Differential Private Stream Processing of Energy Consumption

Ferdinando Fioretto
University of Michigan
Ann Arbor, MI
fioretto@umich.edu

Pascal Van Hentenryck
Georgia Institute of Technology
Atlanta, GA
pvh@isye.gatech.edu

ABSTRACT

A number of applications benefit from continuously releasing streams of personal data statistics. The process, however, poses significant privacy risks. Motivated by an application in energy systems, this paper presents OptStream, a novel algorithm for releasing differential private data streams. OptStream is a 4-step procedure consisting of sampling, perturbation, reconstruction, and post-processing modules. The *sampling* module selects a small set of points to access privately in each period of interest, the *perturbation* module adds noise to the sampled data points to guarantee privacy, the *reconstruction* module re-assembles the non-sampling data points from the perturbed sampled points, and the *post-processing* module uses convex optimization over the private output of the previous modules, as well as the private answers of additional queries on the data stream, to ensure consistency of the data's salient features. OptStream is used to release a real data stream from the largest transmission operator in Europe. Experimental results show that OptStream not only improves the accuracy of the state-of-the-art by at least one order of magnitude on this application domain, but it is also able to ensure accurate load forecasting based on the private data.

1. INTRODUCTION

Differential privacy [10] has emerged as a robust framework to release datasets while limiting the identification of participating individuals. Informally, it ensures, with high probability, that what can be learned about an individual in a differential private dataset is limited to what could be learned about the individual in the dataset, not including her data. In the age of the Internet of Things, however, datasets are becoming increasingly dynamic. Motivated by applications in health monitoring, traffic management, and social networks, there has been increasing interest in releasing streams of data in a private manner [11, 9, 15, 8, 6] where aggregated statistics are continuously reported.

Two common approaches for continuous data release are the *event-level* and *user-level* privacy models [11]. The former focuses on protecting a single *event*, while the latter aims at protecting *all* the events associated with a user, i.e., it focuses on protecting the user's presence in the dataset. Additionally, Kellaris et al. [19] proposed the notion of w-event privacy to attain a balance between event-level and user-level privacy, trading off utility and privacy to protect event sequences within a time window of w time steps.

This paper is motivated by a desire to release private streams of *loads* (energy demands) in transmission systems, protecting changes in consumer loads up to some desired amount within critical time intervals. Although customer identities are typically considered public information (e.g., each facility is served by some energy provider), their loads can be highly sensitive as they may reveal economic activities of grid customers (e.g., the consumption of a plant).

To release continuous statistics on streams of load data, this paper combines two well-studied models of differential privaty: the w-event privacy [19], which protects a stream within a critical time window, and the α-*indistinguishability model* [5], which protects the variation of the users' quantities up to a given amount α. Moreover, since the private data streams are typically inputs to complex analytic tasks, e.g., demand forecasting algorithms [24] and optimal power flows [25], the mechanism accuracy is critical. As shown later in the paper, existing algorithms for time series seem to fall short in this respect.

To remedy this limitation, this paper proposes a novel algorithm, called OptStream, for releasing continuous statistics on stream data under the combined α-indistinguishability and the w-event privacy models. OptStream is a 4-step procedure consisting of sampling, perturbation, reconstruction, and post-processing modules. The *sampling* module selects a small set of points for private measurement in each period of interest, the *perturbation* module induces noise to the sampled data points to guarantee privacy, the *reconstruction* module reconstructs the non-sampling data points from the perturbed sampled points, and the *post-processing* module uses convex optimization over the private output of the previous modules, as well as the private answers of additional queries on the data stream, to ensure consistency of salient features of the data. OptStream is also generalized to answer queries over hierarchical streams, allowing data curators to monitor, at the same time, streams produced by energy profile data at different levels of aggregations.

OptStream is evaluated on real datasets from *Réseau de Transport d'Électricité*, the French transmission operator and the largest in Europe. The dataset contains the energy consumption for one year at a granularity of 30 minutes. OptStream is also compared with state-of-the-art algorithms adapted to the combined privacy model. Experimental results show that OptStream improves the accuracy of state-of-the-art algorithms by at least one order of magnitude for this application domain. The effectiveness of the proposed algorithm is measured, not only in terms of the error between the reported private streams and the original stream, but also in the accuracy of a load forecasting algorithm based on the private data. Finally, the paper also shows that the sampling and optimization-based post-processing are critical in achieving the desired performance and that the improvements are also observed when releasing hierarchical stream data.

The rest of this paper is organized as follows. Section 2 discusses the stream model, summarizes the privacy goals of this work, and reviews the notion of differential privacy over streams. Section 3 describes the proposed framework (OptStream), and describes the design choices of each of the OptStream modules. Section 4 analyses the accuracy of the proposed framework and shows how it can reduce the error introduced to preserve privacy when compared to a standard solution. Section 5 discusses how OptStream can be extended to handle hierarchical stream data. Section 6 performs a comprehensive experimental analysis on real data streams from energy load profiles. Section 7 discusses the related work and, finally, Section 8 concludes the work.

2. PRELIMINARIES

2.1 Stream Data Model

A *data stream* D is an infinite sequence of elements in the *data universe* $\mathscr{U} = \mathcal{I} \times \mathcal{R} \times \mathcal{T}$, where $\mathcal{I}$ denotes the set of users' identifiers, $\mathcal{R}$ the set of possible *events* (i.e., a data item generated by the user), and $\mathcal{T}$ is a possibly unbounded set of time steps. Each tuple (i, r, t) describes an event that occurred at time t in which user i reported value r.

Time is represented through discrete steps $\mathcal{T} = \{1, 2, \ldots\}$. Each user transmits an aggregate summary of the stream periodically (e.g., every 30 minutes) and the discrete time step k represents an aggregate (e.g., the sum or the average) of all values describing the event occurring in the k-th time period. This setting is consistent with other stream data analysis work [11, 19] and implies that a tuple (i, r, t) describes an aggregate behavior of user i during the time period t. The following example illustrates such behavior.

Example 2.1. *Consider a data stream system that collects power consumption data from users of an electric company distributed on a wide geographical region. Users correspond to facilities (such as homes, hospitals, industrial buildings) or electrical substations, transmitting their power consumption at regular intervals (e.g., every* 30 *minutes). The value* $r_t \in \mathcal{R}$ *transmitted by a user at time* t *is a real number denoting the average amount of power (in MegaWatts) it requires during time interval* t.

In a data stream D, tuples are ordered by arrival time. If a tuple (i, r, t) arrives after a tuple (i', r', t'), then $t \geq t'$. In the following, $D[t]$ denotes a *stream prefix*, i.e., the sequence $D_1, \ldots, D_t$ of all tuples observed on or before time t.

2.2 Settings and Privacy Goal

The data curator receives updates from an unbounded data stream D at discrete time steps. At time t, the curator collects a dataset D_t of tuples (u, r, t) where every row corresponds to a *unique* user. For convenience, r^t_u denotes the value r reported by user u at time t. This paper focuses on applications where all reported values are numeric and hence $r^t_u \in \mathbb{R}$. Let $\mathcal{D}$ be the set of all datasets describing collection of tuples in $\mathscr{U}$. A count query is a function $Q : \mathcal{D} \to \mathbb{R}$ that reports the aggregated value associated with each user in the dataset, i.e., $Q(D_t) = \sum_u r^t_u$.

In our target applications, the data curator is interested in publishing every element x_t of the stream $\mathbf{x}$ observed within an arbitrary recurring period of w time steps. We call an w-*period* a set of w contiguous time steps $t{-}w{+}1, \ldots, t$ $(w \geq 1)$. Thus, the answers to each query $Q(D_t)$ are generated in real time within a window of w time steps, i.e., if t is the first step in the time window, then the value x_t is produced prior any tuple $(\_, \_, t + w)$ is reported. As a result, this paper adopts the w-event privacy framework, introduced in [19].

The motivating application requires the private release of load streams. Customer identities are assumed to be public information, as every facility consumes power, but their load fluctuations may be highly sensitive. Indeed. changes in consumption may indirectly reveal production levels and hence strategic investments, decreases in sales, and other similar information. Such changes should not be revealed within some application-specific period of w time steps. Thus, within each w-period, the privacy goal of the data curator is to protect observed increases or decreases of power consumptions. More precisely, consider r^t_u and ${r'}^t_u$ be two distinct values that may be reported by user u at time t and that satisfy $|r^t_u - {r'}^t_u| \leq \alpha$ for some positive real value α. An attacker should not be able to confidently determine that u reported value r^t_u instead of value ${r'}^t_u$ and vice-versa.

2.3 Differential Privacy

We first review the definition of *differential privacy* [10], which is defined over datasets rather than stream data.

Definition 2.1 (Differential Privacy). *Let* $\mathcal{D}$ *be a set of datasets,* $\sim \subseteq \mathcal{D}^2$ *be a symmetric binary relation (called adjacency relation). An algorithm* $\mathcal{A}$ *that takes in input a dataset and outputs an element from a set of possible responses* $\mathcal{O}$ *achieves* ϵ*-differential privacy if, for all sets* $O \subseteq \mathcal{O}$*, and all datasets* $D, D' \in \mathcal{D}$*, such that* $D \sim D'$*:*

$$\frac{Pr[\mathcal{A}(D) \in O]}{Pr[\mathcal{A}(D') \in O]} \leq \exp(\epsilon). \qquad (1)$$

The level of privacy is controlled by the parameter $\epsilon \geq 0$, called the *privacy budget*, with values close to 0 denoting strong privacy. Differential privacy focuses on protecting the privacy of individual users participating to a dataset. An algorithm satisfying Equation (1) prevents an attacker that gains access to the output of the algorithm from learning anything substantial about any individual.

The adjacency relation $\sim$ captures the aspects of the private data D that are considered sensitive. A commonly adopted adjacency relation is the 1-hamming distance: $D \sim D' \;\Leftrightarrow\; \|D - D'\| \leq 1$ defining the notion of *neighboring datasets* in differential privacy.

