An Adaptive Approach to Real-Time Aggregate Monitoring with Differential Privacy

Liyue Fan and Li Xiong

Abstract—Sharing real-time aggregate statistics of private data is of great value to the public to perform data mining for understanding important phenomena, such as Influenza outbreaks and traffic congestion. However, releasing time-series data with standard differential privacy mechanism has limited utility due to high correlation between data values. We propose FAST, a novel framework to release real-time aggregate statistics under differential privacy based on filtering and adaptive sampling. To minimize the overall privacy cost, FAST adaptively samples long time-series according to the detected data dynamics. To improve the accuracy of data release per time stamp, FAST predicts data values at non-sampling points and corrects noisy observations at sampling points. Our experiments with real-world as well as synthetic data sets confirm that FAST improves the accuracy of released aggregates even under small privacy cost and can be used to enable a wide range of monitoring applications.

Index Terms—Statistical databases, differential privacy, time series

1 Introduction

Sharing real-time aggregate statistics of private data enables many important data mining applications. Consider the examples below:

- **Disease Surveillance**: A health care provider gathers data from individual visitors. The collected data, e.g. 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.
- **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.

A common scenario of such applications can be summarized by Fig. 1, where a trusted server gathers data from a large number of individual subscribers. The collected data may be then aggregated and continuously shared with other un-trusted entities for various purposes. The trusted server, i.e. publisher, is assumed to be bound by contractual obligations to protect the user's interests, therefore it must ensure that releasing the data does not compromise the privacy of any individual who contributed data. The goal of our work is to enable the publisher to share useful aggregate statistics over individual users continuously (aggregate time series) while guaranteeing their privacy.

The current state-of-the-art paradigm for privacy-preserving data publishing is *differential privacy*[1], 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.

Most existing works on differentially private data release deal with one-time release of static data [2]–[7]. In the applications we consider, data values at successive time stamps are highly correlated. A straightforward application of differential privacy mechanism which adds a Laplace noise to each aggregate value at each time stamp can lead to a very high perturbation error due to the composition theorem [8]. Few recent works [9]–[11] studied the problem of releasing time series or continual statistics. Rastogi and Nath [11] proposed an algorithm which perturbs d Discrete Fourier Transform (DFT) coefficients of the entire time series and reconstructs a released version with the Inverse DFT series. Since the entire time-series is required to perform those operations, it is not applicable to real-time applications. Dwork *et al*. [10] proposed a differentially private continual counter over a binary stream with a bounded error at each time step. Chan *et al*. [9] studied the same problem and concluded with a similar upper bound. 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.

- *The authors are with the Department of Mathematics and Computer Science, Emory University, Atlanta, GA 30322 USA.
E-mail: lfan3@emory.edu; lxiong@mathcs.emory.edu.*

*Manuscript received 4 Jan. 2013; revised 25 May 2013; accepted 10 June 2013. Date of publication 17 June 2013; date of current version 31 July 2014.
Recommended for acceptance by D. Agrawal.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier 10.1109/TKDE.2013.96*

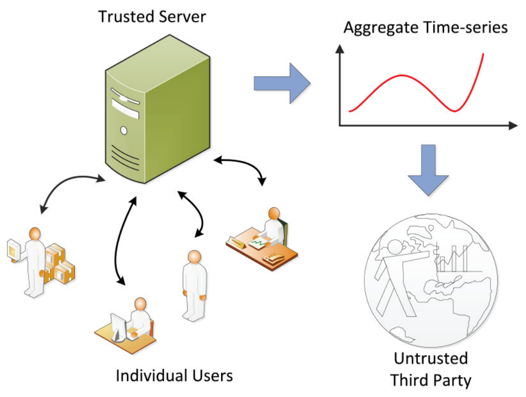


Fig. 1. Aggregate data sharing scenario.

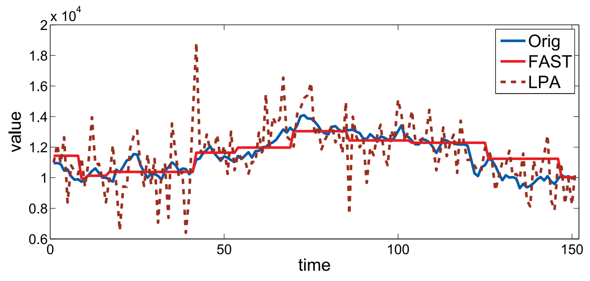


Fig. 2. FAST released series with linear data.

