Journal of Artificial Intelligence Research 65 (2019) 423–456 Submitted 10/2018; published 06/2019

OptStream: Releasing Time Series Privately

Ferdinando Fioretto FIORETTO@GATECH.EDU
School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, GA 30332, USA

Pascal Van Hentenryck PVH@ISYE.GATECH.EDU
School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, GA 30332, USA

Abstract

Many applications of machine learning and optimization operate on data streams. While these datasets are fundamental to fuel decision-making algorithms, often they contain sensitive information about individuals, and their usage poses significant privacy risks. Motivated by an application in energy systems, this paper presents OPTSTREAM, a novel algorithm for releasing differentially private data streams under the w-event model of privacy. OPTSTREAM is a 4-step procedure consisting of sampling, perturbation, reconstruction, and post-processing modules. First, the *sampling* module selects a small set of points to access in each period of interest. Then, the *perturbation* module adds noise to the sampled data points to guarantee privacy. Next, the *reconstruction* module reassembles non-sampled data points from the perturbed sample points. Finally, the *post-processing* module uses convex optimization over the privacy-preserving output of the previous modules, as well as the privacy-preserving answers of additional queries on the data stream, to improve accuracy by redistributing the added noise. OPTSTREAM is evaluated on a test case involving the release of a real data stream from the largest European transmission operator. Experimental results show that OPTSTREAM may not only improve the accuracy of state-of-the-art methods by at least one order of magnitude but also supports accurate load forecasting on the privacy-preserving data.

1. Introduction

Differential privacy (Dwork, McSherry, Nissim, & Smith, 2006) has emerged as a robust framework to release datasets while limiting the disclosure of participating individuals. Informally, it ensures that what can be learned about an individual in a differentially private dataset is, with high probability, limited to what could have been learned about the individual in the same dataset but without her data.

Many applications of machine learning and optimization, in areas such as healthcare, traffic management, and social networks, operate over streams of data. The use of differential privacy for the privacy-preserving release of time series has attracted increased attention in recent years (e.g., Dwork, Naor, Pitassi, & Rothblum, 2010; Dwork, 2010; Fanti, Pihur, & Erlingsson, 2016; Ding, Kulkarni, & Yekhanin, 2017; Chen, Machanavajjhala, Hay, & Miklau, 2017) where aggregated statistics are continuously reported. Two common approaches for time series data release are the *event-level* and *user-level* privacy models (Dwork et al., 2010). The former focuses on protecting a single *event*, while the latter aims at protecting *all* the events associated with a single user, i.e., it focuses on protecting a computation of an individual in the dataset. Additionally, Kellaris, Pa-

padopoulos, Xiao, and Papadias (2014) proposed the notion of w-event privacy to achieve 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 was motivated by a desire to release privacy-preserving streams of energy demands, also called *loads*, in transmission systems. The goal is that of 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 the economic activities of grid customers. For example, changes in load consumption may indirectly reveal production levels and strategic investments and other similar information. Moreover, these time series are often input to complex analytic tasks, e.g., demand forecasting algorithms (Nogales, Contreras, Conejo, & Espínola, 2002) and optimal power flows (Ochoa & Harrison, 2011). As a result, the accuracy of the privacy-preserving datasets is critical and, as shown later in the paper, existing algorithms for time series fall short in this respect for this application.

The main contribution of this paper is a new privacy mechanism that remedies these limitations and is sufficiently precise for use in forecasting and optimization applications. The new algorithm, called OPTSTREAM, is presented under the framework of w-event privacy and is a 4-step procedure consisting of sampling, perturbation, reconstruction, and post-processing modules. The *sampling* module selects a small set of points for privacy-preserving measurement in each period of interest, the *perturbation* module introduces noise to the sampled data points to guarantee privacy, the *reconstruction* module reconstructs the non-sampled data points from the perturbed sampled points, and the *post-processing* module uses convex optimization over the privacy-preserving output of the previous modules, as well as the privacy-preserving answers of additional queries on the data stream, to redistribute the noise to ensure consistency of salient features of the data. OPTSTREAM is also generalized to answer queries over hierarchical streams, allowing data curators to monitor simultaneously streams produced by energy profile data at different levels of aggregation. It is important to emphasize that, although OPTSTREAM was motivated by an energy application, it is potentially useful for many other domains since its design is independent of the underlying problem.