This work focuses on a more general adjacency relation that captures the distance between reported quantities whose magnitudes the mechanism must protect up to some given value $\alpha > 0$, called the *indistinguishability level*. This generalized definition of differential privacy has been adopted in several applications and has sound theoretical foundations (e.g., [20, 2, 5][1]). Consider a dataset D to which n individuals contribute their real-valued data $r_i \in \mathbb{R}$, i.e., $D = (r_1, \ldots, r_n) \in \mathbb{R}^n$. For $\alpha > 0$, an adjacency relation

[1]We refer the interested reader to [5] for a broad analysis of indistinguishablility when differential privacy uses different notions of distance.

that captures the participation of a single individual to the aggregating scheme is:

$$D \sim_\alpha D' \;\Leftrightarrow\; \exists i \text{ s.t. } |r_i - r'_i| \leq \alpha \text{ and } r_j = r'_j, \forall j \neq i. \quad (2)$$

Such adjacency definition, in our target domain, is useful to hide increases or decreases of loads up to some quantity α.

Differential Privacy on Streams. This work also adopts the w-event differential privacy [19], which has become a standard privacy concept for streams (see [26, 12, 2, 3, 4]). For a given $\alpha > 0$, two datasets D_t, D'_t at a time stamp t are called *neighbors*, which is denoted by $D_t \sim_\alpha D'_t$, if they satisfy the adjacency relation in Equation 2.

The algorithms presented in this paper operate on stream prefixes. For a given $\alpha > 0$, two data streams prefixes $D[t], D'[t]$ are said *w-neighbors*, written $D[t] \sim_w D'[t]$, if *(i)* for each D_i, D'_i with $i \in [t]$, $D_i \sim_\alpha D'_i$, and *(ii)* for each D_i, D'_i, D_j, D'_j, with $i < j \in [t]$ and $D_i \neq D'_i, D_j \neq D'_j$, it must be the case that $j - i + 1 \leq w$ holds. In other words, two data streams are w-neighbors if their elements are *pairwise* neighbors and all the differing elements are with a time window of up to w time steps.

Definition 2.2 (w-privacy). *Let $\mathcal{A}$ be a randomized algorithm that takes as input a stream prefix $D[t]$ of arbitrary size and outputs an element from a set of possible output sequences $\mathcal{O}$. $\mathcal{A}$ satisfies w-event ϵ-differential privacy (w-privacy for short) if, for all sets $O \subseteq \mathcal{O}$, all w-neighboring stream prefixes $D[t], D'[t]$, and all t:*

$$\frac{Pr[\mathcal{A}(D[t]) \in O]}{Pr[\mathcal{A}(D'[t]) \in O]} \leq \exp(\epsilon). \quad (3)$$

An algorithm satisfying w-privacy protects the sensitive information that could be disclosed from a sequence of some finite length w. When $w = 1$, w-privacy degenerates in notion of event-level privacy [11] that protects the disclosure of events in a single time step.

In summary, this paper combines w-event privacy [19] and a metric-based generalization of differential privacy [5] in order to meet the requirements of the motivating application.

Fundamental Properties of Differential Privacy. Differential privacy satisfies several important properties, including composability and immunity to post-processing. All the properties discussed below apply to w-privacy.

Composability ensures that a combination of differential private algorithms preserve differential privacy [12].

Theorem 2.1 (Sequential Composition). *The composition $\mathcal{A}(D[t]) = (\mathcal{A}_i(D[t]), \ldots, \mathcal{A}_k(D[t]))$, of a collection $\{\mathcal{A}_i\}_{i=1}^k$ of ϵ_i-differential private algorithms satisfies $(\sum_{i=1}^k \epsilon_i)$-differential privacy.*

Theorem 2.2 (Parallel Composition). *Let S_1 and S_2 be disjoint subsets of $\mathscr{U}$, and $\mathcal{A}$ be an ϵ-differential private algorithm. Then, computing $\mathcal{A}(D[t] \cap S_1)$ and $\mathcal{A}(D[t] \cap S_2)$ satisfies ϵ-differential privacy.*

These properties allow executing a sequence of differential private tasks and reason on their overall privacy guarantee.

Post-processing immunity ensures that privacy guarantees are preserved by arbitrary post-processing steps [12].

Theorem 2.3 (Post-Processing Immunity). *Let $\mathcal{A}$ be an ϵ-differential private algorithm and g be an (arbitrary) mapping from the set of possible output sequences $\mathcal{O}$ to an arbitrary set. Then, $g \circ \mathcal{A}$ is ϵ-differential private.*

A numerical query Q that maps a dataset to $\mathbb{R}^d$ can be made differential private by injecting random noise to its output. The amount of noise to be added depends on the *sensitivity* of the query, denoted by Δ_Q, and defined as

$$\Delta_Q = \max_{D[t] \sim_w D'[t]} \|Q(D[t]) - Q(D'[t])\|_1.$$

It is the maximum L_1 distance between the query outputs from any two w-neighboring data streams $D[t]$ and $D'[t]$.

The Laplace distribution with 0 mean and scale b has a probability density function $\text{Lap}(x|b) = \frac{1}{2b}e^{-\frac{|x|}{b}}$. It can be used to give an ϵ-differential private algorithm to answer numeric queries [10].

Theorem 2.4 (Laplace Mechanism ($\mathcal{M}_{\text{Lap}}$)). *Let Q be a numerical query that maps datasets to $\mathbb{R}^d$. The Laplace mechanism that outputs $Q(D[t]) + z$, where $z \in \mathbb{R}^d$ is a vector of i.i.d. samples drawn from a Laplace distribution $Lap\left(\frac{\Delta_Q}{\epsilon}\right)$ achieves ϵ-differential privacy.*

The Laplace mechanism with parameter w/ϵ achieve w-privacy [19]. We will use $\text{Lap}(\lambda)^d$ to denote the i.i.d. Laplace distribution over d dimensions with parameter λ.

This paper develops algorithms that satisfy Definition 2.2. The results discussed below satisfy the w-event privacy definition and *w-periods* represent windows of recurring w time steps in the data stream. For simplicity, the theoretical analysis of OptStream is derived using $\alpha = 1$ for the adjacency relation but the results generalize to arbitrary positive values. The behavior of OptStream for different indistinguishability levels are presented in Section 6. To simplify notation, the paper assumes that the data stream is a univariate discrete series $\mathbf{x} = \{x_t\}_{t=1}^\infty$ where each x_t represents the result of a count query $Q(D_t)$. When clear from the context, the paper uses ϵ-differential privacy to denote w-event privacy on w-periods.

3. OPTSTREAM FOR STREAM RELEASE

This section describes OptStream, a novel algorithm for private data stream release. OptStream consists of four steps: data sampling, perturbation, reconstruction of the non-sampled data points, and optimization-based post-processing. This section focuses on a single data stream release. An extension of the algorithm that targets hierarchical data streams is described in Section 5.

The algorithm takes as input a stream of real valued data points $\mathbf{x} = (x_1, x_2, \ldots)$ that corresponds to the answer of a numerical query on the stream data D and the size w of the w-period to protect. Its output is a data stream $\hat{\mathbf{x}} = \hat{x}_1, \hat{x}_2, \ldots$ where each $\hat{x}_t$ represents a private estimate of the real data x_t. The four steps can be summarized as follows:

1. *Sample* selects a set of k points for each w-period. Its goal is to perform a dimensionality reduction over the data stream which can be used to generate private answers to each query with low error.
2. *Perturb* adds noise to the sampled data points to guarantee privacy.
3. *Reconstruct* reconstructs the non-sampling data points from the perturbed sampled points. Its goal is to map

Algorithm 1: OptStream *data stream release*

```
input  : The data stream $\mathbf{x} = (x_1, x_2, \ldots)$, the size of
         the period $w$, the number of samples $k$ for
         each period, the privacy budgets $\epsilon_s, \epsilon_p, \epsilon_o$
output: A private data stream $\hat{x} = (\hat{x}_1, \hat{x}_2, \ldots)$
1 for $\ell = 1, 2, \ldots$ do
2     Let $t_a, t_b = \ell n, (\ell + 1)n$
3     Let $x_\ell = \mathbf{x}[t_a, t_b)$
4     $S_\ell = Sample(x_\ell, k, \epsilon_s)$
5     $\tilde{x}_{S_\ell} = Perturb(x_\ell[S_\ell], \epsilon_p)$
6     $\tilde{x}_\ell = Reconstruct(\tilde{x}_{S_\ell}, [t_a, t_b))$
7     $\hat{x}_\ell = PostProcess(\tilde{x}_\ell, \{Q_\mathbf{F}(x_\ell)\}_{\mathbf{F} \in \mathscr{F}}, \epsilon_o)$
8     release $\hat{x}_\ell$
```

the dimension-reduced data stream back to the original space, generating thus w data points.

4. *Post-process* uses the private output of the above modules, as well as the private answer to additional queries on the data stream, to ensure consistency of salient features of the data.

OptStream balances two types of errors: a *perturbation error*, introduced by the application of additive noise at the sampling points and a *reconstruction error* introduced by the reconstruction procedure at the non-sampled points. The higher the number of samples in an w-period the more perturbation error is introduced while the reconstruction error may be reduced, and vice-versa. Section 4 depicts the error brought by these two components and analyzes the number of samples that minimizes the error.

We now describe OptStream and adopt the following notations: Given an interval $[t_a, t_b]$, $\mathbf{x}[t_a, t_b]$ denotes the subsequence $(x_{t_a}, \ldots, x_{t_b})$, where we use ")" to indicate that the boundary element of the interval is not included in it. Similarly, given a set of time steps indexes I, $\mathbf{x}[I]$ denotes the collection of points $(x_i | i \in I)$.

OptStream is depicted in Algorithm 1. When a new w-period stream is processed, at time t, procedure *Sample* takes as input the sequence of values $x_\ell = (x_{t-w+1} \ldots x_t)$ – whose boundary index points are denoted t_a and t_b in line 2 – , the number k of data points to sample, and the portion $\epsilon_s \geq 0$ of the privacy budget ϵ to be used in the sampling process[2] (line 4). It outputs the set S_ℓ of time steps associated to the values of x_ℓ to sample. Procedure *Perturb* takes as input the k-ary vector $x_\ell[S_\ell]$ of sampled data points from x_ℓ – where $x_\ell[S_\ell] = (x_i | i \in S_\ell)$ – and outputs a noisy version $\tilde{x}_{S_\ell}$, using a portion $\epsilon_p > 0$ of the overall privacy budget (line 5). Procedure *Reconstruct* takes as input the k-ary data vector $\tilde{x}_{S_\ell}$ outputted by the perturbation step and the w-period range, and outputs a w-ary vector $\tilde{x}_\ell$ whose values are private estimates of the stream data in time steps $[t_a, t_b)$ (line 6). Finally, procedure *PostProcess* takes as input the vector of points $\tilde{x}_\ell$, additional *feature queries* $Q_\mathbf{F}(x_\ell)$ for each feature $\mathbf{F}$ in the set of data features $\mathscr{F}$ (which are defined in detail in Section 3.4) over the w-period stream data and uses a portion $\epsilon_o > 0$ of the overall privacy budget ϵ to compute a final estimate $\hat{x}_\ell$ of x_ℓ (line 7).