In this paper, we present FAST, a real-time framework with Filtering and Adaptive Sampling for differentially private Time-series monitoring. It uses sampling to query and perturb selected values in the time series with the differential privacy mechanism, and simultaneously uses filtering to dynamically predict the non-sampled values and correct the sampled values. The key innovation is that FAST utilizes feedback loops based on observed (perturbed) values to dynamically adjust the prediction/estimation model as well as the sampling rate. To this end, we examine two challenges in our system: predictability and controllability. The former raises the question: given a perturbed value, can we derive an estimate which is close to the true value and dynamically adjust the estimation model based on current observation? The latter imposes another question: suppose an accurate estimate can be derived at any time stamp, can we dynamically adjust the sampling rate according to the rate of data change? We extend our recent work [12] and present several contributions:

1) 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.
2) 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.
3) We provide formal analysis on filtering as well as the fixed-rate sampling process to understand their performance. We empirically evaluate our solution with both real-world and synthetic data sets. Our experiments show that FAST provides accurate data release and robust performance despite different data dynamics. Fig. 2 provides an example of the original *linear* times series, the released series by FAST, and that of the baseline LPA algorithm (introduced in Section 3.3). We observe that FAST released series retains much higher accuracy (i.e. data value, trend) than the LPA released series while providing the same level of privacy guarantee. With the real-time feature and improved utility, we believe that our solution is applicable to a wide range of monitoring applications.

The rest of the paper is organized as follows: Section 2 reviews previous works related to differential privacy, time series analysis, and filtering techniques. Section 3 provides the problem definition, preliminaries on differential privacy, and outlines of existing solutions. Section 4 presents an overview of FAST framework, as well as the technical details of filtering and sampling. Section 5 presents a set of empirical studies and results. Section 6 concludes the paper and states possible directions for future work.

2 Related Work

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

Differential privacy on static data. Dwork *et al.* [2] 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.* [1] proved the possibility of non-interactive data release satisfying differential privacy for queries with polynomial VC-dimension, such as predicate queries. Dwork *et al.* [13] further proposed more efficient algorithms to release private sanitization of a data set with hardness results obtained. The work of Hay *et al.* [3] improved the accuracy of a tree of counting queries through consistency check, which is done as a post-processing procedure after adding Laplace noise. This hierarchical structure of queries is referred to as *histograms* by several techniques [3], [5], [6], [7], [14], [15], where each level in the tree is an increasingly fine-grained summary of the data. A recent study [4], 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 [16], [17], sparse data [18], and set-valued data [19]. 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.

Time series analysis and privacy. Time series data is pervasively encountered in the fields of engineering, science, sociology, and economics. Various techniques [20], such as ARIMA modeling, exponential smoothing, ARAR, and Holt-Winters methods, have been studied for time-series forecasting. Papadimitriou *et al*. [21] 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 [11] 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 [9], [10] 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.

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 [22] in 1960. Since then, it has become widely applied to areas of signal processing [23] and assisted navigation systems [24]. It has also gained popularity in other areas of engineering. One particular application is to wireless sensor networks. Jain *et al*. [25] 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) [26] which linearizes the current mean and error covariance. Masreliez [27] 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 [28] 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 [29] introduced auxiliary particle filter and adaptation. Doucet *et al*. [30] proposed the optimal importance distribution, approximation, smoothing, and Rao-Blackwellization. A few tutorials, for instance[31], [32], on particle methods have been published and cover most sequential Monte Carlo algorithms in particle filtering to facilitate implementation.

3 Preliminaries

3.1 Problem Statement

We formally define an aggregate time series as follows:

Definition 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 $0 \le k < T$ 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 the paper. 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 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 $\mathbf{X}$:*

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

where δ is a user-specified constant (also referred to as sanitary bound in [4]) 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.

3.2 Differential Privacy

The privacy guarantee provided by FAST is *differential privacy* [1]. 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 (α-Differential Privacy [1]). *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 $\widetilde{D} \in Range(\mathcal{A})$,*

$$Pr[\mathcal{A}(D_1) = \widetilde{D}] \le e^{\alpha} \times Pr[\mathcal{A}(D_2) = \widetilde{D}], \tag{2}$$

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

Algorithm 1 Laplace Perturbation Algorithm(LPA)

```
Input: Raw series X, privacy budget $\alpha$
Output: Released series R
1: for each $k \in 0, 1, ..., T-1$ do
2:     $r_k \leftarrow$ perturb $x_k$ by $Lap(0, \frac{T}{\alpha})$;
```

The privacy parameter α, also called the *privacy budget* [8], 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*. [2] show that α-differential privacy can be achieved by adding i.i.d. noise $\tilde{N}$ to the query result $q(D)$:

$$\tilde{q}(D) = q(D) + \tilde{N}. \quad (3)$$

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* [2] 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 1 (Sequential Composition [8]). *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 [10] 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 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.

3.3 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 [11]. 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 appending $T-d$ 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: $IDFT(\mathbf{X})_j = \frac{1}{T}\sum_{i=0}^{T-1} e^{-\frac{2\pi\sqrt{-1}}{T}ji}x_i$.