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 w-event privacy. 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 privacy-preserving streams and the original stream but also in the accuracy of a load forecasting algorithm based on the privacy-preserving data. Finally, the paper shows that the sampling and optimization-based post-processing steps are critical in achieving the desired performance and that the improvements are also observed when releasing hierarchical streams of 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 OPTSTREAM and the design choices of its components. Section 4 analyzes the accuracy of the proposed framework and shows how it reduces the error introduced to preserve privacy when compared to a standard solution. Section 5 extends OPTSTREAM to the α-*indistinguishability* privacy model, allowing privacy protection of arbitrary quantities and which is critical for the motivating application. Additionally, OPTSTREAM is extended to handle hierarchical stream data.

Section 6 performs a comprehensive experimental analysis of real data streams from energy load profiles. Section 7 discusses key aspects of the privacy model adopted to privately releasing streams of data, as well as differences with the event-based model for data streams. Section 8 discusses the related work and, finally, Section 9 concludes the work.

2. Preliminaries

This section first reviews basic concepts in differential privacy. It then presents the w-event privacy model used to protect privacy in data streams and its definition of differential privacy.

2.1 Differential Privacy

This section reviews the standard definition of *differential privacy* (Dwork et al., 2006). Differential privacy focuses on protecting the participation of an individual user in a computation. Such notion relies on the definition of a *adjacency relation* $\sim$ between datasets. Two datasets X, X' are called $neighbors$ if their content differs in at most one tuple: $X \sim X' \Leftrightarrow \|X - X'\| \leqslant 1$.

Definition 1 (Differential Privacy) *Let $\mathcal{A}$ be a randomized algorithm that takes as input a dataset and outputs an element from a set of possible responses. $\mathcal{A}$ achieves ϵ-differential privacy if, for all sets $O \subseteq \mathcal{O}$ and all neighboring datasets $X, X' \in \mathcal{D}$:*

$$\frac{Pr[\mathcal{A}(X) \in O]}{Pr[\mathcal{A}(X') \in O]} \leqslant \exp(\epsilon). \tag{1}$$

The level of privacy is controlled by the parameter $\epsilon \geqslant 0$, describing the *privacy loss*, with values close to 0 denoting strong privacy. The adjacency relation $\sim$ captures the participation of an individual into the dataset. While differential privacy algorithms commonly adopts the 1-Hamming distance as adjacency relation between datasets, the latter can be generalized to be any symmetric binary relation $\sim \subseteq \mathcal{D}^2$. In particular, the relation $\sim$ captures the difference in the data D caused by the addition or removal a single individual and has been generalized to protect locations of individuals (Fawaz & Shin, 2014) and quantities in general (Chatzikokolakis, Andrés, Bordenabe, & Palamidessi, 2013). When a single entry is associated with a user in the dataset, an algorithm satisfying Equation (1) prevents an attacker with access to the algorithm's output from learning anything substantial about any individual.

2.1.1 PROPERTIES OF DIFFERENTIAL PRIVACY

Differential privacy enjoys several important properties, including composability and immunity to post-processing.

Composability *Composability* ensures that a combination of differentially private algorithms preserve differential privacy (Dwork & Roth, 2013).

Theorem 1 (Sequential Composition) *The composition $\mathcal{A}(D) = (\mathcal{A}_1(D), \ldots, \mathcal{A}_k(D))$ of a collection $\{\mathcal{A}_i\}_{i=1}^k$ of ϵ_i-differentially private algorithms satisfies $(\sum_{i=1}^k \epsilon_i)$-differentially privacy.*

Theorem 2 (Parallel Composition) *Let D_1 and D_2 be disjoint subsets of D, obtained through a* data independent *process, and $\mathcal{A}$ be an ϵ-differentially private algorithm. Then computing $\mathcal{A}(D \cap D_1)$ and $\mathcal{A}(D \cap D_2)$ satisfies ϵ-differential privacy.*

Post-processing Immunity *Post-processing immunity* ensures that privacy guarantees are preserved by arbitrary, data independent, post-processing steps (Dwork & Roth, 2013).

