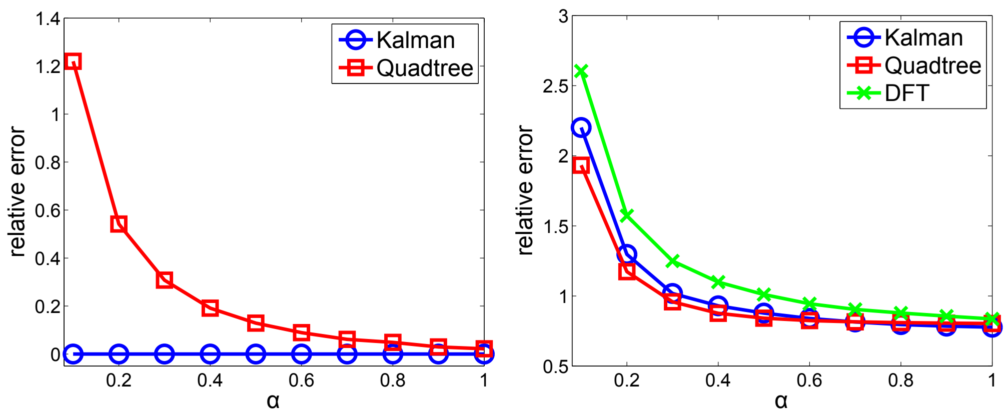


(b) Privacy vs. Accuracy for *Dense* Cells

Figure 5.9: Privacy vs. Accuracy for Individual Cell Counts

5.6.2 Utility Performance

Utility vs. Privacy. Here we examine the trade-off between utility and privacy. Our proposed solutions, i.e. `Kalman` and `Quadtree`, are compared against the baseline LPA and state-of-the-art DFT algorithm, in terms of utility of individual cells. Figure 5.9(a) and Figure 5.9(b) plot the utility of sparse cells and dense cells respectively when varying the overall privacy budget, i.e. α value, from $[10^{-3}, 10^{0}]$. As we can see, the baseline LPA algorithm results in highest relative error in both figures. The DFT algorithm results in high relative error with sparse cells even with high privacy cost ($\alpha = 1$), due to the perturbation and reconstruction error. Our solutions `Kalman` and `Quadtree` outperform both LPA and DFT especially with sparse cells, as `Quadtree` only results in 10% error and `Kalman` produces 0% error when $\alpha = 1$. As for the dense cells, both `Kalman` and `Quadtree` slightly outperforms DFT, which is supposed to be optimal. When $\alpha = 1$, DFT results in 83% error due to lack of smoothness in the original frequency series, while our solutions provide comparable utility to DFT and real-time data release. Figure 5.10(a) and Figure 5.10(b) provide a closer look at the utility



(a) Privacy vs. Accuracy for *Sparse* Cells (b) Privacy vs. Accuracy for *Dense* Cells

Figure 5.10: Privacy vs. Accuracy for Individual Cell Counts: $\alpha \in [0.1, 1]$

curves within a more practical range of privacy budget $\alpha \in [0.1, 1]$. DFT and LPA are not plotted in one or both figures because the errors they result in are prohibitive. For sparse cells, `Kalman` provides optimal performance even under small privacy budget ($\alpha = 0.1$), thanks to the accurate modeling. `Quadtree` is able to approach 0% error as α value increases. For dense cells, we observe that `Quadtree` provides the best utility in the same privacy budget range. We conclude that both our proposed solutions outperforms existing methods, allowing for real-time data sharing without compromising the utility.

Utility of Range Queries. Here we evaluate our solutions with range queries, where each query is a square window that covers a neighborhood of $m \times m$ cells. For each m value, we randomly generate 100 queries of size $m \times m$, evaluate each method with the same set of queries, and plot the average relative error. Note that when $m = 1$, each set query consists of one cell only and therefore the set query error is equivalent to individual cell error. Our findings are summarized in Figure 5.11. Our temporal estimation algorithm based on the Kalman filter clearly outperforms `Quadtree` and `LPA`