Algorithm 2 Discrete Fourier Transform(DFT)

```
Input: Raw series X, privacy budget $\alpha$, parameter $d$
Output: Released time-series 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 $\mathbf{R} \leftarrow \mathbf{IDFT}(\mathbf{PAD}^T(\widetilde{\mathbf{F}}^d))$;
```

The number of coefficients d is a user-specified parameter. In our empirical study, we set $d = 20$ according to the original paper [11]. 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 paper.

4 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. Fig. 3 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 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.

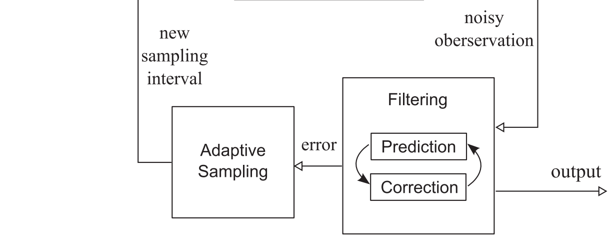


Fig. 3. FAST framework.

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:        $r_k \leftarrow prior$;
```

- 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 **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 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 2. *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.

4.1 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:

$$x_k = x_{k-1} + \omega \tag{4}$$

$$\omega \sim \mathcal{N}(0, Q). \tag{5}$$

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:

$$z_k = x_k + \nu \tag{6}$$

$$\nu \sim Lap(0, 1/\alpha_0), \tag{7}$$

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:

$$\hat{x}_k = E(x_k|\mathbb{Z}^k), \tag{8}$$

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:

$$f(x_k|\mathbb{Z}^k) = \frac{f(x_k|\mathbb{Z}^{k-1})f(z_k|x_k)}{f(z_k|\mathbb{Z}^{k-1})}, \tag{9}$$

where the prior and the normalizing constant are given by:

$$f(x_k|\mathbb{Z}^{k-1}) = \int f(x_{k-1}|\mathbb{Z}^{k-1})f(x_k|x_{k-1})dx_{k-1} \tag{10}$$

$$f(z_k|\mathbb{Z}^{k-1}) = \int f(x_k|\mathbb{Z}^{k-1})f(z_k|x_k)dx_k\ . \tag{11}$$

In general, Equations (9-11) 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.

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: $P_k \leftarrow (1 - K_k)P_k^-$;
```

4.1.1 Gaussian Approx. of Measurement Noise

In our previous work [12], we propose to model the Laplace noise ν with an approximate, white Gaussian error

$$\nu \sim \mathcal{N}(0, R) \tag{12}$$

and therefore the estimation of $\hat{x}_k$ in Equation (8) can be solved with the classic Kalman filter [22].

The Kalman Filter. At time stamp k, the prior state estimate $\hat{x}_k^-$ is made according to the process model in Equation (4) and is related to the posterior estimate of the previous step:

$$\hat{x}_k^- = \hat{x}_{k-1} \; . \tag{13}$$

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

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - \hat{x}_k^-) \; . \tag{14}$$

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

$$P_k^- = E[(x_k - \hat{x}_k^-)(x_k - \hat{x}_k^-)^T] \tag{15}$$
$$P_k = E[(x_k - \hat{x}_k)(x_k - \hat{x}_k)^T] \; . \tag{16}$$

By the Gaussian assumption regarding ω and ν and given the prior error variance P_k^- at time stamp k, we can substitute Equation (14, 15) into Equation (16) and apply the gradient descendant method to minimize P_k. Therefore, we obtain an optimal value for K_k as

$$K_k = P_k^-(P_k^- + R)^{-1} \tag{17}$$

and thus the optimal P_k is

$$P_k = (1 - K_k)P_k^- \; . \tag{18}$$

Similarly, given P_k, we can easily project the prior error variance at $k+1$ according to Equation (4):

$$P_{k+1}^- = P_k + Q \; . \tag{19}$$

The classic Kalman filter recursively performs two operations: *Prediction* and *Correction*, which correspond to prior and posterior estimation respectively. Algorithms 4, 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 (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, which is available in the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/96, for the detailed least square analysis.

Theorem 3 (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:*

$$var(x_k - \hat{x}_k) = \frac{R^2[var(x_{k-1} - \hat{x}_{k-1}) + Q]}{(P_k^- + R)^2} + \frac{2P_k^{-2}T^2}{(P_k^- + R)^2\alpha^2} \tag{20}$$

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

Proof. See Appendix A, available online. □

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 Algorithms 4, 5.