Theorem 3 (Post-Processing Immunity) *Let $\mathcal{A}$ be an ϵ-differentially 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-differentially private.*

We now introduce two useful differentially private algorithms.

2.1.2 THE LAPLACE MECHANISM

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

$$\Delta_Q = \max_{D \sim D'} \left\| Q(D) - Q(D') \right\|_1 .$$

In other words, the sensitivity of a query is the maximum L_1-distance between the query outputs from any two neighboring dataset D and D'. For instance, $\Delta_Q = 1$ for a query Q that counts the number of users in a dataset.

The Laplace distribution with 0 mean and scale b, denoted by $\mathit{Lap}(b)$, has a probability density function $\mathit{Lap}(x|b) = \frac{1}{2b} e^{-\frac{|x|}{b}}$. It can be used to obtain an ϵ-differentially private algorithm to answer numeric queries (Dwork et al., 2006). In the following, we use $\mathit{Lap}(\lambda)^d$ to denote the i.i.d. Laplace distribution over d dimensions with parameter λ.

Theorem 4 (Laplace Mechanism $\mathcal{M}_{\mathit{Lap}}$) *Let Q be a numeric query that maps datasets to $\mathbb{R}^d$. The Laplace mechanism that outputs $Q(D) + z$, where $z \in \mathbb{R}^d$ is drawn from the Laplace distribution $\mathit{Lap}\left(\frac{\Delta_Q}{\epsilon}\right)^d$, achieves ϵ-differential privacy.*

2.1.3 SPARSE VECTOR TECHNIQUE

The Sparse Vector Technique (SVT) is an important tool of differential privacy (Dwork & Roth, 2013; Hardt & Rothblum, 2010) that allows answering a sequence of queries without incurring a high privacy cost. The SVT mechanism is given a sequence of queries $Q = q_1, q_2, \ldots$ and a sequence of real valued thresholds $\Theta = \theta_1, \theta_2, \ldots$ and outputs a vector indicating whether each query answer q_i is above or below the corresponding threshold θ_i. In other words, the output is a vector $\{\top, \bot\}^\ell$ where ℓ is the number of queries answered and $\top$ (resp. $\bot$) indicates that the answer to a noisy query is (resp. is not) above a noisy threshold.

The SVT mechanism is summarized in Algorithm 1. It takes as input a dataset D, a sequence of queries $Q = q_1, q_2, \ldots$, each with sensitivity no more than Δ_Q, a sequence of thresholds $\Theta = \theta_1, \theta_2, \ldots$, and a constant k, denoting the maximum number of queries to be answered with value $\top$. Its output consists of a sequence of answers $a_1, a_2, \ldots$ with each $a_i \in \{\top, \bot\}$. For each query, SVT perturbs the corresponding threshold and checks if the perturbed individual query answer is above the noisy threshold.

Theorem 5 (SVT) *The SVT mechanism achieves ϵ-differential privacy (Hardt & Rothblum, 2010).*

The SVT mechanism is useful especially in situations when one expects that most answers fall below the threshold, since the noise depends on k. The SVT mechanism will be used in this paper to find good sample points in a stream.

Algorithm 1: SVT - *Sparse Vector Technique*

input : D: the dataset
$Q = q_1, q_2, \ldots$: a sequence of queries
$\Theta = \theta_1, \theta_2, \ldots$: a sequence of thresholds
k: the maximum number of queries to be answered positively
ϵ: the privacy loss

```
ρ = Lap(2Δ_Q/ε);
count = 0;
foreach query q_i ∈ Q do
    ν_i = Lap(4kΔ_Q/ε);
    if q_i(D) + ν_i ⩾ θ_i + ρ then
        output a_i = ⊤;
        count = count + 1;
        Break if count ⩾ k;
    else
        output a_i = ⊥;
```

2.2 The w-Event Privacy Model for Data Streams

This section presents the privacy model for streams adopted in this paper. A *data stream* D is an infinite sequence of elements in the *data universe* $\mathscr{U} = \mathcal{I} \times \mathcal{T}$, where $\mathcal{I}$ denotes the set of user identifiers and $\mathcal{T}$ is a possibly unbounded set of time steps. In other words, each tuple (i, t) describes an event reported by user i that occurred at time t. Time is represented through discrete steps $\mathcal{T} = \{1, 2, \ldots\}$ and user events are recorded periodically (e.g., every 30 minutes). In a data stream D, tuples are ordered by arrival time. If tuple (i, t) arrives after tuple (i', t'), then $t \geqslant t'$. Additionally, 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. Additionally, $\mathcal{D}$ denotes the set of all datasets describing collections of tuples in $\mathscr{U}$.