[2] ϵ_s could be 0 when the sampling processes does not require access to the real data to make a decision on the points to sample.

Theorem 3.1. *Let $\epsilon_s + \epsilon_p + \epsilon_o = \epsilon$. When the* Sample, *the* Perturb, *and the* Post-Process *procedures satisfy ϵ_s-, ϵ_p-, and ϵ_o-differential privacy respectively, Algorithm 1 satisfies ϵ-differential privacy.*

Any sampling, perturbation, and reconstruction algorithms can be used within Algorithm 1, provided that they achieve the intended purpose and satisfy the required privacy guarantees. The next section describes two variants of the sampling and reconstruction algorithms that can reduce the error (see Section 4) and perform well in our experimental analysis (see Section 6). The perturbation procedure is a standard application of the Laplace mechanism (Theorem 2.4) on a set of k points and parallel composition (Theorem 2.2). The post-processing step is described in Section 3.4.

3.1 The Sampling Procedures

The goal of the sampling procedure is to select k points in an w-period that can summarize the entire data stream period well. We investigate two possible strategies.

3.1.1 *Equally-Spaced Sampling*

A first strategy is to sample k equally spaced data points in the w-period. Since this approach does not look at the data stream, it does not consume a portion of the privacy budget ϵ, i.e., $\epsilon_s = 0$.

3.1.2 *AUC-Based Sampling*

The second strategy selects k out of w points in the w-period so to minimize the error between the points generated by linearly interpolation through the k selected points and the original data points.

Let $\xi^{[i,j]}(t)$ be the function describing a ray passing through points x_i and x_j, with $i, j \in [t_a, t_b)$. The *Area under the curve (AUC)-based sampling* procedure finds the combination of k points $\iota_1, \ldots, \iota_k$ in an w-period $[t_a, t_b)$ that minimizes the *AUC scoring function*:

$$\min_{\iota_1 < \iota_2 < \ldots < \iota_k} \sum_{j=1}^{k-1} \int_{\iota_j}^{\iota_{j+1}} \left| \xi^{[\iota_j, \iota_{j+1}]}(t) - x_t \right| dt \qquad (4)$$

Intuitively, the set of k points that minimizes (4) represents the set of points whose *reconstruction error* is minimal when using linear interpolation as a reconstruct procedure.

Given k points in an w-period, the sensitivity Δ_s of the AUC scoring function can be bounded, yielding a procedure to privately sample k points in the w-period using an efficient, suboptimal, version of *Equation* (4).

Theorem 3.2. *For an arbitrary w-period and k fixed points in the period, the sensitivity Δ_s of the AUC score is bounded by $2(w - k)$.*

Proof. Consider two data stream prefixes $D[t]$ and $D'[t]$ such that $D[t] \sim_w D'[t]$ and let us focus on the resulting stream counts x and x' in a w-period $[1, w]$. Let S be the set of k sampling points in the w-period and $\text{AUC}(x, S)$ denote the area under the curve between the points x and the polynomial ξ interpolating x at each point in S.

The sensitivity of the AUC scoring function is defined as

$$\max_{x \sim_w x'} \Delta_s = \left| \text{AUC}(x', S) - \text{AUC}(x, S) \right|.$$

Without loss of generality and for notation simplicity we focus on the case when $\text{AUC}(x, S) = 0$ (i.e., when $\xi(t) = x_t$

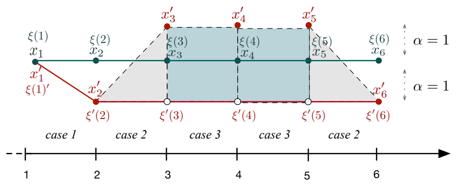


Figure 1: Illustration example of the AUC score sensitivity cases (proof of Theorem 3.2). In the example, $k = 3$, $S = \{1, 5, 6\}$. The areas of the polygons associated to cases 2 and 3 are shaded in gray and blue, respectively.

for all $t \in [w]$). Thus, Δ_s is maximized when AUC(x',S) is the largest. Notice that for any $t \in [w]$ $|\xi'(t) - x'_t| \leq 2$, since $\xi(t) = x_t$ (by assumption, above) and both $|\xi'(t) - \xi(t)| \leq 1$ and $|x'_t - x_t| \leq 1$ due to that $D_t \sim_\alpha D'_t$ for $\alpha = 1$, as stated in Section 2.3. The AUC(x', S) can be bounded by summing the areas of the polygons P_t with vertexes $x'_t, x'_{t+1}, \xi'(t), \xi'(t+1)$ for each $t = 1, \ldots, w-1$. There are three possible cases to consider (see Figure 1):

1. Both $t \in S$ and $t+1 \in S$. Thus, $\xi'(t) = x'_t$ and $\xi'(t+1) = x'_{t+1}$ as both x'_t and x'_{t+1} are interpolated exactly by ξ, and hence $P_t = 0$.
2. $t \in S$ and $t+1 \notin S$ (or $t \notin S$ and $t+1 \in S$). Let us consider the first case: If follows that $\xi'(t) = x'_t$, while $|\xi'(t+1) - x'_{t+1}| \leq 2$. Therefore, P_t can be inscribed into a triangle with points $x'_t, \xi'(t+1)$, and x'_{t+1}, whose area is 1. The second case is symmetric to the above.
3. Both $t \notin S$ and $t+1 \notin S$. Thus, $|\xi'(t) - x'_t| \leq 2$ and $|\xi'(t+1) - x'_{t+1}| \leq 2$. Therefore, P_t can be inscribed into a square with points $\xi'(t), x'_t, \xi'(t+1)$, and x'_{t+1}, whose area is 2.

The AUC(x', S) is maximized when there are $w - k - 1$ adjacent indexes where condition 2 holds, and 2 adjacent indexes where where condition 1 is true. Thus

$$\begin{aligned}\Delta_s &= \max_{x \sim_w x'} |\mathrm{AUC}(x, S) - \mathrm{AUC}(x', S)| \\ &= \mathrm{AUC}(x', S) = 2(w - k - 1) + 2 \\ &= 2(w - k). \quad \square\end{aligned}$$

To privately choose a good combination of k points to sample, one can rank each combination through the application of the Exponential Mechanism [21] in conjunction with the AUC scoring function. However, the computational complexity of such operation is in $\Theta(k^n)$, and thus such approach is infeasible for any realistic data stream and combinations of k and w-periods. Therefore, this paper uses a polynomial-time version of this method which relies on a combination of the *Sparse Vector Technique* (SVT) [12] and an incremental computation of the AUC scoring function.

The sampling procedure is depicted in Algorithm 2 and can be viewed as an instantiation of the SVT [12]. It takes as input the data stream x_ℓ processed within the current w-period, the number of points k to sample for measurement, along with the privacy budget ϵ_s and a user defined threshold $\Theta \in \mathbb{R}_+$ which influences the acceptable AUC score for choosing the next point to sample. Line 9 initializes the set of sample points S_ℓ with the first element of the w-period (this choice is necessary for executing the interpolation in the next steps), and it tracks the last previous point t_p selected for sampling. The algorithm first generates the noise

Algorithm 2: SAMPLE - *Adaptive AUC-based sampler*

```
   input  : The data stream $x_\ell$ seen in the current
            $w$-period $[t_a, t_b)$, the number of samples $k$, the
            privacy budget $\epsilon_s$, a threshold $\Theta$.
   output: A sample set $S_\ell$ of $k$ points in $[t_a, t_b)$
 9 $S_\ell = \{t_a\}$;  $t_p = t_a$
10 $\rho = \mathrm{Lap}(2\Delta_s/\epsilon_s)$
11 for $t = t_a + 1 \ldots, t_b - 1$ do
12     $\mu_t = \mathrm{Lap}(4k\Delta_s/\epsilon_s)$
13     if $AUC(\xi^{[t_p,t]}, x[t_p, t]) + \mu_t \geq \Theta + \rho$ then
14         $S_\ell = S_\ell \cup \{t\}$
15         $t_p = t$
16     if $|S_\ell| = k$ then
17         break
18 return $P$
```

ρ (line 10) which is added to the threshold Θ during comparison between each query and the threshold value (line 13). For each time stamp t, excluding t_a, the algorithm generates a noisy value μ_t to be added to the AUC score query which scales with $4k\Delta_s/\epsilon$ (line 12), where the AUC score $AUC(\xi^{[t_p,t]}, x[t_p, t])$ is defined as $\int_{t_p}^{p} |\xi^{[t_p,t]}(t) - x_t| dt$, and Δ_s is its the sensitivity(see Theorem 3.2). Notice that ϵ_s is equally divided for the noise generation of variables ρ and μ_t. Line 13 compares the perturbed query answer with the noisy threshold. If the result is above the threshold, then the point t is added to the set S_ℓ (line 14) and the previous selected point t_p is updated (line 15). Next, the algorithm keeps track of the number of index points stored and stops if the number matches k (lines 16–17). For a given threshold value, Algorithm 2 may generate a number of points in S_ℓ that is lower than k. To remedy to such limitation, when such a case occurs, we halve the threshold Θ and repeat the process until k points are selected.