4.1.2 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

Algorithm 6 PFPredict(k)

```
Input: Particles at time $k-1$ $\{x^i_{k-1}, \pi^i_{k-1}\}^N_{i=1}$
Output: Prior estimate $\hat{x}^-_k$
1: for each $i \in 1, ..., N$ do
2:     draw $x^i_k \sim f(x_k|x^i_{k-1})$
3: $\hat{x}^-_k \leftarrow \frac{1}{N}\sum^N_{i=1} x^i_k$
```

posterior estimation based on the Sampling-Importance-Resampling(SIR) particle filter, which is also known as bootstrap filtering [28] and condensation algorithm [33].

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

$$f(x_k|\mathbb{Z}^k) = \sum^N_{i=i} \pi^i_k \delta(x_k - x^i_k), \tag{21}$$

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

The weights $\{\pi^i_k\}^N_{i=1}$ are chosen according to the importance sampling method, where particles $\{x^i_k\}^N_{i=1}$ 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 [34]. 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 derived:

$$\pi^i_k \propto \pi^i_{k-1} \frac{f(z_k|x^i_k) f(x^i_k|x^i_{k-1})}{q(x^i_k|x^i_{k-1}, z_k)} \ . \tag{22}$$

According to Arulampalam et al [31], it is offen convenient to choose the importance density $q(\cdot)$ to be the prior density $f(x_k|x^i_{k-1})$. Substituting it into Equation (22) then yields

$$\pi^i_k \propto \pi^i_{k-1} f(z_k|x^i_k), \tag{23}$$

where

$$f(z_k|x^i_k) = f(\nu = z_k - x^i_k) \tag{24}$$

and ν follows the Laplace distribution in Equation (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 [31]. Since $\pi^i_{k-1} = 1/N$ for every i after resampling, weights at time k can be simplified as follows:

$$\pi^i_k \propto f(z_k|x^i_k) \ . \tag{25}$$

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

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

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 $f(x_k|x^i_{k-1})$. Note that

Algorithm 7 PFCorrect(k)

```
Input: Particles $\{x^i_k\}^N_{i=1}$, noisy measurement $z_k$
Output: Posterior estimate $\hat{x}_k$
1: for each $i \in 1, ..., N$ do
2:     assign particle weight $\pi^i_k$ according to (25)
3: normalize $\{\pi^i_k\}^N_{i=1}$
4: $\hat{x}_k \leftarrow \sum^N_{i=1} \pi^i_k x^i_k$
5: resample from $\{x^i_k, \pi^i_k\}^N_{i=1}$
```

$f(x_k|x^i_{k-1})$ represents the distribution $\mathcal{N}(x^i_{k-1}, Q)$, according to Equation (4). Once the particles are drawn, the prior estimate can be given with the uncorrected weights (Line 3 in Algorithm 6):

$$\hat{x}^-_k = \sum^N_{i=1} \pi^i_{k-1} x^i_k = \frac{1}{N}\sum^N_{i=1} x^i_k \ . \tag{26}$$

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):

$$\hat{x}_k = \sum^N_{i=1} \pi^i_k x^i_k \ . \tag{27}$$

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 $\{x^i_0, \pi^i_0\}^N_{i=1}$ 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 (26-27), at time 0 and release the noisy measurement z_0. Particles $\{x^i_0\}^N_{i=1}$ are then uniformly drawn from the vicinity of z_0 and $\{\pi^i_0\}^N_{i=1}$ are initialized as $1/N$.

Accuracy. We refer the readers to [32] 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 Algorithms 6, 7 is $O(N)$ per time stamp. We will examine the trade-off between accuracy and run time of Algorithms 6, 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.

4.2 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

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
```

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 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, available online. 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. Fig. 4 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, 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 4 (Feedback Error). *At time step k_n ($0 \le k_n < T$), where the subscript n indicates the n-th sampling point ($0 \le n < M$), we define the feedback error E_{k_n} as follows:*

$$E_{k_n} = |\hat{x}_{k_n} - \hat{x}^-_{k_n}| / \max\{\hat{x}_{k_n}, \delta\} . \quad (28)$$

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.

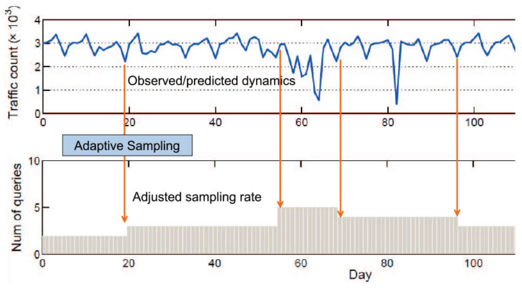


Fig. 4. Adaptive sampling with traffic data.

FAST adopts a PID controller, the most common form of feedback control [35], 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 (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

$$\Delta = C_p E_{k_n} + \frac{C_i}{T_i} \sum_{j=n-T_i+1}^{n} E_{k_j} + C_d \frac{E_{k_n} - E_{k_{n-1}}}{k_n - k_{n-1}} . \quad (29)$$

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, they are constrained by:

$$C_p, C_i, C_d \ge 0 \quad (30)$$

$$C_p + C_i + C_d = 1 . \quad (31)$$

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:

$$I' = max\{1, I + \theta(1 - e^{\frac{\Delta - \xi}{\xi}})\}, \quad (32)$$

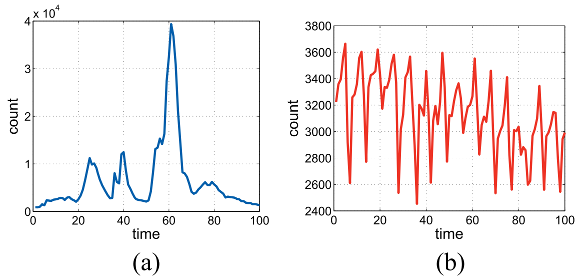


Fig. 5. Data snapshots. (a) *Flu* data. (b) *Traffic* data.

TABLE 1
FAST Parameters

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	$(10, 0.1)$

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
```