Example 1 *Consider a data stream system that collects location data from WiFi access points (APs) in a collection of buildings. Users correspond to MAC addresses of individual devices that connect to an AP. An item (i, t) in the data stream represents the fact that user i has made at least one connection to an access point at time step t. In the following table, user `02:18:98:09:1a:a4` reports a connection at times $t = 1$ and $t = 2$. The example represents a data stream prefix $D[3]$.*

D_1	D_2	D_3	$\ldots$
02:18:98:09:1a:a4	*02:18:98:09:1a:a4*	*05:12:11:0a:30:03*	
05:12:11:0a:30:03	*05:12:11:0a:30:03*	*11:28:18:a9:1b:a3*	
11:28:18:a9:1b:a3	*11:28:18:a9:1b:a3*	*07:22:1a:12:31:33*	
	07:22:1a:12:31:33	*01:11:e2:14:43:b2*	
	01:11:e2:14:43:b2		

The data curator receives information from an unbounded data stream D in discrete time steps. At time t, the curator collects a dataset D_t of tuples (i, t) where every row corresponds to a *unique* user. The curator reports the result of a count query $Q : \mathcal{D} \to \mathbb{R}$ that describes how many users are in the dataset at time t, i.e., $Q(D_t) = |\{i : (i, t) \in D_t\}|$.

Example 2 *Consider the data stream prefix of Example 1. The associated result of the count queries executed onto each dataset of the data stream is given in the following table.*

$Q(D_1)$	$Q(D_2)$	$Q(D_3)$	...
3	5	4	

In our target application, the data curator is interested in publishing every element $Q(D_t)$ for a recurring period of w time steps. A *w-period* is a set of w contiguous time steps $t-w+1,\ldots,t$ ($w \geqslant 1$). Thus, the answers to each query $Q(D_t)$ are generated in real time for windows of w time steps. As a result, this paper adopts the w-event privacy framework by Kellaris et al. (2014).

2.3 Differential Privacy on Streams

The w-event privacy framework (Kellaris et al., 2014) extends the definition of differential privacy to protect data streams and has become a standard privacy notion for data streams (see, for instance, Rastogi & Nath, 2010; Dwork & Roth, 2013; Andrés, Bordenabe, Chatzikokolakis, & Palamidessi, 2013; Bolot, Fawaz, Muthukrishnan, Nikolov, & Taft, 2013; Chan, Shi, & Song, 2011). The framework operates on stream prefixes and two data streams prefixes $D[t]$ and $D'[t]$ are *w-neighbors*, denoted by $D[t] \sim_w D'[t]$, if

i. for each D_i, D'_i with $i \in [t]$, $D_i \sim D'_i$, and

ii. for each D_i, D'_i, D_j, D'_j such that $i < j \in [t]$ and $D_i \neq D'_i, D_j \neq D'_j$, $j - i + 1 \leqslant w$ holds.

In other words, two stream prefixes are w-neighbors if their elements are *pairwise* neighbors and all the differing elements are within a time window of up to w time steps. As a result, when ensuring the privacy guarantees, the w-event framework does not consider data streams where the differences are beyond a time window of size w: It only needs to consider windows of w elements.

Definition 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 $\mathcal{O}$. Algorithm $\mathcal{A}$ satisfies w-event ϵ-differential privacy (w-privacy for short) if, for all t, all sets $O \subseteq \mathcal{O}$, and all w-neighboring stream prefixes $D[t]$ and $D'[t]$:*

$$\frac{Pr[\mathcal{A}(D[t]) \in O]}{Pr[\mathcal{A}(D'[t]) \in O]} \leqslant \exp(\epsilon). \tag{2}$$

An algorithm satisfying w-privacy protects the sensitive information that could be disclosed from a sequence of finite length w. When $w = 1$, w-privacy reduces to event-level privacy (Dwork et al., 2010) that protects the disclosure of events in a single time step.