THEOREM 3.3. *Algorithm 2 is ϵ_s-differential private.*

PROOF. The algorithm is an instantiation of the SVT mechanism [12] on disjoint sub-streams applied to the AUC scoring function. Both the threshold value and each output of the scoring function are perturbed to ensure privacy. The privacy budget ϵ_s is split equally to compute the noise ρ for the threshold and the noise μ_t for the scoring value. Thus, the noise applied to the threshold is given by $Lap(2\Delta_s)/\epsilon_s) = Lap(\Delta_s)/(0.5\epsilon_s))$ (line 10) and the noise to compute k releases of the AUC score is given by $Lap(4k\Delta_s/\epsilon_s) = Lap(2k\Delta_s/(0.5\epsilon_s))$ (line 12) where Δ_s is the sensitivity of the AUC score (Theorem 3.2). The original SVT proof from [12] shows that such noise ensures ϵ_s-differential privacy and, by parallel composition on the w-periods, Algorithm 2 it ensures ϵ_s-differential privacy on the entire data stream. $\square$

3.2 The Perturbation Procedure

Given the set S_ℓ of k sampling indexes for an w-period, the perturbation process takes as input the k-ary vector of the data stream measurements $x_\ell[S_\ell]$ at the sampling points and outputs a noisy version $\tilde{x}_\ell[S_\ell]$ of such vector satisfying ϵ_p-differential privacy. The process is performed by applying the Laplace mechanism with parameter k/ϵ_p.

THEOREM 3.4. Perturb *satisfy ϵ_p-differential privacy.*

Algorithm 3: *Optimization-based Post-processing*

```
input  : The private data stream $\tilde{x}_\ell$ seen in the current
         $w$-period $[t_a, t_b)$, the set of feature queries
         $Q_{\mathbf{F}}(\mathbf{x}_\ell)$, the privacy budget $\epsilon_o$
output: A private $w$-period data stream $\hat{x}_\ell$
19 $\mathbf{c} = \mathcal{M}_{\text{Lap}}(x_\ell; Q_{\mathbf{F}_i}, \epsilon/(p-1)$
20 $\mathbf{x}^* = \mathbf{argmin}_{\dot{\mathbf{x}}} \|\dot{\mathbf{x}} - \tilde{\mathbf{c}}\|^2_{2,w} =$
       $\sum_{i=1}^{p} \frac{1}{m_i} \sum_{j=1}^{m_i} (\dot{x}_{ij} - \tilde{c}_{ij})^2$        (O1)
   subject to :
       $\forall i', i : \mathbf{F}_{i'} \prec \mathbf{F}_i, \quad j \in [m_i] : \dot{x}_{ij} = \sum_{l : \mathbf{d}_{i'l} \subseteq \mathbf{d}_{ij}} \dot{x}_{i'l}$        (O2)
       $\forall i, j : \dot{x}_{ij} \geq 0.$        (O3)
   return $\hat{x}_\ell = x_{11}, \ldots, x_{1w}$
```

The above result follows by straightforward application of the Laplace mechanism (Theorem 2.4) on a set of k points and parallel composition (Theorem 2.2).

3.3 The Reconstruction Procedure

The reconstruct procedure takes as input the noisy measurements $\tilde{x}_{S_\ell} \in \mathbb{R}^k$ at the sample points S_ℓ in the w-period $[t_a, t_b)$ and outputs a vector $\tilde{x}_\ell \in \mathbb{R}^n$ of private estimates for the sub-stream x_ℓ. Each value $\tilde{x}_t$ of $\tilde{x}_\ell$ is obtained from the polynomial ξ at t, where ξ is the linear interpolation of the points in $\tilde{x}_{[S_\ell]}$. Section 4 analyzes how well the polynomial approximates the data stream x_t at any point $t \in [t_a, t_b)$. Notice that the reconstruction procedure is not required to query the real data stream and uses exclusively private information to compute its output. Hence, the output $\tilde{x}_\ell$ remains $\epsilon_s + \epsilon_p$-differential private by post-processing immunity of differential privacy.

3.4 The Optimization-based Post-Processing

The post-processing step computes the final estimates $\hat{x}_\ell$ of the sub-stream x_ℓ using the private sub-stream $\tilde{x}_\ell$ and additional queries over the data-stream w -period x_ℓ. The procedure is summarized in Algorithm 3. It uses the concept of *features* to capture semantic properties of the application of interest and queries these features in addition to noisy input $\tilde{x}_\ell$ of the original data stream. For example, two important features in the power load profile data stream are the period of times where peaks occur, as well as the total amount of load demand in an w-period.

A *feature* is a partition of the w-period and the size of the feature is the number of elements in the partition. A feature $\mathbf{F}'$ is a *sub-feature* of $\mathbf{F}$, denoted by $\mathbf{F}' \prec \mathbf{F}$, if $\mathbf{F}'$ is obtained by sub-partitioning $\mathbf{F}$. The *feature query* $Q_{\mathbf{F}}(\mathbf{x}_\ell)$ on data stream period x_ℓ associated with feature $\mathbf{F} = \{\mathbf{d}_1, \ldots, \mathbf{d}_m\}$ returns an m-dimensional vector $(c_1, \ldots, c_m)$ where each c_i is the sum of the values x_j for $j \in \mathbf{d}_i$.

The optimization-based post-processing takes as input the noisy data stream $\hat{x}_\ell$ from the reconstruction procedure and a collection of features queries $Q_{\mathbf{F}}(\mathbf{x}_\ell)$ for each $\mathbf{F}$ in the set of features $\mathscr{F} = \{\mathbf{F}_1, \ldots, \mathbf{F}_p\}$. For notational simplicity, we assume that the first feature always partitions the data stream w-period into singletons, i.e., $\mathbf{F}_1 = \{\{i\} : i \in [ta, tb)\}$. The noisy answer to this query is the output $\tilde{x}_\ell$ of the perturbation procedure (line 5 of Algorithm 1). When viewed as queries, the inputs to the mechanism can be represented as a set of values $Q_{\mathbf{F}_i}(x_\ell) = \mathbf{c}_i = (c_{i1}, \ldots, c_{im_i})$ $(1 \leq i \leq p)$ or, more concisely, as $\mathbf{c} = (c_{11}, \ldots, c_{pm_p})$. Finally, we assume that the partial ordering $\prec$ of features is given.

This first step of Algorithm 3 (line 19) applies the Laplace mechanism with privacy parameter $\frac{\epsilon}{p-1}$ to each feature query (excluding the first one whose answers, $\tilde{x}_\ell$ are already private), i.e.,

$$\mathcal{M}_{\text{Lap}}(x_\ell; Q_{\mathbf{F}_i}, \epsilon/p-1) = \tilde{c}_i = (\tilde{c}_{i1}, \ldots, \tilde{c}_{im_i}) \quad (2 \leq i \leq p).$$

The resulting values $\tilde{\mathbf{c}} = (\tilde{c}_{11}, \ldots, \tilde{c}_{pm_p})$ are then post-processed by the optimization algorithm depicted in line (20) to obtain the values $\mathbf{x}^* = (x^*_{11}, \ldots, x^*_{pm_p})$. Finally, the mechanism outputs a data sub-stream $\hat{x}_\ell = (\dot{x}^*_{11}, \ldots, \dot{x}^*_{1w})$.

The essence of Algorithm 3 is the optimization model depicted in line (20). Its decision variables are the post-processed values $\dot{\mathbf{x}} = (\dot{x}_{11}, \ldots, \dot{x}_{pm_p})$, and $\mathbf{w} = (w_1, \ldots, w_p) \in (0, 1]^p$ is a vector of reals representing weights for the terms of the objective function. The objective minimizes the squared weighted L$_2$-Norm of $\dot{\mathbf{x}} - \tilde{\mathbf{c}}$, where the weight w_i of element $x_{ij} - \tilde{c}_{ij}$ is $\frac{1}{m_i}$. The optimization is subject to a set of *consistency constraints* among comparable features and non-negativity constraints on the variables. For each pair of features $(\mathbf{F}_{i'}, \mathbf{F}_i)$ with $\mathbf{F}_{i'} \prec \mathbf{F}_i$, constraint O2 selects an element $\mathbf{d}_{ij} \in \mathbf{F}_i$ and all its subsets $\mathbf{d}_{i'l} \in \mathbf{F}_{i'}$ and imposes the constraint

$$\dot{x}_{ij} = \sum_{l : \mathbf{d}_{i'l} \subseteq \mathbf{d}_{ij}} \dot{x}_{i'l}$$

which ensures that the post-processed value $\dot{x}_{ij}$ is consistent with the sum of the post-processed values of its partition in $\mathbf{F}_{i'}$. By definition of sub-features, there exists a set of elements in $\mathbf{F}_{i'}$ whose union is equal to $\mathbf{d}_{ij}$.

Intuitively, Algorithm 3 can be thought as redistributing the noise introduced by the Laplace mechanism to obtain a consistent data set. The post-processing step searches for a solution that satisfies all the feature constraints and is as close as possible to the output Laplace-based noisy values.

THEOREM 3.5. *The optimization-based post-process achieves ϵ_o-differential privacy.*

PROOF. Since each feature partitions the w-period over the data stream, each feature query is a counting query with sensitivity 1. Thus, each $\tilde{c}_{ij}$ $(i > 1)$ obtained from the Laplace mechanism is ϵ_o-differential-private by Theorem 2.4 and the values $\tilde{x}_\ell = \tilde{c}_{11}, \ldots, \tilde{c}_{1w}$ are differential private (Theorems 3.3 and 3.4). Additionally, $(\tilde{c}_{11}, \ldots, \tilde{c}_{pm_p})$ is ϵ_o-differential-private by Theorem 2.1. Finally, result follows from post-processing immunity (Theorem 2.3). □

Observe that the mechanisms considered in this paper all operate over the universe of the data stream w-period. This is the case for instance of the Laplace mechanisms which runs in polynomial time in the size of the w-period. The next theoretical result characterizes the complexity of the optimization model depicted in line (20) and hence the complexity of Algorithm 3. Recall that a δ-solution to an optimization problem is a solution whose objective value is within distance δ of the optimum.

THEOREM 3.6. *A δ-solution to the optimization to the optimization model in line 20 (Algorithm 3) can be obtained in time polynomial in the size of the universe, the number of features, and $\frac{1}{\delta}$.*

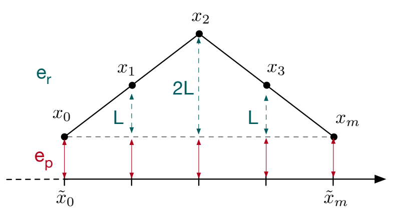


Figure 2: Illustration of the perturbation error e_p and the reconstruction error e_r introduced during the *Sample* and the *Reconstruct* procedure of Algorithm 1.