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 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.

5 Experiment

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.

- **Linear** is a synthetic series generated by our process model in Equation (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.

Fig. 5 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* 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 1.

5.1 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

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

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

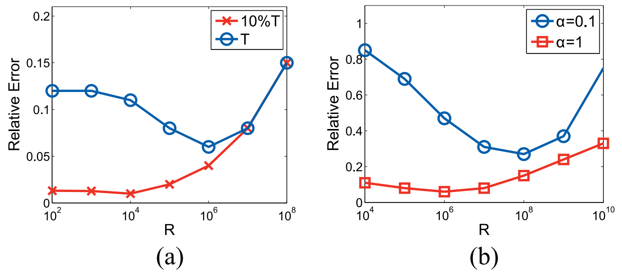


Fig. 6. Choice of R in the Kalman filter. (a) R vs. T. (b) R vs. α.

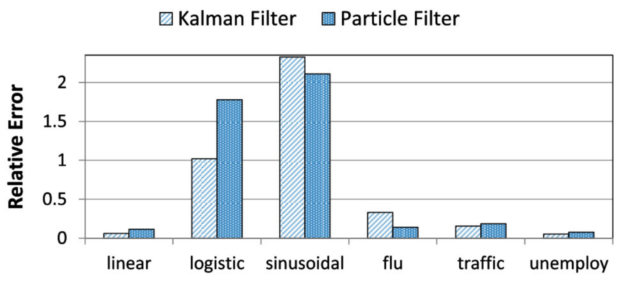


Fig. 8. Comparison of two filtering algorithms.

Gaussian measurement noise we have proposed to approximate the Laplace perturbation noise. Fig. 6(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. Fig. 6(b) plots the utility with various R values when α set to 0.1 versus 1. As can be seen in Fig. 6(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 Fig. 6(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. Fig. 7(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 Fig. 7(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.

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 Fig. 8. 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.

5.2 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. Fig. 9(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 Fig. 9(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.

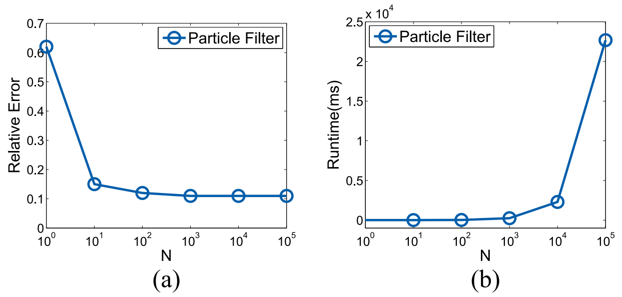


Fig. 7. Choice of N in particle filter. (a) Accuracy. (b) Runtime.

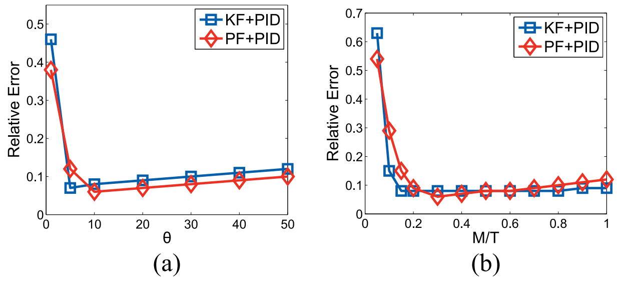


Fig. 9. Choice of adaptive sampling params. (a) Choice of θ. (b) Choice of M.

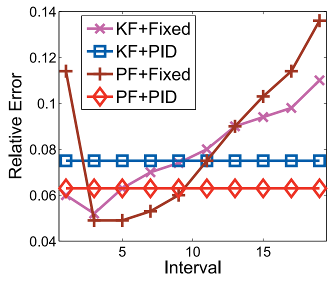


Fig. 10. Fixed-rate sampling vs. adaptive sampling.