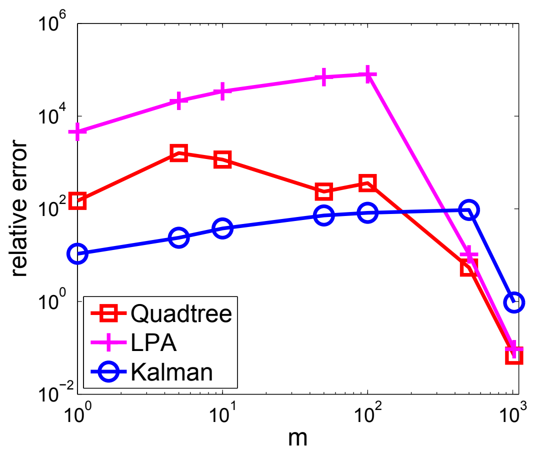


Figure 5.11: Accuracy for Spatial Range Queries

with smaller query windows ($m \leq 100$). For all three methods, the relative error shows a growing trend to different extent as the query set size increases, mainly due to the data sparsity in the space. When $m = 500$, we observe that the error of `Kalman` keeps accumulating while `Quadtree` and `LPA` show reduced relative error. We believe that it is because `Kalman` does not explicitly utilize the spatial correlation between cells. When querying the entire space ($m = 1024$), both `Quadtree` and `LPA` provide good utility because the Laplace noise added to each cell is from a zero-mean distribution and the sum of a large set of such noises is likely to be small. Overall, `Quadtree` outperforms `LPA` by making sound estimation within close-to-uniform partitions.

5.6.3 Runtime Performance

Lastly we compare the runtime performance of our solutions against the baseline since computation time is critical to real-time applications. We measure and plot the runtime for releasing the two-dimensional aggregates for 100 timestamps in order to mitigate random disturbance from the operating system. The results are summarized in Figure 5.12. As we can see, all three

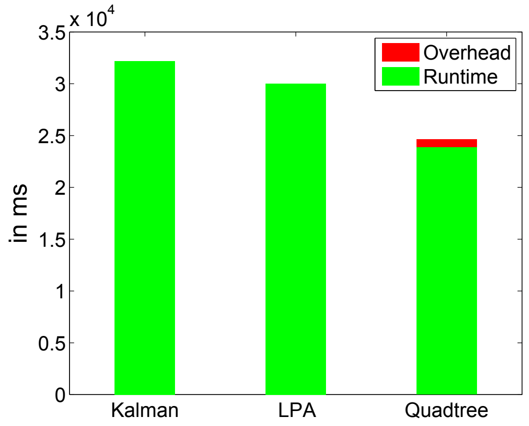


Figure 5.12: Runtime Performances for Traffic Monitoring

methods take less than 35 seconds to release 100 snapshots of 1024×1024 cell frequencies with differential privacy guarantee. Note that the state-of-the-art *kd-hybrid* takes hours to release/evaluate one time stamp. Compared to `LPA`, our solution `Kalman` takes roughly 2 more seconds in total to perform prediction and correction at every time stamp. `Quadtree` turns out to be the most time efficient, even though it has a small overhead in building the spatial index. This is because less perturbation is performed by `Quadtree`, as at every time stamp we only generate one perturbation noise for each partition rather than for each cell as in `LPA`.

Chapter 6

Conclusion and Future Work

The work described in this dissertation has been concerned with the development of differentially private aggregate release algorithms for a class of real-time spatio-temporal data analytical tasks. A novel framework utilizing an internal data model and feedback mechanism has been proposed. An investigation of individual privacy risk with domain statistics was presented, which shows the privacy cost can be reduced for monitoring tasks over a long period of time. Two ways to model the spatio-temporal correlation between aggregates and time-stamps have been described and they were shown effective for improving the accuracy of released data. The contributions of this dissertation are summarized below:

- A generic framework, FAST, for releasing real-time aggregates with differential privacy guarantee [28, 29].
- A software toolkit which implements FAST filtering and sampling algorithms, along with baseline and offline solutions [32].
- Statistical analysis of individual privacy risk when FAST monitoring period is long [30].
- Modeling spatio-temporal correlation with user behavioral patterns when FAST is applied to multi-variate aggregate monitoring [26].

- Modeling complex/non-linear correlation with background road network for differentially private traffic monitoring [31].

6.1 Summary of Dissertation

An introduction of the potential and challenges of privacy preservation in spatio-temporal data analytics was first presented in Chapter 1. The increasing amount of available data has enabled many data mining applications to understand important phenomena, such as disease surveillance, web mining, and traffic monitoring. However, the data generated by individuals contain sensitive information regarding their health, confidential preferences, locations, and etc, which should not be disclosed to untrusted parties, e.g. data researchers, to protect individual privacy. The state-of-the-art paradigm for private data publication, i.e. differential privacy, provides rigorous guarantee for individual data contributors. Due to the sequential composition property, when applied to continual data publication, the perturbation cost becomes high and the released data contain large privacy perturbation error.