PROOF. First observe that the number of variables and constraints in the optimization model are bounded by a polynomial in size of the period w and the number of features p. Indeed, since the features are partitions, every set $\mathbf{d}_{i'l}$ in Constraint (O2) is a subset of exactly one $\mathbf{d}_{ij}$. The result then follows from the fact that the optimization model is convex, which implies that a δ-solution can be found in time polynomial in the size of the universe, the number of features, and $\frac{1}{\delta}$ [23]. □

4. ERROR ANALYSIS

This section analyzes how the sampling and reconstruction procedure can improve the accuracy of the output. It characterizes the error $\mathbb{E}\|\hat{x}_\ell - x_\ell\|_2^2$ of the results of Algorithm 1 over a w-period stream release, when the equally-spaced approach is selected as a sampling procedure and no post-processing is applied (i.e., when the algorithm releases $\tilde{x}_\ell$ (line 6)).

4.1 Sample and Reconstruct

Consider a w-period of the data stream x_ℓ and assume, for notational simplicity, that the w-period is $[0, w)$. Let $k = |S_\ell|$ be the number of samples selected for *measurements*, by the equally-spaced sampling in addition to the initial point. This determines the length of each *segment* $m = w/k$ during which the *Reconstruct* procedure interpolates values without extra measurements. We assume m to be an even number. Let L be the Lipschitz constant of the function x describing the data stream, and defined as $L := \sup_{t\in[w]} x_t - x_{t-1}$.

THEOREM 4.1. *The error introduced by Algorithm 1 (ignoring post-processing) with the equally-spaced sampling procedure with parameter k is bounded by $O\left(m^2L^2w + 2\frac{w^2L}{\epsilon} + 2\frac{w^3}{m^2\epsilon^2}\right)$.*

PROOF. For notational simplicity, consider the first period, where the sample points 0 and m are selected and x_0, x_m are measured privately. The reconstruct procedure uses linear interpolation between x_0 and x_m to recover the values of the non-sampling points $x_1, \ldots, x_{m-1}$.

There are two sources of error: the *perturbation error* e_p and the *reconstruction error* e_r. The worst case reconstruction error is given by $e_r = L\max(t, m-t)$ (see Figure 2). The perturbation error e_p is given by the additive Laplace noise used for privacy. There are k measurements taken, and thus to ensure privacy, we must divide the privacy budget ϵ into k, which results in: $e_r = |\tilde{x}_0 - x_0| = \text{Lap}(k/\epsilon)$. This error adds to the perturbation error at every point in the w-period. Therefore, for all $t \in [w]$:

$$\|\tilde{x}_t - x_t\|_2^2 \leq (e_p + e_r)^2$$

$$(\max(t, m-t)L + \text{Lap}(k/\epsilon))^2$$

and in expectation,

$$\mathbb{E}\|\tilde{x}_t - x_t\|_2^2 \leq \mathbb{E}_{Z\sim\text{Lap}(k/\epsilon)}\left[2\sum_{i=1}^{m/2}(iL+Z))^2\right]$$

$$= 2\sum_{i=1}^{m/2} i^2L^2 + 4\sum_{i=1}^{m/2} iL\mathbb{E}[Z] + m\mathbb{E}[Z^2]$$

$$= \frac{1}{3}\frac{m}{2}(\frac{m}{2}+1)(m+1)L^2 + 2\frac{m}{2}(\frac{m}{2}+1)L\frac{k}{\epsilon} + 2m(\frac{k}{\epsilon})^2.$$

As a result, the error $\|\tilde{x}_t - x_t\|_2^2$ is bounded by $O(m^3L^2 + 2m^2L\frac{k}{\epsilon} + 2m(\frac{k}{\epsilon})^2)$. Multiplying the above by $k = \frac{w}{m}$ gives the final error which is bounded by $O\left(m^2L^2w + 2\frac{w^2L}{\epsilon} + 2\frac{w^3}{m^2\epsilon^2}\right)$. □

For $m = \sqrt{\frac{w}{\epsilon L}}$ the above expression produces an error of $O(w^2L/\epsilon)$. In comparison, applying the Laplace mechanism to produce a private sub-stream in the w-period produces an error of $w\,\mathbb{E}_{\text{Lap}(w/\epsilon)}[Z^2] = \frac{w^3}{\epsilon^2}$.

The result above shows that choosing a sampling parameter k, to sample uniformly every m time steps, may allow Algorithm 1 to produce outputs with a substantially lower error than those obtained by the Laplace mechanism. Although this result applies to the equally-spaced sampling procedure, Section 6 demonstrates experimentally that the AUC-based sampling procedure outperforms its equally-spaced counterpart.

4.2 Optimization-based Post-Processing

The following result is from [17]. It bounds the error of the optimization-based post-processing. It is most likely not tight but proves that the post-processing step can accommodate any side-constraints without degrading the accuracy of the mechanism significantly.

THEOREM 4.2. *The optimal solution to the optimization model in line 20 of Algorithm 1 satisfies*

$$\|\mathbf{x}^* - \mathbf{c}\|_{2,w} \leq 2\|\tilde{\mathbf{c}} - \mathbf{c}\|_{2,w}.$$

5. HIERARCHICAL DATA STREAMS

This section describes an extension to OPTSTREAM to support *hierarchical data streams*. A data stream D is called hierarchical if, for any time t, there is an *aggregation entry* $a = (i_a, r_a, t)$ associated with an *aggregation* set $S_a \subseteq \mathcal{I}$ that reports the sum of values reported by all entries in S_a at time t, i.e., $r_a = \sum_{u\in S_a} r_u^t$. More formally, a data stream is hierarchical if, for any two aggregation entries a_1, a_2 $S_{a_1} \subset S_{a_2}$ or $S_{a_2} \subset S_{a_1}$, or $S_{a_1} \cap S_{a_2} = \emptyset$.

A set of aggregation entries $\mathcal{S}$ can be represented hierarchically through a tree (called *aggregation tree*), in which the root is defined by the entry that is not contained in any other entries and the children of an aggregation entry p are all the entries in D whose identifiers are contained in S_p. The *height* of a hierarchical data is the maximum path length from the root to any leaf of its tree.

EXAMPLE 5.1. *In our target application, a data curator is interested in the aggregated load consumption stream at the level of a small geographical region, at a group of regions within the same electrical sub-station, and finally at* *the nation-level. These aggregations form a hierarchy which root is represented by the nation-level aggregation entry, its children by the electrical sub-station entries, and the tree leaves by the region-level entries.*

To answer each query of the hierarchical stream, OptStream can be extended as follows. For each w-period, the algorithm runs the sampling, perturbation, and reconstruction procedure for each level of the aggregation tree representing the hierarchical stream. Finally, the post-processing optimization takes as input the answers to all the aggregation entries queries with the set of features being equal to the set of aggregation entries. The post-processing step thus enforces consistency between the aggregation counts at a node and the sum of counts at its children nodes in the tree, in addition to the features described earlier in the paper.

Theorem 5.1. *For a hierarchy of height h,* OptStream *when using a privacy budget of ϵ/h for each level of the hierarchy satisfies ϵ-differential privacy.*

The result follows from the privacy guarantee of OptStream (Theorem 3.1), parallel composition of differential privacy across each level of the hierarchy (Theorem 2.2), and sequential composition of differential privacy, for each of the h hierarchical queries (Theorem 2.1).

6. EVALUATION

This section evaluates OptStream on real data streams for a number of tasks. It first evaluates the accuracy of the private release of a data stream. It then studies the accuracy of privately releasing hierarchical streams on aggregated queries. Finally, it analyzes the accuracy of forecasting tasks from the released private data streams.

Dataset. The source data was obtained through a collaboration with *Réseau de Transport d'Électricité*, the largest energy transmission system operator in Europe. It consist of a one-year national-level load energy consumption data with a granularity of 30 minutes. The data is aggregated at a regional level and $\mathscr{R}$ denotes the set of regions. Each data point in the stream represents the total load consumption of the customers served within a region during a 30 minute time period. Thus, for every region $R \in \mathscr{R}$, a stream of data $x(R) = x_1(R), x_2(R), \ldots$ is generated where $x_t(R)$ represents the energy demand to supply in order to serve the region R at time t. When the streams of all regions are aggregated, the resulting load consumption data constitutes a data stream of the energy profile at a national level.

For evaluation purposes, the experiments often consider a representative region (Auvergne - Rhône-Alpes) to evaluate the data stream release. For the evaluation of hierarchical data streams, the experiments consider a hierarchical aggregation tree of hight 2, where the root node is the national level and each of the leaves correspond to one region. Table 1 lists an overview of the data streams derived from real energy load consumption data for each France region during 2016. Each data stream contains 17,520 entires.

6.1 Private Stream Data-Release

Answering queries over contiguous w-periods corresponds to releasing the private stream over the entire available duration. Figure 3 illustrates the real and private versions