All the properties of differential privacy discussed above carry over to w-privacy. The Laplace mechanism which adds Laplace noise to each element of the stream with parameter $w\Delta_Q/\epsilon$ achieves w-privacy (Kellaris et al., 2014).

The algorithms studied in this paper satisfy Definition 2. When clear from the context, the paper uses ϵ-differential privacy to denote w-event privacy. A list of common symbols is summarized in Table 1.

Symbol	Definition	Symbol	Definition
$\mathcal{U}$	Data universe	$\sim_w$	Neighboring relation
D	Data stream	x_t	Result of query $Q(D_t)$
D_t	Dataset at time t	$\hat{x}_t$	Privacy-preserving estimate of x_t
$D[t]$	Stream prefixes	$\boldsymbol{x}$	Data stream summary
Q	A query on a dataset	$\hat{\boldsymbol{x}}$	Privacy-preserving data stream summary
Δ_Q	Sensitivity of query Q	r_u^t	User's u value at time t

Table 1: Commonly used symbols and notations.

3. OptStream For Stream Release

This section describes OPTSTREAM, a novel algorithm for privacy-preserving data stream release. OPTSTREAM consists of four steps: (1) data sampling, (2) perturbation, (3) reconstruction of the non-sampled data points, and (4) optimization-based post-processing. The algorithm takes as input the data stream, denoted by $\mathbf{D}^I = D_1^I, D_2^I, \ldots$, the period size w whose privacy is to be protected, the privacy loss ϵ, and some hyper-parameters used by its procedures, which will be described later. Its output is a data stream *summary* $\hat{\boldsymbol{x}} = (\hat{x}_1, \hat{x}_2, \ldots)$ where each $\hat{x}_t$ represents a privacy-preserving version of the aggregated real data in D_t^I. Here, an aggregation operation is one that transforms the dataset into a numerical representation (e.g., a count). The four steps can be summarized as follows:

1. SAMPLE: Selects a subsample of k points for each w-period. Its goal is to perform a dimensionality reduction over the data stream whose sample points can be used to generate privacy-preserving answers to each query with low error. Two sampling strategies are considered: One that chooses k equally-spaced points in the w-period, and another that selects k points such that the linear interpolation between their values minimizes the error.

2. PERTURB: Uses the canonical Laplace mechanism to guarantee privacy perturbing the k data points sampled in Step 1.

3. RECONSTRUCT: Reconstructs the non-sampled data points values from the perturbed sampled ones. Its goal is to map the dimension-reduced data stream back to the original space, generating thus w data points.

4. POST-PROCESS: Uses convex optimization to enforce consistency of salient features of the data, employing the privacy-preserving output of the above modules, as well as privacy-preserving answers to additional queries on the data stream.

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

OPTSTREAM processes the data stream in consecutive and disjoint w-periods. To simplify notation, throughout this section, the paper uses $\mathbf{D} = D_1, \ldots, D_w$ and $\boldsymbol{x} = x_1, \ldots, x_w$ to denote,

Algorithm 2: OPTSTREAM *data stream release*

input : $\mathbf{D}^I = (D_1^I, D_2^I, \ldots)$: the data stream
w: the size of the period
ϵ: the privacy loss
$k, \theta, \mathscr{F}$: hyperparameters
output: $\hat{\mathbf{x}} = (\hat{x}_1, \hat{x}_2, \ldots)$: a privacy-preserving data stream summary
Let t be the current time step
1 **if** $t \pmod w \equiv 0$ **then**
2 $\quad \mathbf{D} \leftarrow D_{t-w+1}^I, \ldots D_t^I$
3 $\quad$ RELEASE ($\mathbf{D}$, k, w, ϵ, θ, $\mathscr{F}$)