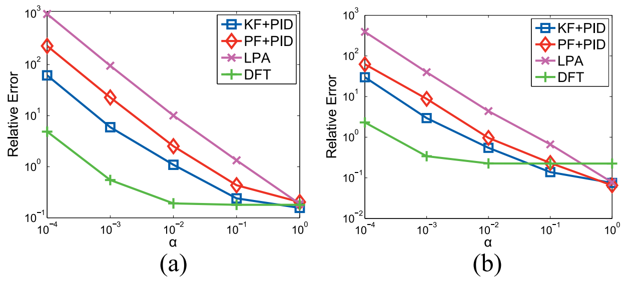


Fig. 12. Utility vs. privacy with real-world datasets. (a) Traffic data. (b) Unemploy data.

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 of adaptive sampling, denoted as PID in Fig. 10, 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 Fig. 10. 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.

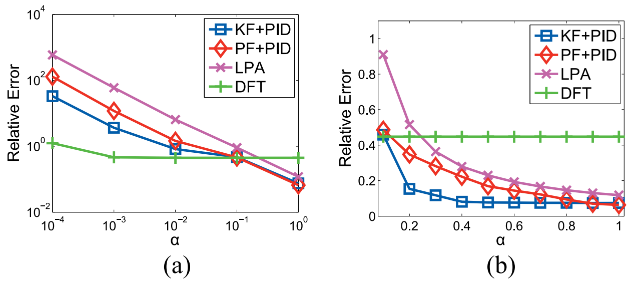


Fig. 11. Utility vs. privacy with linear data. (a) Overall performance. (b) Performance with larger budget.

5.3 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. Fig. 11 shows the empirical study conducted with the *linear* data set. Fig. 11(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 $\alpha \in [10^{-1}, 1]$. Fig. 11(b) presents a closer look at this privacy budget interval, where FAST algorithms outperform both existing methods, providing high utility 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. Fig. 12 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 Fig. 12(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 Fig. 12(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.

5.4 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.

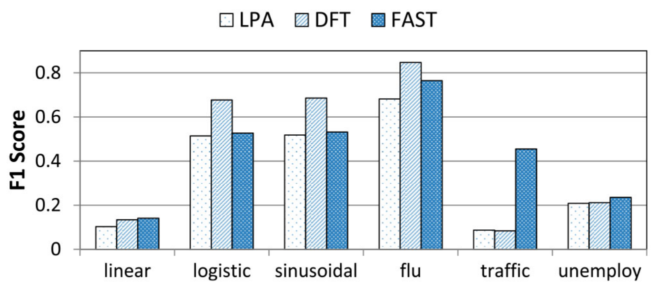


Fig. 13. F1 Metric of Outbreak Detection.

There have been extensive studies on effective methods in epidemic outbreak detection and various definitions of an outbreak/signal period have been adopted [36]–[38]. We take a simplified interpretation of outbreak similar to Pelecanos *et al*. [38] 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. Fig. 13 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 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. Fig. 14 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 Fig. 13. 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.

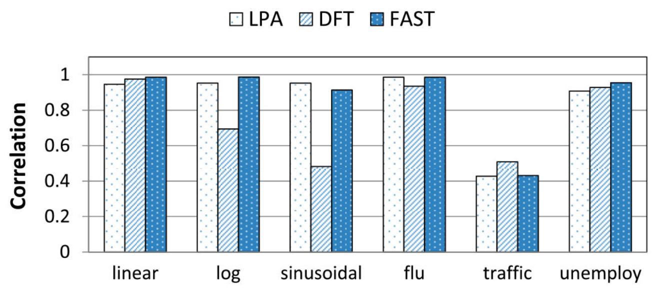


Fig. 14. Spearman's rank correlation between original series and released series.

6 Conclusion and Discussion

We have proposed FAST, an adaptive framework with filtering and sampling for monitoring real-time aggregates under differential privacy. The key innovation is that our approach 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. Our 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 our adaptive approach improves utility of time-series release and has excellent performance even under small privacy cost.

FAST framework can be easily generalized to monitor other aggregates supported by differential privacy, such as *average*, *sum*, *min*, and *max*, with the perturbation mechanism as well as the state-space model adapted to the specific type of query and its global sensitivity. Note that certain types of analysis, such as *min* and *max*, are in general difficult under differential privacy, since they rely only on very few records in the underlying database. Future work directions may include the study of alternative differential privacy mechanisms, such as geometric mechanism [39] and exponential mechanism [40], and investigate their privacy-utility tradeoff under FAST framework.

Acknowledgments

This research was supported in part by US NSF under grant CNS-1117763 and in part by AFOSR under grant FA9550-12-1-0240. The authors would like to thank anonymous reviewers for their comments and suggestions.