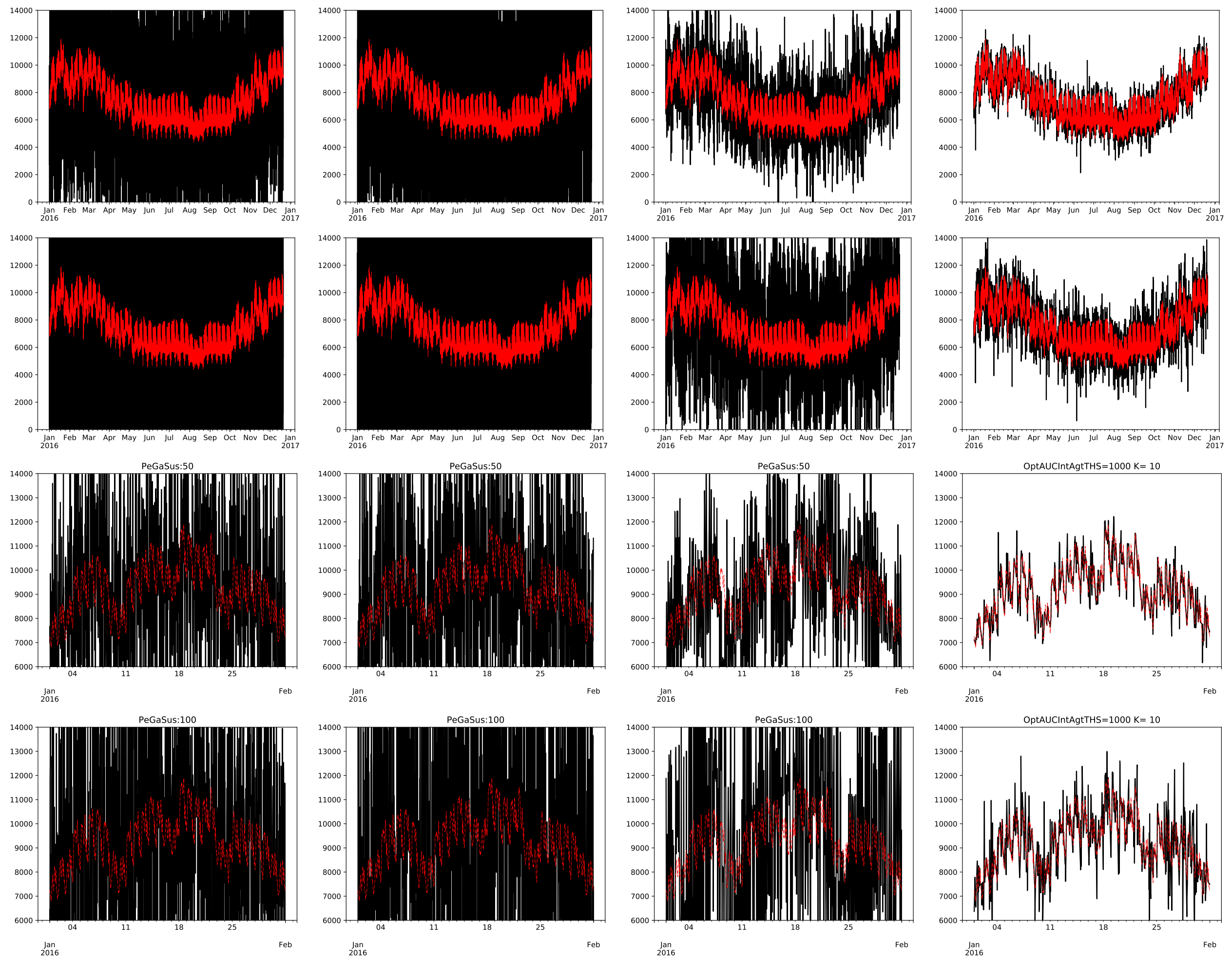
Stream Data Regions ($\mathscr{R}$)	Daily Average (MW)	Length
Auvergne-Rhône-Alpes	7717.58	17,520
Bretagne	2554.23	17,520
Bourgogne - Franche-Comtè	2498.23	17,520
Centre - Val de Loire	2157.97	17,520
Grand Est	5286.24	17,520
Hauts de France	5832.26	17,520
Ile de France (Paris)	8315.13	17,520
Nouvelle Aquitaine	4985.68	17,520
Normandie	3267.22	17,520
Occitanie	4314.7	17,520
Pays de la Loire	3174.17	17,520
Provence - Cote d'Azur	4782.4	17,520

Table 1: Overview of the power load consumption stream data derived from the France regions in 2016. *Daily Average* refers to the average power (in MW) demanded daily, and *Length* refers to the number of time steps available.

of the stream for w-periods of size 48 (i.e., one day) given privacy budget of $\epsilon = 1$ and indistinguishability parameter $\alpha = 50$ (first and third rows) and $\alpha = 100$ (second and fourth rows). Recall that the choice of the indistinguishability parameter α allows the data curator to ensure the protection of the observed increase/decrease power consumption up to α MegaWatts (MW), while the the w-period specify the length of the obfuscation period.

The first two rows of Figure 3 illustrate the private energy load stream released for the Auvergne-Rhône-Alpes region in 2016, while the bottom two rows details the results for the month of January 2016. The real loads are highlighted in red in each of the plots, while their private counterparts are shown in black. The figure compares our proposed OptStream algorithm against the *Laplace mechanism* (a), the *PeGaSus* algorithm [6], which uses a combination of the Laplace mechanism, a partitioning strategy to group values in contiguous time steps whose deviation is small, and a smoothing function (b), as well as the *Discrete Fourier Transform (DFT)* algorithm [26], which ensures differential privacy by perturbing the discrete Fourier coefficients of a time-series and adapted to release streams on w-periods (c).

For *PeGaSus*, the privacy budget is divided equally for each time step in the w-period. Additionally, the experiments use the values of the meta-parameters indicated in the original paper [6]. For *DFT* and OptStream, the experiments set the number of Fourier coefficients and sampling steps, to 10 (when $\alpha = 50$) and 5 (when $\alpha = 100$). The privacy budget allocated to perform each measurement is split equally. Additionally, for OptStream $\epsilon_s = \epsilon_p = \epsilon_o = \frac{1}{3}\epsilon$, and the AUC-based Sampling procedure uses a threshold value Θ of 1000 (which is about one tenth of the average load consumption). Different, larger Θ values introduce no substantial differences in the average L1 error. Finally, OptStream uses the following feature query set $\mathscr{F} = \{\mathbf{F}_1, \mathbf{F}_2, \mathbf{F}_3\}$ in the optimization-based post processing step, with $\mathbf{F}_1 \prec \mathbf{F}_2 \prec \mathbf{F}_3$. $\mathbf{F}_1$ is defined as described in Section 3.4; $\mathbf{F}_2$ partitions each w-period in 4 sets, the intervals $[0, 14)$, $[14, 24)$, $[24, 18)$, and $[18, 48)$ that correspond to aggregated consumption for the following times of the day: [0am-7am), [7am-12pm), [12pm-6pm), and [6pm-0am) respectively. $\mathbf{F}_3$ partitions each w-period in a single set, listing all the time steps within the w-period and thus describing the aggregated daily energy consumption. If an algorithm reports negative noisy value for a stream point,



(a) Laplace (b) PeGaSus (c) DFT (d) OPTSTREAM

Figure 3: Original load consumption data for the Auvergne-Rhône-Alpes region in 2016, shown in red, and its private versions obtained using Laplace (a) PeGaSus (b), DFT (c), and Opt (d) with privacy budget $\epsilon = 1$ and $\alpha = 50$ (top) and 100 (bottom). The last two rows show the same content as above but detail the results for the month of January 2016.

we truncate it to zero. The query set represents salient moments in the day associated with different consumption patterns. These are proxy of consumer behaviors and thus energy consumption. Because these queries return private answers, the privacy is guaranteed by the post-processing immunity of DP (see Theorem 2.3).

Figure 3 clearly illustrates that OPTSTREAM produces private streams that are more accurate than its competitors when visualized. The next paragraph quantifies the reduction in error produced by OPTSTREAM.

Average L_1-Error Analysis. We now report the average L_1 error for each w-period produced by the algorithms. For each w-period, each input data stream x_ℓ illustrated in Table 1, and each reported private stream $\hat{x}_\ell$, we compare the average L_1 error defined as: $\sum_{t \in [w]} |\hat{x}_\ell - x_\ell|$. Figure 4 reports the average results for each streaming region for the months of February (left), June (middle), and October (right), at varying of the indistinguishability parameter $\alpha = \{1, 50, 100, 200\}$ for privacy budgets $\epsilon = 1.0$ (top) and $\epsilon = 0.1$ (bottom). The consumption in these three months captures different customers load profile behaviors due to different weather and duration of the day light. Each histogram reports the $\log_{10}$ value of the average error of 30 random trials. Two version of OPTSTREAM (OPTSTREAMES and OPTSTREAMAS) are presented and correspond to the Equally-Spaced Sampling, and AUC-Based Sampling procedures, respectively. While all algorithms obviously induce an L_1-error which increases as the privacy budget decreases, the figure highlights that OPTSTREAM consistently outperforms the other algorithms.

Evaluation of the Sample and Post-Process strategy. We also evaluate the effect of the different sampling strategies as well as the benefits of the post-processing procedure of OPTSTREAM. Figure 5 illustrates the average L_1-error using the same settings as previously described when OPTSTREAM uses the equally-spaced sampling procedure without post-processing step (OptStreamES) and with post-processing step (OptStreamES+P) and when it exploits the adaptive AUC-based sampling without post-processing step (OptStreamAS) and with post-processing step (OptStrea-

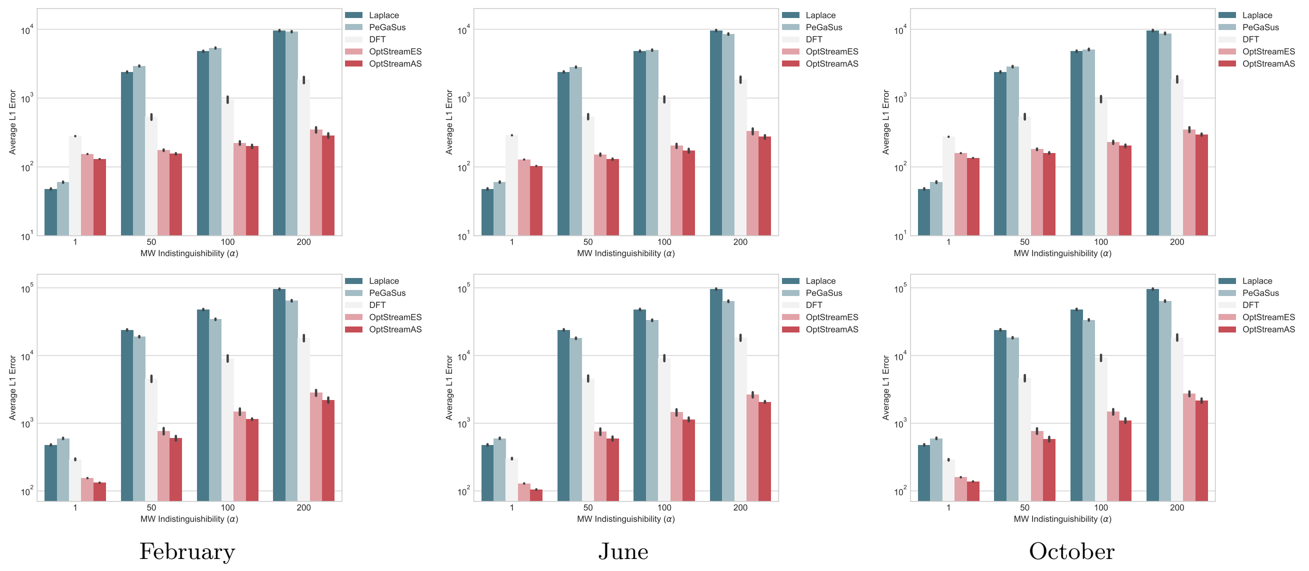


Figure 4: L_1 error analysis for energy load streams for the months of February (left), June (middle), and October (right) for $\epsilon = 1.0$ (top) and $\epsilon = 0.1$ (bottom). The y-axis reports $\log_{10}$ of the average L_1 error for each stream of region $R \in \mathscr{R}$.