Function RELEASE ($\mathbf{D}$, w, ϵ, k, θ, $\mathscr{F}$) **:**
4 $\quad \boldsymbol{x} \leftarrow aggr(D_1), \ldots aggr(D_w)$
5 $\quad \epsilon_s, \epsilon_p, \epsilon_o \leftarrow$ split loss ϵ
6 $\quad S = \text{SAMPLE}(\boldsymbol{x}, \epsilon_s, k, \theta)$
7 $\quad \tilde{\boldsymbol{x}}_S = \text{PERTURB}(\boldsymbol{x}[S], \epsilon_p)$
8 $\quad \tilde{\boldsymbol{x}} = \text{RECONSTRUCT}(\tilde{\boldsymbol{x}}_S, w)$
9 $\quad \hat{\boldsymbol{x}} = \text{POSTPROCESS}(\tilde{\boldsymbol{x}}, \epsilon_o, \{Q_{\mathbf{F}}(\mathbf{D})\}_{\mathbf{F} \in \mathscr{F}})$
10 $\quad$ release $\hat{\boldsymbol{x}}$

respectively, the current w-period being processed, and its univariate discrete series representation, where each x_i denotes the result of an aggregation function $aggr(D_i)$ that transforms the dataset into a numerical representation. Similarly, we use $\mathbf{D}'$ and $\boldsymbol{x}'$ to denote their neighboring counterparts. Additionally, given a set of time steps indexes S, $\boldsymbol{x}[S]$ denotes the collection of points $\{x_i | i \in S\}$. For simplicity, we will assume that the sensitivity of the aggregation operations is one. The results directly generalize to arbitrary sensitivities.

OPTSTREAM is depicted in Algorithm 2. When a new w-period is observed (line 1), the algorithm extracts the relevant portion of the stream (line 2) and calls function RELEASE, which releases a privacy-preserving version of the data stream in the current w-period (line 3).

In addition to the portion $\mathbf{D}$ of the data stream to release, the size of the w-period w, and the privacy loss ϵ, function RELEASE takes, as inputs, three hyperparamers: k, θ, and $\mathscr{F}$. Parameters k and θ are used by procedure SAMPLE and represent the maximum number of data points to extract in the w-period and a threshold value, respectively. Parameter $\mathscr{F}$ is a set of *features queries* used by procedure POSTPROCESS; these queries and their uses will be covered in detail later. The four steps of Function RELEASE operate on a discrete series representation $\boldsymbol{x}$ of the data stream (line 4). The privacy loss to be used in each step is computed in line 5. Procedure SAMPLE takes as input the sequence of values $\boldsymbol{x}$, the maximum number k of data points to sample, and the portion $\epsilon_s \geqslant 0$ of the privacy loss ϵ used in the sampling process[1] (line 6). It outputs the set S of indexes associated with the values of $\boldsymbol{x}$ whose privacy must be protected. Procedure PERTURB takes, as input, the vector $\boldsymbol{x}[S]$ of sampled data points from $\boldsymbol{x}$ and outputs a noisy version $\tilde{\boldsymbol{x}}_S$ of $\boldsymbol{x}$, using a portion $\epsilon_p > 0$ of the overall privacy loss (line 7). Procedure RECONSTRUCT takes, as inputs, the vector $\tilde{\boldsymbol{x}}_S$ from the perturbation step and the size w of the period and outputs a vector $\tilde{\boldsymbol{x}}$ of size

1. ϵ_s can be 0 if the sampling processes does not access the real data to make a decision on which points to sample.

w whose values are privacy-preserving estimates of the data stream in the w-period (line 8). Next, procedure POSTPROCESS takes as input the vector of points $\tilde{\boldsymbol{x}}$, additional feature queries $Q_{\mathbf{F}}(\mathbf{D})$ for each feature $\mathbf{F}$ in the set of data features $\mathscr{F}$ (which are defined in detail in Section 3.4), and uses a portion $\epsilon_o > 0$ of the overall privacy loss ϵ to compute a final estimate $\hat{\boldsymbol{x}}$ of $\boldsymbol{x}$ (line 9). Finally, the privacy-preserving estimates $\hat{\boldsymbol{x}}$ are released (line 10).

Lemma 1 *Let $\epsilon_s + \epsilon_p + \epsilon_o = \epsilon$. When the* SAMPLE, PERTURB, *and* POST-PROCESS *procedures satisfy ϵ_s-, ϵ_p-, and ϵ_o-differential privacy, respectively, Algorithm 2 satisfies ϵ-differential privacy.*

Any sampling, perturbation, and reconstruction algorithms can be used within Algorithm 2, 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 may reduce the error (Section 4) and are shown to perform well experimentally (Section 6). The perturbation procedure is a canonical application of the Laplace mechanism (Theorem 4) on the set of sampled data points and parallel composition (Theorem 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 of the given w-period that summarize the entire data stream period well. This section considers two strategies.