References

[1] A. Blum, K. Ligett, and A. Roth, "A learning theory approach to non-interactive database privacy," in *Proc. 40th Annu. ACM STOC*, New York, NY, USA, 2008.
[2] C. Dwork, F. Mcsherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Proc. 3rd TCC*, New York, NY, USA, 2006, pp. 265–284.
[3] M. Hay, V. Rastogi, G. Miklau, and D. Suciu, "Boosting the accuracy of differentially private histograms through consistency," *PVLDB*, vol. 3, no. 1–2, pp. 1021–1032, Sept. 2010.
[4] X. Xiao, G. Bender, M. Hay, and J. Gehrke, "iReduct: Differential privacy with reduced relative errors," in *Proc. ACM SIGMOD*, Athens, Greece, 2011, pp. 229–240.
[5] X. Xiao, G. Wang, and J. Gehrke, "Differential privacy via wavelet transforms," *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 8, pp. 1200–1214, Aug. 2011.
[6] Y. Xiao, L. Xiong, and C. Yuan, "Differentially private data release through multidimensional partitioning," in *Proc. 7th VLDB Conf. SDM*, vol. 6358. Singapore, 2010, pp. 150–168.
[7] J. Xu, Z. Zhang, X. Xiao, Y. Yang, and G. Yu, "Differentially private histogram publication," in *Proc. IEEE 28th ICDE*, Washington, DC, USA, 2012.
[8] F. McSherry, "Privacy integrated queries: An extensible platform for privacy-preserving data analysis," in *Proc. ACM SIGMOD*, New York, NY, USA, 2009.

[9] T.-H. H. Chan, E. Shi, and D. Song, "Private and continual release of statistics," in *Proc. 37th ICALP (2)*, Bordeaux, France, 2010, pp. 405–417, LNCS 6199.

[10] C. Dwork, M. Naor, T. Pitassi, and G. N. Rothblum, "Differential privacy under continual observation," in *Proc. 42nd ACM STOC*, New York, NY, USA, 2010, pp. 715–724.

[11] V. Rastogi and S. Nath, "Differentially private aggregation of distributed time-series with transformation and encryption," in *Proc. ACM SIGMOD*, Indianapolis, IN, USA, 2010.

[12] L. Fan and L. Xiong, "Real-time aggregate monitoring with differential privacy," in *Proc. 21st ACM Int. CIKM*, New York, NY, USA, 2012.

[13] C. Dwork, M. Naor, O. Reingold, G. N. Rothblum, and S. Vadhan, "On the complexity of differentially private data release: Efficient algorithms and hardness results," in *Proc. 41st Annu. ACM STOC*, Bethesda, MD, USA, 2009, pp. 381–390.

[14] C. Li, M. Hay, V. Rastogi, G. Miklau, and A. Mcgregor, "Optimizing linear counting queries under differential privacy," in *Proc. 29th PODS*, Indianapolis, IN, USA, 2010, pp. 123–134.

[15] C. Li and G. Miklau, "An adaptive mechanism for accurate query answering under differential privacy," *PVLDB*, vol. 5, no. 6, pp. 514–525, Feb. 2012.

[16] A. Korolova, K. Kenthapadi, N. Mishra, and A. Ntoulas, "Releasing search queries and clicks privately," in *Proc. 18th Int. Conf. WWW*, New York, NY, USA, 2009, pp. 171–180.

[17] Y. Hong, J. Vaidya, H. Lu, and M. Wu, "Differentially private search log sanitization with optimal output utility," in *Proc. 15th Int. Conf. EDBT*, Berlin, Germany, 2012.

[18] G. Cormode, C. M. Procopiuc, D. Srivastava, and T. T. L. Tran, "Differentially private summaries of sparse data," in *Proc. ICDT*, Berlin, Germany, 2012.

[19] R. Chen, N. Mohammed, B. C. M. Fung, B. C. Desai, and L. Xiong, "Publishing set-valued data via differential privacy," *PVLDB*, vol. 4, no. 11, pp. 1087–1098, Aug. 2011.

[20] P. Brockwell and R. Davis, *Introduction to Time Series and Forecasting.* New York, NY, USA: Springer, 2002.

[21] S. Papadimitriou, F. Li, G. Kollios, and P. S. Yu, "Time series compressibility and privacy," in *Proc. 33rd Int. conf. VLDB*, 2007, pp. 459–470.

[22] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Trans. ASME J. Basic Eng.*, vol. 82, pp. 35–45, Mar. 1960.

[23] R. G. Brown and P. Y. C. Hwang, *Introduction to Random Signals and Applied Kalman Filtering*, vol. 2. New York, NY, USA: Wiley, 1997.