There have been a plethora of works developed for differentially private data publication, the majority of which focus on one-time release of static data. Few works studied the problem of releasing time series or continual statistics with differential privacy guarantee. The work of [59] relies on time series Discrete Fourier Transform (denoted as DFT), which is not compatible with real-time data mining tasks. Dwork et al. [23] and Chan et al. [12] studied a differentially private continual counter over a binary stream. However, both works adopt an event-level privacy model, with the perturbation mechanism designed to protect the presence of an individual event, i.e. a user's contribution to the data stream at a single time point, rather than the presence or privacy of a user. Chapter 2 reviewed relevant works on differential privacy, time series analysis, and other data analytics.

Chapter 3 presented FAST, an adaptive framework with filtering and sampling for monitoring real-time aggregates under differential privacy. The key innovation is that FAST utilizes feedback loops based on observed values to dynamically adjust the estimation model as well as the sampling rate. To minimize the overall privacy cost, FAST uses the PID controller to adaptively sample long time-series according to detected data dynamics. As to improve the accuracy of data release per time stamp, FAST adopts the state-space model and filtering techniques to predict data values at non-sampling points and to estimate true values from perturbed values at sampling points. Experiments with real-world and synthetic data sets show that it is beneficial to incorporate feedback into both the estimation model and the sampling process. The results confirmed that the adaptive framework improves utility of time-series release and has excellent performance even under small privacy cost.

The software [32] implements all FAST algorithms, including three filtering options and two sampling methods, along with baseline and offline solutions. The graphical interface allows users to upload the input aggregate time series in batch mode and to enter one aggregate value at a time in real-time mode. By generating a private, released value in real-time, this software demonstrates the efficiency and effectiveness of FAST algorithms. It is the first software that provides trusted servers, i.e. data holders, necessary tools to continuously release differentially private aggregated data. We believe it will enable a wide range aggregate monitoring tasks.

When applying differential privacy to release long series of aggregates, the high perturbation cost, which is proportional to the length of aggregate series, prevents the released data from being useful. Chapter 4 investigated domain-specific individual privacy risk in order to reduce the perturbation cost to achieve differential privacy. This practical estimate of individual privacy risk with domain statistics quantifies the rareness of the worst-case privacy disclosure, showing that the majority of the individuals, e.g. 99%, do not contribute to the 10 years' long daily aggregates more than a very small number of times, e.g. 2 daily aggregates. The method for analysis can be applied to general monitoring tasks, given domain-specific usage statistics. This result shows when collecting individual data for privacy-preserving data analysis, excluding a small percentage of individuals with high disclosure risk allows for lower perturbation error and more accurate analysis results from the collected data.

To monitor multiple aggregates simultaneously, such as traffic counts at road intersections within a city, data sparsity imposes a new challenge to privacy preserving mechanisms, as released aggregates are dominated by privacy perturbation noise. The straight-forward solution is to model and apply FAST algorithms to each individual aggregate series separately. However, the aggregates between timestamps are correlated, as individual behavior follows certain movement patterns. Furthermore, the aggregates of interest can be correlated at the same time instance, due to distributional or topological constraints.

Chapter 5 first presented a study of incorporating domain-specific individual behavioral patterns in FAST process model for the spatio-temporal correlation between multiple aggregates. As a result, the prediction of future aggregates becomes more accurate and the posterior estimates by FAST filtering mechanism improve over the noisy, perturbed aggregates. We also present the computation efficiency of FAST framework for releasing multivariate aggregate time series and point out the most computation-intensive procedure for potential speed-up. We believe this method can be applied to a wide range of monitoring tasks where individual usage patterns, especially Markovian properties, can be employed.

For complex correlation, a Quadtree-based differentially private mechanism was presented to continuously release traffic histogram over a given road network. Results show that the utility of released data can be improved by taking into consideration of spatial correlation between adjacent locations. This work illustrates how domain knowledge, e.g. road network, can be used to model complex spatio-temporal correlation between multiple aggregates, e.g. traffic count at different locations. Moreover, the private, accurate traffic histogram released by this approach will enable a wide range of geo-spatial data mining tasks.

6.2 Recommendations for Future Work

Although the results in this dissertation demonstrated effectiveness of the proposed methods for a variety of aggregate monitoring tasks, they can be extended in a number of ways:

Extending FAST framework with non-linear data models and alternative privacy mechanisms

When applied to various domain statistics, FAST framework can be easily generalized to monitor aggregate series which follow non-linear process models, such as sinusoidal functions, in which case the particle filter based estimation algorithm can be used for prediction and correction. It will be interesting to study the applicability of alternative differential privacy mechanisms, such as geometric mechanism [35] and exponential mechanism [55], and investigate their privacy-utility tradeoff under FAST framework.