Equally-Spaced Sampling A first strategy is to sample k equally-spaced data points in the w-period. Since this approach does not inspect the values of the data stream to make its decisions, it does not consume any privacy loss, i.e., $\epsilon_s = 0$.

L_1-Based Sampling The second strategy also selects k out of w points but tries to minimize the error between the values associated to the original data points and those associated to the points generated by a linear interpolation of the k selected points. Let $\xi^{[i,j]}$ be a function capturing the line segment between two points x_i and x_j, i.e.,

$$\xi^{[i,j]}(t) = (t-i)\frac{x_j - x_i}{j-i} + x_i$$

for $t \in [i, j]$. Consider an ordered sequence $S = (\iota_1, \ldots, \iota_k)$ of k indexes in $[w]$ and define ξ^S as a piecewise linear function whose pieces are line segments between every two adjacent points in S,

$$\xi^S(t) = \begin{cases} \xi^{[\iota_1,\iota_2]}(t) & \text{if } t \in [\iota_1, \iota_2] \\ \xi^{[\iota_2,\iota_3]}(t) & \text{if } t \in [\iota_2, \iota_3] \\ \quad\vdots & \\ \xi^{[\iota_{k-1},\iota_k]}(t) & \text{if } t \in [\iota_{k-1}, \iota_k]. \end{cases}$$

Define the $L_1^{[a,b]}$-scoring function for $1 \leqslant a < b \leqslant w$ as,

$$L_1^{[a,b]}(\mathbf{D}) = \sum_{i=a}^{b} \left| \xi^{[a,b]}(i) - x_i \right|. \tag{3}$$

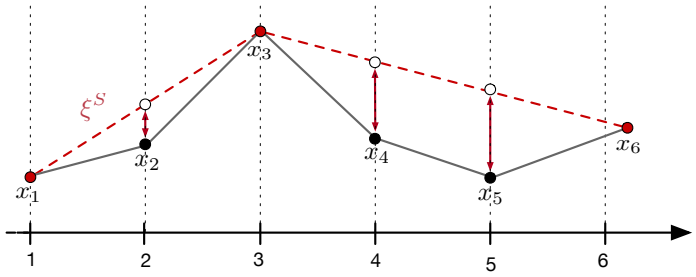


Figure 1: Illustration example of the L_1^S function with $S = (1, 3, 6)$. The solid black curve connects the data point $\boldsymbol{x}$, the dashed red curve denotes the function ξ^S, and the red arrows denote the distance between the points in $\boldsymbol{x}$ and the function ξ^S. The $L_1^S(\mathbf{D})$ value is the sum of these distances.

The L_1*-based sampling* procedure aims at selecting a sequence S of k indexes $\iota_1 < \iota_2 < \ldots < \iota_k$ in a w-period that minimizes the L_1^S*-scoring function* defined as,

$$\min_S \mathrm{L}_1^S(\mathbf{D}) = \min_{\iota_1 < \iota_2 < \ldots < \iota_k} \sum_{j=1}^{k-1} L_1^{[\iota_j, \iota_{j+1}]}(\mathbf{D}). \tag{4}$$

Figure 1 provides an example with a graphical illustration of the L_1^S function. The black solid curve delineate $\boldsymbol{x}$. The set $S = (1, 3, 6)$ and the sampled points x_1, x_3, and x_6 are colored red. ξ^S is the piecewise linear function passing through the sampled points and is represented with dashed red lines. The red arrows denote the distance between the points in $\boldsymbol{x}$ and their corresponding values in ξ^S and the L_1^S-score for the set S is the sum of these distances. The procedure assumes that the first and the last points of $\boldsymbol{x}$ are in S.

Intuitively, the set of k points that minimizes Equation (4) produces the minimal *reconstruction error* when linear interpolation is used as a reconstruction procedure. However, finding the set of k points minimizing the L_1-scoring function may be computationally expensive. Hence this paper presents a differentially private greedy algorithm that approximates Equation (4). The sampling procedure is depicted in Algorithm 3 and is an instantiation of the SVT mechanism where the queries compute the L_1-scores of the potential interpolation steps.