[24] Y. Bar-Shalom, X.-R. Li, and T. Kirubarajan, "State estimation in discrete-time linear dynamic systems," in *Estimation with Applications to Tracking and Navigation*. New York, NY, USA: Wiley, 2002, pp. 199–266.

[25] A. Jain, E. Y. Chang, and Y.-F. Wang, "Adaptive stream resource management using Kalman filters," in *Proc. ACM SIGMOD*, New York, NY, USA, 2004, pp. 11–22.

[26] H. Sorenson, *Kalman Filtering: Theory and Application.* New York, NY, USA: IEEE Press, 1985.

[27] C. Masreliez, "Approximate non-gaussian filtering with linear state and observation relations," *IEEE Trans. Autom. Control*, vol. 20, no. 1, pp. 107–110, Feb. 1975.

[28] N. Gordon, D. Salmond, and A. Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," *IEEE Proc. F Radar Signal Process.*, vol. 140, no. 2, pp. 107–113, Apr. 1993.

[29] M. K. Pitt and N. Shephard, "Filtering via simulation: Auxiliary particle filters," *J. Amer. Statist. Assoc.*, vol. 94, no. 446, pp. 590–599, 1999.

[30] A. Doucet, S. Godsill, and C. Andrieu, "On sequential Monte Carlo sampling methods for Bayesian filtering," *Statist. Comput.*, vol. 10, no. 3, pp. 197–208, Jul. 2000.

[31] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for on-line non-linear/non-Gaussian Bayesian tracking," *IEEE Trans. Signal Process.*, vol. 50, no. 2, pp. 174–188, Feb. 2001.

[32] A. Doucet and A. M. Johansen, "A tutorial on particle filtering and smoothing: Fifteen years later," in *Handbook of Nonlinear Filtering*, vol. 12. Oxford, NY, USA: Oxford Univ. Press, 2009, pp. 656–704.

[33] J. MacCormick and A. Blake, "A probabilistic exclusion principle for tracking multiple objects," in *Proc. 7th IEEE ICCV*, vol. 1. Kerkira, Greece, 1999, pp. 572 –578.

[34] A. Doucet, N. de Freitas, N. Gordon, and A. Smith, *Sequential Monte Carlo Methods in Practice.* New York, NY, USA: Springer, 2001.

[35] M. King, *Process Control: A Practical Approach*. Chichester, U.K.: Wiley, 2011.

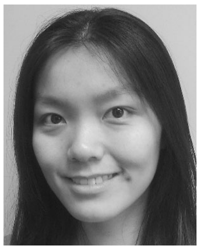
[36] M. Wagner *et al.*, "The emerging science of very early detection of disease outbreaks," *J. Public Health Manag. Pract.*, vol. 7, no. 6, pp. 51–59, 2001.

[37] K. P. Kleinman and A. M. Abrams, "Assessing surveillance using sensitivity, specificity and timeliness," *Statist. Meth. Med. Res.*, vol. 15, no. 5, pp. 445–464, 2006.

[38] A. Pelecanos, P. Ryan, and M. Gatton, "Outbreak detection algorithms for seasonal disease data: A case study using ross river virus disease," *BMC Med. Inform. Decis. Mak.*, vol. 10, Article 74, Nov. 2010.

[39] A. Ghosh, T. Roughgarden, and M. Sundararajan, "Universally utility-maximizing privacy mechanisms," in *Proc. 41st Annu. ACM STOC*, New York, NY, USA: ACM, 2009, pp. 351–360.

[40] F. McSherry and K. Talwar, "Mechanism design via differential privacy," in *Proc. 48th Annu. IEEE Symp. FOCS*, Washington, DC, USA: IEEE Computer Society, 2007, pp. 94–103.



Liyue Fan is a Ph.D. candidate with the Department of Mathematics and Computer Science at Emory University, Atlanta, GA, USA. Prior to Emory, she received the B.S. degree in mathematics at Zhejiang University, China. Her current research interests include data privacy, data mining, machine learning, data integration, temporal, and spatial databases.

Li Xiong is an Associate Professor in the Department of Mathematics and Computer Science and the Department of Biomedical Informatics, at Emory University, Atlanta, GA, USA, where she directs the Assured Information Management and Sharing (AIMS) Research Group. She received the B.S. degree from the University of Science and Technology of China, the M.S. degree from Johns Hopkins University, Baltimore, MD, USA, and the Ph.D. degree from Georgia Institute of Technology, Atlanta, GA, USA, all in computer science. She also worked as a Software Engineer with IT industry for several years prior to pursuing her doctorate. Her current research interests include data privacy and security, distributed data management, and bio and health informatics. She is a recent recipient of the Career Enhancement Fellowship by Woodrow Wilson Foundation. Her research is supported by US NSF, AFOSR, a Cisco Research Award, and an IBM Faculty Innovation Award.