Extending private web monitoring for large scale implementation

The first potential is to combine user browsing requests to different websites/servers. Browsing across platforms is very common in the real world, as users often switch between search engines and shopping sites or social networks. Combining the browsing requests to different servers which are made within a short time window could provide a broader view of user navigation patterns and thus enable more data mining applications. The second aspect, i.e. the scalability of the multivariate method, becomes an immediate challenge when the domain of web pages is large. As the m value increases, the multivariate state-space method yields high time complexity, i.e. $\mathcal{O}(m^3)$, due to matrix operations. Classic techniques that exploit matrix properties, such as sparsity, rank, and decomposability, can be utilized to reduce the computation requirement at the cost of accuracy.

Extending private traffic monitoring with data-dependent techniques and alternative spatial indexing structures

It has been confirmed that considering the spatial correlation is beneficial to privacy-preserving traffic monitoring, even by simply assuming uniform distribution within each partition as above. In order to capture the fine-grained correlation among grid cells, data-dependent approaches, which includes [16, 66], can be adopted to reflect the potential range of actual cell density. Furthermore, the partitioning algorithm can be adapted to further explore alternative spatial indexing structures, such as Binary Space Partition, Hilbert Curve, R-Tree, and etc.

Appendix

The Kalman Filter Posterior Analysis

Given that prediction and correction are performed at every time stamp, here we analyze the posterior error variance where $\hat{x}_k$ is derived with the Gaussian assumption, i.e. Equation 3.14, while z_k is perturbed by the Laplace mechanism, i.e. Equation 3.6 and 3.7. The posterior error variance is defined as follows:

$$var(\hat{x}_k - x_k) = E(\hat{x}_k - x_k)^2 - E^2(\hat{x}_k - x_k)\ . \tag{A.1}$$

Note that both process noise and measurement noise are white and mutually, serially independent. By definition of $\hat{x}_k$, we get the following:

$$E(\hat{x}_k - x_k) = 0\ .$$

Therefore, we will only need to estimate the first term in Equation (A.1). Substituting Equation 3.14 leads to

$$\begin{aligned}
E(\hat{x}_k - x_k)^2 &= E[(1-K_k)(\hat{x}_k^- - x_k) + K_k(z_k - x_k)]^2 \\
&= E[(1-K_k)(\hat{x}_{k-1} - x_k) + K_k\nu]^2 \\
&= E[(1-K_k)(\hat{x}_{k-1} - x_{k-1}) \\
&\quad + (1-K_k)(x_{k-1} - x_k) + K_k\nu]^2 \\
&= E(1-K_k)^2(\hat{x}_{k-1} - x_{k-1})^2 \\
&\quad + E[(1-K_k)^2\omega^2] + E(K_k^2\nu^2) \\
&= (1-K_k)^2 E(\hat{x}_{k-1} - x_{k-1})^2 \\
&\quad + (1-K_k)^2 Q + K_k^2\frac{2T^2}{\alpha^2}
\end{aligned} \tag{A.2}$$

where α is the overall privacy budget and T is the lifetime of the time-series.

Substituting the Kalman gain, i.e. Equation 3.17, into Equation (A.2), we get

$$E(\hat{x}_k - x_k)^2 = \frac{R^2[E(\hat{x}_{k-1} - x_{k-1})^2 + Q]}{(P_k^- + R)^2} + \frac{2P_k^{-2} T^2}{(P_k^- + R)^2 \alpha^2} \; .$$

Applying the gradient descendant method to minimize the posterior error variance, we obtain the following result for R:

$$R = \frac{T^2}{\alpha^2} \frac{2P_k^-}{E(\hat{x}_{k-1} - x_{k-1})^2 + Q} \; .$$

Fixed-Rate Sampling Posterior Analysis

Posterior Error at a Sampling Point

When sampling techniques are combined with the Kalman filter based estimation algorithm, measurements are obtained only at sampling points. Assume fixed rate sampling is applied with interval I and the current time stamp k is a sampling point. The prior estimate at time stamp k is actually the posterior estimate of time $k - I$ (the last sampling point):

$$\hat{x}_k^- = \hat{x}_{k-I} \; .$$

By definition of $\hat{x}_k$, we get

$$\begin{aligned} \hat{x}_k - x_k &= (1 - K_k)(\hat{x}_{k-I} - x_k) + K_k(z_k - x_k) \\ &= (1 - K_k)[(\hat{x}_{k-I} - x_{k-I}) - (x_k - x_{k-I})] + K_k \nu \\ &= (1 - K_k)[(\hat{x}_{k-I} - x_{k-I}) - \sum_{k-I+1}^{k} \omega_j] + K_k \nu \; . \end{aligned}$$