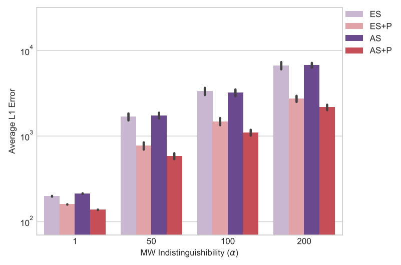


Figure 5: OptStream post-processing (October, $\epsilon = 0.1$).

mAS+P). Each histogram reports the $\log_{10}$ value of the average error of 30 random trials. Figure 5 clearly shows that (1) the optimization-based post-processing step plays a key role in reducing the L_1-error for all the ϵ setting and for both sampling strategies, and (2) the AUC-based sampling is able to generate streams with lower errors than those generated by the Equally-spaced Sampling.

6.2 Hierarchical Private Data-Stream Release

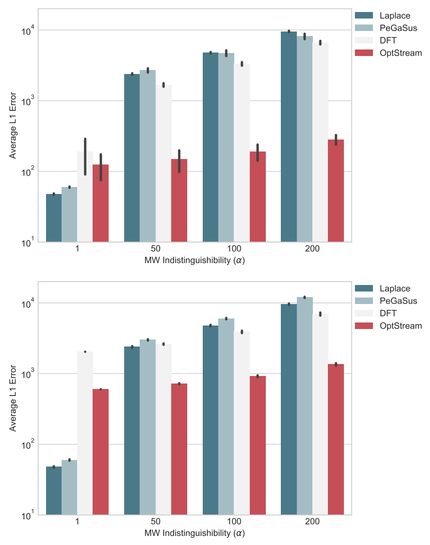
This section evaluates the extensions proposed in Section 5 for releasing aggregated queries over hierarchical data streams. We answer contiguous w-periods over the whole duration of the stream for the following queries: count queries over the data stream $\mathbf{x}(R) = x_1(R), x_2(R), \ldots$ for each region $R \in \mathscr{R}$ listed in Table 1, as well as count queries $\mathbf{x} = x_1, x_2, \ldots$, where each $x_t = \sum_{R \in \mathscr{R}} x_t(R)$ represents the aggregated load consumption at national level. Thus, we create a hierarchy of two levels and answer simultaneously all queries. We allocate a privacy budget of $\frac{\epsilon}{2}$ to each level of the hierarchy when using OptStream.

We compare against (1) the Laplace mechanism applied to each stream data, using a uniform allocation of the privacy budget ($\frac{\epsilon}{2}$) for each query in a different level of the stream hierarchy; (2) a *hierarchical version of PeGaSus* that was introduced in [6] to optimize PeGaSus to answer hierarchical queries; and (3) the DFT algorithm which answers queries over each data streams by uniformly allocating a portion of the privacy budget at each level of the hierarchy.

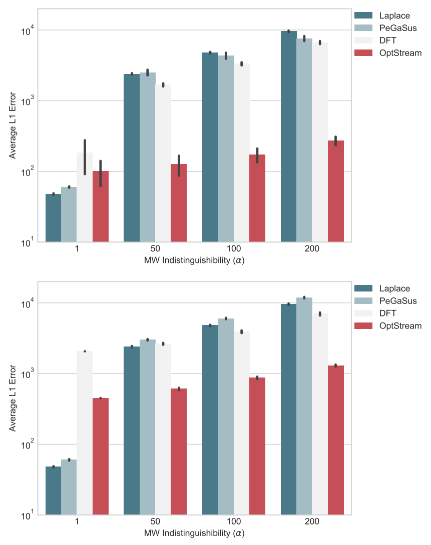
Figure 6 shows the results for three different months of the year: February (left), June (middle), and October (right), and under different indistinguishability parameters α for privacy budget $\epsilon = 1.0$. The top row of the figure reports the average-L_1 errors when releasing stream data $x(R)$ associated with each region R, while the bottom row gives the L_1-errors when releasing the stream data x at national level. Each histogram reports the $\log_{10}$ value of the average error of 30 random trials. The results illustrates similar trends to those in previous experiments. Overall, OptStream with the adaptive AUC-based Sampling produces stream data with the lowest average L_1-errors for both levels of the stream hierarchy.

6.3 Impact of Privacy on Forecasting Demand

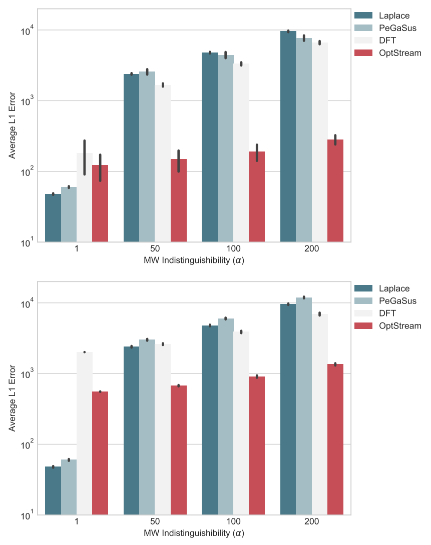
Finally, we evaluate the capability of released private streams to predict future time steps. To do so, we adopt the Autoregressive Moving Average (ARMA) model [1, 29, 7, 18]. ARMA is a popular stochastic time series model used for predicting future points in a time series (forecast). It combines an Autoregressive (AR) model [1] and a Moving average model [1, 7], i.e., ARMA(p,q) combines AR(p) and MA(q) and is suitable for univariate time series modeling. In an AR(p) model, the future value of a variable x_t is assumed to be a linear combination of p past observations and a random error: $x_t = c + \sum_{i=1}^{p} \phi_i x_{t-i} + \beta_t$, where c is constant, β_t is a random variable modeling white noise at time t, and the $\phi_i (i = 1, \ldots, p)$ are model parameters. An MA(q) model uses the past q errors in the time series as the explanatory variables. It estimates a variable x_t using $\mu + \beta_t + \sum_{i=1}^{q} \theta_i \beta_{t-i}$, where the $\theta_i (i = 1, \ldots, q)$ are model parameters, μ is the expectation of X_t, and the β_t terms are white noise error terms. The ARMA model with parameters p and q refers to the model with p autoregressive terms and q moving-average terms: It estimates a future time step



February

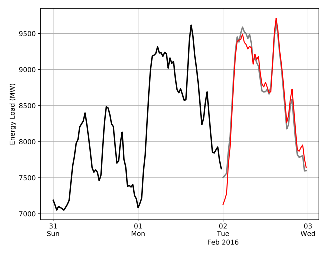


June

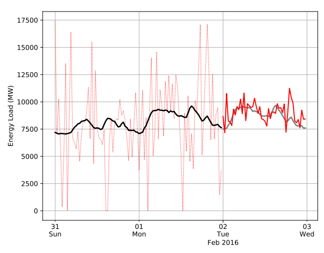


October

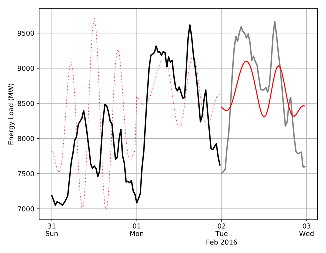
Figure 6: L_1 errors analysis on hierarchical energy load stream data for the months of February (left), June (middle), and October (right), for each stream of region $R \in \mathscr{R}$ (top) and at national level (bottom).



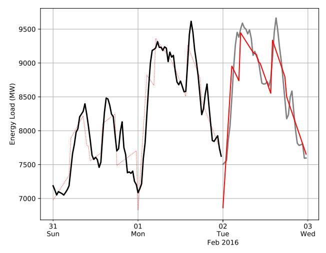
(a) $\alpha = 50$, Real



(b) $\alpha = 50$, PeGaSus



(c) $\alpha = 50$, DFT



(d) $\alpha = 50$, OPTSTREAM

Figure 7: Forecast for a one day load consumption through an ARMA model on the original load consumption data (a) and its private versions obtained using PeGaSus (b), DFT (c), and OPTSTREAM (d) with privacy budget $\epsilon = 1.0$, and indistinguishibility parameter $\alpha = 50$.

value x_t as $c + \beta_t + \sum_{i=1}^{p} \phi_i x_{t-i} + \sum_{i=1}^{q} \theta_i \beta_{t-i}$. In our experiments, we use an ARMA model with parameters $p = q = 1$ to estimate the future 48 time steps (corresponding to a day) when trained with the past four weeks of the private data stream estimated using DFT, PeGaSus, and OPTSTREAM with AUC-Based sampling. All models use the same parameters adopted in the previous sections.

Figure 7 visualizes the forecast for the load consumptions in the Auvergne-Rhône-Alpes region for February 2, 2016. The black and gray solid lines respectively illustrates the real load values observed so far and those of the day to be forecasted. The dotted red lines illustrates the private stream data estimated so far (and used as input to the prediction model), and the solid red lines depict the prediction obtained using the ARMA model. Figure 7(a–d) show the forecast results using the real data, and the private stream obtained through DFT, PeGaSus, and OPTSTREAM, respectively. The figure clearly shows that OPTSTREAM is able to visually produce better estimates for the next day forecast.

Average L_1-error Analysis. We also quantitatively evaluate the average L_1 error for each prediction produced by the mechanisms. We adopt the same setting as above for the prediction and report, in Figure 4, the average L_1 error for predicting each day in the month of February (a), June (b), and October (c) for the Auvergne-Rhône-Alpes region. Each histogram reports the $\log_{10}$ value of the average error of 30 random trials. We observe that OPTSTREAM reports the smallest errors compared to all other privacy-preserving algorithms, and that the error made by OPTSTREAM in reporting the next day forecast is closer to the error made in the forecast prediction using the real data than when using another method.