According to the process model in Equation 3.4, I independent white Gaussian process noise variables, i.e. ω_j's, are introduced between time $k - I + 1$

and time k. Therefore,

$$\begin{aligned}
E(\hat{x}_k - x_k) &= (1 - K_k)[E(\hat{x}_{k-I} - x_{k-I}) \\
&\quad - \sum_{k-I+1}^{k} E\omega_j] + K_k E(\nu) \\
&= 0
\end{aligned}$$

and

$$\begin{aligned}
var(\hat{x}_k - x_k) &= E(\hat{x}_k - x_k)^2 \\
&= (1 - K_k)^2 E(\hat{x}_{k-I} - x_{k-I})^2 \\
&\quad + (1 - K_k)^2 \sum_{k-I+1}^{k} E\omega_j^2 + K_k^2 E\nu^2 \\
&= (1 - K_k)^2 [E(\hat{x}_{k-I} - x_{k-I})^2 + IQ] \\
&\quad + K_k^2 \frac{2T^2}{I^2\alpha^2} \qquad \text{(B.1.1)}
\end{aligned}$$

where $\nu \sim Lap(0, \frac{T}{I\alpha})$ for fixed rate sampling with interval I.

Prediction Error at a Non-Sampling Point

At non-sampling points, the prior estimates will be released and we will derive the error variance below. Assume the current time stamp k is a non-sampling point and the most recent sample occurs at time stamp $k - t$. Therefore, by applying process model, we get

$$\begin{aligned}
\hat{x}_k^- - x_k &= \hat{x}_{k-t} - x_k \\
&= \hat{x}_{k-t} - x_{k-t} - (x_k - x_{k-t}) \\
&= \hat{x}_{k-t} - x_{k-t} - \sum_{k-t+1}^{k} \omega_j
\end{aligned}$$

and

$$E(\hat{x}_k^- - x_k) = 0 \ .$$

Therefore, the prediction error at a non-sampling point is:

$$\begin{aligned} var(\hat{x}_k^- - x_k) &= E(\hat{x}_k^- - x_k)^2 \\ &= E(\hat{x}_{k-t} - x_{k-t})^2 + \sum_{k-t+1}^{k} E\omega_j^2 \\ &= E(\hat{x}_{k-t} - x_{k-t})^2 + tQ \ . \end{aligned} \tag{B.2.1}$$

Overall Error

Let S denote the set of sampling points, for instance, $S = \{0, I, 2I, ...\}$. Suppose $k \in S$ and let var_j denote the error variance at any time stamp j. The sum of error between two samples, i.e. from time stamp k to $k + I - 1$, can be found by applying Equation (B.2.1):

$$\begin{aligned} \sum_{k}^{k+I-1} var_j &= \sum_{k}^{k+I-1} [var_k + (j-k)Q] \\ &= I \cdot var_k + \sum_{0}^{I-1} jQ \\ &= I \cdot var_k + \frac{I(I-1)}{2} Q \ . \end{aligned} \tag{B.3.1}$$

The overall error for the entire time series can be derived by applying Equation (B.3.1) for every sampling period:

$$\begin{aligned} sumErr &= \sum_{k \in S} \sum_{k}^{k+I-1} var_j \\ &= I \cdot \sum_{k \in S} var_k + |S| \frac{I(I-1)}{2} Q \\ &= I \cdot \sum_{k \in S} var_k + \frac{T(I-1)}{2} Q \end{aligned}$$

since the size of S is T/I.

We can further substitute each var_k at sampling points, i.e. Equation (B.1.1). Therefore the sum of error variance over T can be viewed as a function of interval length I:

$$\begin{aligned} sumErr = I^2 \cdot Q \sum_{k \in S} (1 - K_k)^2 \\ + I \cdot [\sum_{k \in S} var_{k-I}(1 - K_k)^2 + \frac{TQ}{2}] \\ + I^{-1} \cdot \frac{2T^2}{\alpha^2} \sum_{k \in S} K_k^2 - \frac{TQ}{2} \end{aligned}$$

which is very challenging to minimize *a priori* due to the recursive filtering procedures.

Bibliography

[1] Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp. A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50:174–188, 2001.

[2] Yaakov Bar-Shalom, X.-Rong Li, and Thiagalingam Kirubarajan. *State Estimation in Discrete-Time Linear Dynamic Systems*, pages 199–266. John Wiley & Sons, Inc., 2002.

[3] Michael Barbaro and Tom Zeller. A face is exposed for aol searcher no. 4417749. *The New York Times*, August 2006.

[4] Avrim Blum, Katrina Ligett, and Aaron Roth. A learning theory approach to non-interactive database privacy. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, pages 609–618, New York, NY, USA, 2008. ACM.

[5] C. A. Bradley, H Rolka, D Walker, and J Loonsk. Biosense: implementation of a national early event detection and situational awareness system. *MMWR: Morbidity Mortality Weekly Report*, 54:11 – 19, 2005.

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

[7] P.J. Brockwell and R.A. Davis. *Introduction to time series and forecasting*. Springer texts in statistics. Springer, 2002.

[8] Robert Grover Brown and Patrick Y C Hwang. *Introduction to Random Signals and Applied Kalman Filtering*, volume 2. John Wiley & Sons, 1997.

[9] J.B.D. Caberera, B. Ravichandran, and R.K. Mehra. Statistical traffic modeling for network intrusion detection. In *Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 2000. Proceedings. 8th International Symposium on*, pages 466–473, 2000.

[10] Igor Cadez, David Heckerman, Christopher Meek, Padhraic Smyth, and Steven White. Visualization of navigation patterns on a web site using model-based clustering. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '00, pages 280–284, New York, NY, USA, 2000. ACM.

[11] Davide Canali and Davide Balzarotti. Behind the scenes of online attacks: an analysis of exploitation behaviors on the web. In *NDSS 2013, 20th Annual Network and Distributed System Security Symposium, February 24-27, 2013, San Diego, CA, United States*, San Diego, UNITED STATES, 02 2013.

[12] T.-H. Hubert Chan, Elaine Shi, and Dawn Song. Private and continual release of statistics. In *Proceedings of the 37th international colloquium conference on Automata, languages and programming: Part II*, pages 405–417, Heidelberg, 2010. Springer-Verlag.

[13] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):15:1–15:58, July 2009.

[14] R. Chen, N. Mohammed, B. C. M. Fung, B. C. Desai, and L. Xiong. Publishing set-valued data via differential privacy. *The Proceedings of the VLDB Endowment (PVLDB)*, 4(11):1087–1098, August 2011.

[15] Edward Chlebus and Jordy Brazier. Nonstationary poisson modeling of web browsing session arrivals. *Inf. Process. Lett.*, 102(5):187–190, May 2007.

[16] Graham Cormode, Cecilia Procopiuc, Divesh Srivastava, Entong Shen, and Ting Yu. Differentially private spatial decompositions. In *Proceedings of the 2012 IEEE 28th International Conference on Data Engineering*, pages 20–31, Washington, DC, 2012. IEEE Computer Society.

[17] Graham Cormode, Cecilia Magdalena Procopiuc, Divesh Srivastava, and Thanh T. L. Tran. Differentially private summaries of sparse data. In *EDBT*, 2012.

[18] Yves-Alexandre de Montjoye, César A. Hidalgo, Michel Verleysen, and Vincent D. Blondel. Unique in the Crowd: The privacy bounds of human mobility. *Scientific Reports*, 3, March 2013.

[19] A. Doucet, N. de Freitas, N. Gordon, and A. Smith. *Sequential Monte Carlo Methods in Practice.* Statistics for Engineering and Information Science. Springer, 2001.

[20] Arnaud Doucet, Simon Godsill, and Christophe Andrieu. On sequential monte carlo sampling methods for bayesian filtering. *Statistics and Computing*, 10(3):197–208, July 2000.

[21] Arnaud Doucet and Adam M Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of Nonlinear Filtering*, 12:656–704, 2009.

[22] Cynthia Dwork, Frank Mcsherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *In Proceedings of the 3rd Theory of Cryptography Conference*, pages 265–284. Springer, 2006.

[23] Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N. Rothblum. Differential privacy under continual observation. In *Proceedings of the 42nd ACM symposium on Theory of computing*, pages 715–724, New York, 2010. ACM.

[24] Cynthia Dwork, Moni Naor, Omer Reingold, Guy N. Rothblum, and Salil Vadhan. On the complexity of differentially private data release: efficient algorithms and hardness results. STOC '09, pages 381–390, New York, NY, USA, 2009. ACM.

[25] Magdalini Eirinaki and Michalis Vazirgiannis. Web mining for web personalization. *ACM Trans. Internet Technol.*, 3(1):1–27, February 2003.