7. RELATED WORK

Continuous release of aggregated real-time data has been studied in previous work including [9, 11]. Most of the state-of-the-art either focuses on event-level privacy on infinite streams [27] or on user-level privacy on finite streams [11]. Dwork proposed an adaptation of differential privacy to a continuous observation setting [11]. Her work focused on releasing bit-streams and proposed an algorithm for counting the number of 1s in the stream under event-level differential privacy. Mir et al. [22] proposed *pan-privacy* for estimating

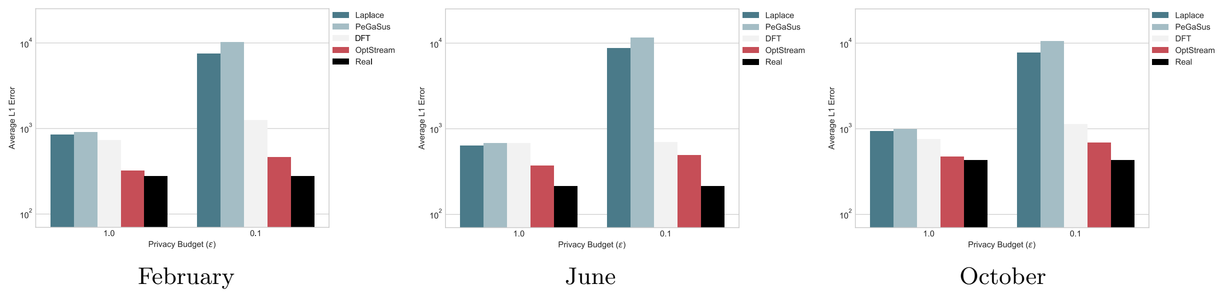


Figure 8: Average L_1 error for the ARMA forecasting model on stream data for the energy loads of the months of February (left), June (middle), and October (right) for the Auvergne-Rhône-Alpes region.

counts, moments, and heavy-hitters on data streams while preserving differential privacy even if the internal memory of the algorithms is observed by an attacker. Kellaris et al. [19] proposed the notion of w-event privacy to balance event-level and user-level privacy, trading off utility and privacy to protect event sequences within a time window of w time steps. OptStream uses the last model.

Within the differential privacy proposal for data streams that fit such proposals, [28] proposed *Rescue DP*, which is designed explicitly for spatiotemporal traces. PeGaSus [6] is another seminal work that allows for the protection of both event privacy as well as w-event privacy using a combination of perturbation, partitioning, and smoothing. The algorithm uses a combination of Laplace noise to perturb the data stream for privacy, a grouping strategy which incrementally partitions the space of observed data points by grouping points with small deviations, and a smoothing schema which is used to post-process the private data stream. Both algorithms rely on the idea of contiguous grouping time steps and average the perturbed data within every region in a group. We compared OptStream against PeGaSus, which is the state-of-the-art on private data-stream release and has been showed to be effective in several settings.

Rastogi and Nath [26] proposed an algorithm which perturbs a small number of Discrete Fourier Transform (DFT) coefficients of the entire time series and reconstructs a released version of the Inverse DFT series. While, in the original paper, the algorithm requires the entire time-series, such approach has been used for w-event privacy in the context of data streams [13]. We also compare OptStream against DFT.

Fan et al. proposed Fast [14], an adaptive algorithm for private release of aggregate statistics which uses a combination of sampling and filtering. Their algorithm is based on user-level differential privacy, which is not comparable to the chosen model of w-event level differential privacy. Additionally, in their work, the authors compare the proposed approach against DFT [26] and, while showing improvements, the error remains within the same error magnitude of those produced by DFT. In contrast, our experimental analysis (Section 6) clearly illustrates that OptStream reduces the error of several orders of magnitude when compared to DFT.

Chan et al. [4] focused on releasing prefix-sums of the streaming data counts while adopting an event-based privacy model. Bolot et al. [3] also used an event-based model and proposed an algorithm for answering sliding window queries on data streams. The model adopted in their work is however incompatible with the privacy model adopted in this work, and hence we did not compare against such approaches.

8. CONCLUSIONS

This paper presented OptStream, a novel algorithm for privately releasing stream data in the w-event privacy model of Kellaris et al. OptStream is a 4-step procedure consisting of sampling, perturbation, reconstruction, and post-processing modules. The *sampling* module selects a small set of data points to measure privately in each period of interest. The *perturbation* module injects noise to the sampled data points to ensure privacy. The *reconstruction* module reconstructs the data points excluded from measurement from the perturbed sampled points. Finally, the *post-processing* module uses convex optimization over the private output of the previous modules, along with the private answers of additional queries on the data stream, to ensure coherence of quantities associated to salient features of the data. OptStream was evaluated on a real data set from the largest transmission operator in Europe. Experimental results on multiple test cases show that OptStream improves the accuracy of state-of-the-art by at least one order of magnitude on this application domain. The accuracy improvements are measured, not only in terms of the error distance to the original stream but also in the accuracy of a popular load forecasting algorithm trained on the private data substreams. The results additionally show that OptStream exhibits similar benefits on hierarchical stream data which is also highly desirable in practice. Finally, the experimental analysis has demonstrated that both the adaptive sampling and post-processing optimization are critical in obtaining strong accuracy. Future work will be devoted to generalizing these results to the streaming setting where a data element is emitted at each time step and ensuring that the released data elements can produce feasible solutions for the optimization problem of interest, as explored in [16].

Acknowledgments

This research is partly funded by the Michigan Institute of Data Science. The views and conclusions contained in this document are those of the authors only.

REFERENCES

[1] L. C. Alwan and H. V. Roberts. Time-series modeling for statistical process control. *Journal of Business & Economic Statistics*, 6(1):87–95, 1988.

[2] M. E. Andrés, N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi. Geo-indistinguishability: Differential privacy for location-based systems. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 901–914. ACM, 2013.

[3] J. Bolot, N. Fawaz, S. Muthukrishnan, A. Nikolov, and N. Taft. Private decayed predicate sums on streams. In *Proceedings of the 16th International Conference on Database Theory*, pages 284–295. ACM, 2013.

[4] T.-H. H. Chan, E. Shi, and D. Song. Private and continual release of statistics. *ACM Transactions on Information and System Security (TISSEC)*, 14(3):26, 2011.

[5] K. Chatzikokolakis, M. E. Andrés, N. E. Bordenabe, and C. Palamidessi. Broadening the scope of differential privacy using metrics. In *International Symposium on Privacy Enhancing Technologies Symposium*, pages 82–102. Springer, 2013.

[6] Y. Chen, A. Machanavajjhala, M. Hay, and G. Miklau. Pegasus: Data-adaptive differentially private stream processing. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1375–1388. ACM, 2017.

[7] J. H. Cochrane. Time series for macroeconomics and finance. *Manuscript, University of Chicago*, 2005.

[8] B. Ding, J. Kulkarni, and S. Yekhanin. Collecting telemetry data privately. In *Advances in Neural Information Processing Systems*, pages 3574–3583, 2017.

[9] C. Dwork. Differential privacy in new settings. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 174–183. SIAM, 2010.

[10] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, volume 3876, pages 265–284. Springer, 2006.

[11] C. Dwork, M. Naor, T. Pitassi, and G. N. Rothblum. Differential privacy under continual observation. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 715–724. ACM, 2010.

[12] C. Dwork and A. Roth. The algorithmic foundations of differential privacy. *Theoretical Computer Science*, 9(3-4):211–407, 2013.

[13] L. Fan and L. Xiong. An adaptive approach to real-time aggregate monitoring with differential privacy. *IEEE Transactions on Knowledge and Data Engineering*, 26(9):2094–2106, Sept 2014.

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

[15] G. Fanti, V. Pihur, and Ú. Erlingsson. Building a rappor with the unknown: Privacy-preserving learning of associations and data dictionaries. *Proceedings on Privacy Enhancing Technologies*, 2016(3):41–61, 2016.

[16] F. Fioretto and P. V. Hentenryck. Constrained-based differential privacy: Releasing optimal power flow benchmarks privately. In *Proceedings of the International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research (CPAIOR)*, pages 215–231, 2018.

[17] F. Fioretto, C. Lee, and P. V. Hentenryck. Constrained-based differential privacy for private mobility. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1405–1413, 2018.

[18] K. W. Hipel and A. I. McLeod. *Time series modelling of water resources and environmental systems*, volume 45. Elsevier, 1994.

[19] G. Kellaris, S. Papadopoulos, X. Xiao, and D. Papadias. Differentially private event sequences over infinite streams. *Proceedings of the VLDB Endowment*, 7(12):1155–1166, 2014.

[20] F. Koufogiannis, S. Han, and G. J. Pappas. Optimality of the laplace mechanism in differential privacy. *arXiv preprint arXiv:1504.00065*, 2015.

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

[22] D. Mir, S. Muthukrishnan, A. Nikolov, and R. N. Wright. Pan-private algorithms via statistics on sketches. In *Proceedings of the thirtieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 37–48. ACM, 2011.

[23] A. Nemirovski. Interior point polynomial time methods in convex programming. Technical Report ISYE 8813, Georgia Institute of Technology, Atlanta, GA, Spring 2004.

[24] F. J. Nogales, J. Contreras, A. J. Conejo, and R. Espínola. Forecasting next-day electricity prices by time series models. *IEEE Transactions on power systems*, 17(2):342–348, 2002.

[25] L. F. Ochoa and G. P. Harrison. Minimizing energy losses: Optimal accommodation and smart operation of renewable distributed generation. *IEEE Transactions on Power Systems*, 26(1):198–205, 2011.

[26] V. Rastogi and S. 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. ACM, 2010.

[27] V. Rastogi and S. 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. ACM, 2010.

[28] Q. Wang, Y. Zhang, X. Lu, Z. Wang, Z. Qin, and K. Ren. Rescuedp: Real-time spatio-temporal crowd-sourced data publishing with differential privacy. In *INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications, IEEE*, pages 1–9. IEEE, 2016.

[29] G. P. Zhang. Time series forecasting using a hybrid arima and neural network model. *Neurocomputing*, 50:159–175, 2003.