[26] Liyue Fan, Luca Bonomi, Li Xiong, and Vaidy Sunderam. Monitoring web browsing behavior with differential privacy. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14, pages 177–188, Republic and Canton of Geneva, Switzerland, 2014. International World Wide Web Conferences Steering Committee.

[27] Liyue Fan and Li Xiong. Adaptively sharing time-series with differential privacy. *CoRR*, abs/1202.3461, 2012.

[28] Liyue Fan and Li Xiong. Real-time aggregate monitoring with differential privacy. CIKM '12, pages 2169–2173. ACM, 2012.

[29] Liyue Fan and Li Xiong. An adaptive approach to real-time aggregate monitoring with differential privacy. *IEEE Transactions on Knowledge and Data Engineering*, 99(PrePrints):1, 2013.

[30] Liyue Fan and Li Xiong. Differentially private anomaly detection with a case study on epidemic outbreak detection. In *Data Mining Workshops (ICDMW), 2013 IEEE 13th International Conference on*, Dec 2013.

[31] Liyue Fan, Li Xiong, and Vaidy Sunderam. Differentially private multi-dimensional time series release for traffic monitoring. DBSec'13, pages 33–48. Springer-Verlag, 2013.

[32] Liyue Fan, Li Xiong, and Vaidy Sunderam. Fast: Differentially private real-time aggregate monitor with filtering and adaptive sampling. SIGMOD '13, pages 1065–1068. ACM, 2013.

[33] Centers for Disease Control and Surveillance. Early aberration reporting system, 2007.

[34] MICHELLE L. GATTON, LOUISE A. KELLY-HOPE, BRIAN H. KAY, and PETER A. RYAN. Spatial-temporal analysis of ross river virus disease patterns in queensland, australia. *The American Journal of Tropical Medicine and Hygiene*, 71(5):629–635, 2004.

[35] Arpita Ghosh, Tim Roughgarden, and Mukund Sundararajan. Universally utility-maximizing privacy mechanisms. In *Proceedings of the 41st annual ACM symposium on Theory of computing*, STOC '09, pages 351–360, New York, NY, USA, 2009. ACM.

[36] N.J. Gordon, D.J. Salmond, and A.F.M. Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. *Radar and Signal Processing, IEE Proceedings F*, 140(2):107 –113, apr 1993.

[37] Michaela Götz, Suman Nath, and Johannes Gehrke. Maskit: privately releasing user context streams for personalized mobile applications. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, SIGMOD '12, pages 289–300, New York, NY, USA, 2012. ACM.

[38] Traffic Monitoring Guide. Fhwa. *US Department of Transportation, Washington, DC*, 38, 2001.

[39] D.M. Hawkins and D.H. Olwell. *Cumulative Sum Charts and Charting for Quality Improvement*. Information Science and Statistics Series. Springer Verlag, 1998.

[40] Michael Hay, Vibhor Rastogi, Gerome Miklau, and Dan Suciu. Boosting the accuracy of differentially private histograms through consistency. *Proc. VLDB Endow.*, 3(1-2):1021–1032, September 2010.

[41] Yuan Hong, Jaideep Vaidya, Haibing Lu, and Mingrui Wu. Differentially private search log sanitization with optimal output utility. In *EDBT*, 2012.

[42] L. C. Hutwagner, W. W. Thompson, G. M. Seeman, and T. Treadwell. A simulation model for assessing aberration detection methods used in public health surveillance for systems with limited baselines. *Statistics in Medicine*, 24(4).

[43] Michael Jackson, Atar Baer, Ian Painter, and Jeff Duchin. A simulation study comparing aberration detection algorithms for syndromic surveillance. *BMC Medical Informatics and Decision Making*, 7(1):6, 2007.

[44] Ankur Jain, Edward Y. Chang, and Yuan-Fang Wang. Adaptive stream resource management using kalman filters. SIGMOD '04, pages 11–22, New York, NY, USA, 2004. ACM.

[45] Rudolph Emil Kalman et al. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

[46] M. King and M. King. *Process Control: A Practical Approach*. John Wiley & Sons, 2011.

[47] Ken P Kleinman and Allyson M Abrams. Assessing surveillance using sensitivity, specificity and timeliness. *Statistical Methods in Medical Research*, 15(5):445–464, 2006.

[48] Aleksandra Korolova, Krishnaram Kenthapadi, Nina Mishra, and Alexandros Ntoulas. Releasing search queries and clicks privately. In *WWW*, pages 171–180. ACM, 2009.

[49] Raymond Kosala and Hendrik Blockeel. Web mining research: a survey. *SIGKDD Explor. Newsl.*, 2(1):1–15, June 2000.

[50] Ravi Kumar and Andrew Tomkins. A characterization of online browsing behavior. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 561–570, New York, NY, USA, 2010. ACM.

[51] Chao Li and Gerome Miklau. An adaptive mechanism for accurate query answering under differential privacy. *Proc. VLDB Endow.*, 5(6):514–525, 2012.

[52] J. MacCormick and A. Blake. A probabilistic exclusion principle for tracking multiple objects. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 1, pages 572 –578 vol.1, 1999.

[53] C. Masreliez. Approximate non-gaussian filtering with linear state and observation relations. *Automatic Control, IEEE Transactions on*, 20(1):107 – 110, feb 1975.

[54] Frank McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. volume 53, pages 89–97, New York, 2010. ACM.

[55] Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '07, pages 94–103, Washington, DC, USA, 2007. IEEE Computer Society.

[56] Spiros Papadimitriou, Feifei Li, George Kollios, and Philip S. Yu. Time series compressibility and privacy. VLDB '07, pages 459–470. VLDB Endowment, 2007.

[57] A. Pelecanos, P. Ryan, and M. Gatton. Outbreak detection algorithms for seasonal disease data: a case study using ross river virus disease. *BMC medical informatics and decision making*, 10(1):74, 2010.

[58] Michael K. Pitt and Neil Shephard. Filtering via simulation: Auxiliary particle filters. *Journal of the American Statistical Association*, 94(446):590–599, 1999.

[59] Vibhor Rastogi and Suman Nath. Differentially private aggregation of distributed time-series with transformation and encryption. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 735–746, New York, 2010. ACM.

[60] H.W. Sorenson. *Kalman filtering: theory and application.* IEEE Press selected reprint series. IEEE Press, 1985.

[61] Donna F. Stroup, G. David Williamson, Joy L. Herndon, and John M. Karon. Detection of aberrations in the occurrence of notifiable diseases surveillance data. *Statistics in Medicine*, 8(3):323–329, 1989.

[62] Fu-Chiang Tsui, Jeremy U Espino, Virginia M Dato, Per H Gesteland, Judith Hutman, and Michael M Wagner. Technical description of rods: A real-time public health surveillance system. *Journal of the American Medical Informatics Association*, 10(5):399–408, 2003.

[63] M.M. Wagner, F.C. Tsui, J.U. Espino, V.M. Dato, D.F. Sittig, R.A. Caruana, L.F. McGinnis, D.W. Deerfield, M.J. Druzdzel, and D.B.

Fridsma. The emerging science of very early detection of disease outbreaks. *Journal of Public Health Management and Practice*, 7(6):51–59, 2001.

[64] Di Wang, Yeye He, Elke Rundensteiner, and Jeffrey F. Naughton. Utility-maximizing event stream suppression. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, SIGMOD '13, pages 589–600, New York, NY, USA, 2013. ACM.

[65] Xiaokui Xiao, Gabriel Bender, Michael Hay, and Johannes Gehrke. ireduct: differential privacy with reduced relative errors. SIGMOD '11, pages 229–240, New York, NY, USA, 2011. ACM.

[66] Xiaokui Xiao, Guozhang Wang, and J. Gehrke. Differential privacy via wavelet transforms. *Knowledge and Data Engineering, IEEE Transactions on*, 23(8):1200 –1214, aug. 2011.

[67] Yonghui Xiao, Li Xiong, and Chun Yuan. Differentially private data release through multidimensional partitioning. In *Proceedings of the 7th VLDB conference on Secure data management*, SDM'10, pages 150–168, Heidelberg, 2010. Springer-Verlag.

[68] Jia Xu, Zhenjie Zhang, Xiaokui Xiao, Yin Yang, and Ge Yu. Differentially private histogram publication. In *Proceedings of the 2012 IEEE 28th International Conference on Data Engineering*, pages 32–43, Washington, DC, 2012. IEEE Computer Society.

[69] Jianguo Yan, Dongli Yuan, Xiaojun Xing, and Qiuling Jia. Kalman filtering parameter optimization techniques based on genetic algorithm. In *Automation and Logistics, 2008. ICAL 2008. IEEE International Conference on*, pages 1717–1720, 2008.

[70] Hongliang Yu, Dongdong Zheng, Ben Y. Zhao, and Weimin Zheng. Understanding user behavior in large-scale video-on-demand systems. In *Proceedings of the 1st ACM SIGOPS/EuroSys European Conference on Computer Systems 2006*, EuroSys '06, pages 333–344, New York, NY, USA, 2006. ACM.