The mechanism takes, as inputs, the data stream $\boldsymbol{x}$ processed for the current w-period, the number of points k to sample for measurements, along with the privacy loss ϵ_s and a user defined threshold $\theta \geqslant 0$ which influences the acceptable L_1-score for choosing the next point to sample. The values choice for θ in our application of interest are detailed in Section 6. Line 1 initializes the set of sample points S 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 point t_p selected for sampling. The algorithm first generates the noise ρ (line 2) for the threshold θ (line 5). For all but the first time step, the algorithm adds Laplace noise with parameter $4k\Delta_L/\epsilon$ to the $L_1^{[t_p,i]}$-query (line 4), where Δ_L is the largest sensitivity associated with any of the L_1 scoring functions invoked by the algorithm (see Theorem 6). If the result is above the threshold, then point i is added to S (line 6) and the last selected point t_p is updated (line 7). The mechanism keeps track of the number of index points

Algorithm 3: SAMPLE - *Adaptive L_1-based sampler*

input : $\boldsymbol{x}$: the data stream seen in the current w-period
k: the number of samples
ϵ_s: the privacy loss
θ: a threshold
output: S: A sample set of k points in $[w]$

1 $S = \{1\};\ t_p = 1$
2 $\rho = \text{Lap}(2\Delta_L/\epsilon_s)$
3 **for** $i = 2 \ldots, w$ **do**
4 $\quad \mu_i = \text{Lap}(4k\Delta_L/\epsilon_s)$
5 $\quad$ **if** $L_1^{[t_p,i]}(\boldsymbol{x}) + \mu_i \geqslant \theta + \rho$ **then**
6 $\quad\quad S = S \cup \{i\}$
7 $\quad\quad t_p = i$
8 $\quad$ **if** $w - i \leqslant k - |S|$ **then**
9 $\quad\quad S = S \cup \{j \mid i < j \leqslant w\}$
10 $\quad$ **Break** if $|S| = k$
11 **return** S

already stored. It stops when the size of S matches k (line 10). The mechanism also tests if there are enough points to reach k (line 8) and adds the remaining points if needed (line 9).

To run the mechanism, it is necessary to determine the sensitivity $\Delta_{L_1}^{[a,b]}$ of the L_1-score defined in Equation (3), i.e.,

$$\Delta_{L_1}^{[a,b]} = \max_{\mathbf{D} \sim_w \mathbf{D}'} \left| L_1^{[a,b]}(\mathbf{D}') - L_1^{[a,b]}(\mathbf{D}) \right|$$

where $\mathbf{D}$ and $\mathbf{D}'$ are two w-neighboring data streams in a w-period.

The following theorem shows that $\Delta_{L_1}^{[a,b]}$ can be bounded, yielding a procedure to privately sample k points in the w-period using an efficient, suboptimal, version of Equation (4).

Theorem 6 *For an arbitrary w-period and fixed indexes $a, b \in [w]$ with $a < b$, the sensitivity $\Delta_{L_1}^{[a,b]}$ of the $L_1^{[a,b]}$ score is bounded by $2(b-a)$.*

Proof. Consider two data stream w-periods $\mathbf{D}$ and $\mathbf{D}'$ such that $\mathbf{D} \sim_w \mathbf{D}'$ and focus, without loss of generality, on their associated stream counts $\boldsymbol{x}$ and $\boldsymbol{x}'$. Let ξ and ξ' be shorthands for $\xi^{[a,b]}$ and $\xi'^{[a,b]}$. The goal is to bound

$$\left| L_1^{[a,b]}(\mathbf{D}') - L_1^{[a,b]}(\mathbf{D}) \right| = \sum_{i=a}^{b} \left| |\xi'(i) - x'_i| - |\xi(i) - x_i| \right|,$$

and each term of the summation can be bounded independently. If $i = a \vee i = b$, then $\xi'(i) = x'_i$ and $\xi(i) = x_i$, since x'_i and x_i are interpolated exactly and $\left| |\xi'(i) - x'_i| - |\xi(i) - x_i| \right| = 0$. Otherwise, note that $|x'_i - x_i| \leqslant 1$ since $D_i \sim D'_i = 1$ for all $i \in [w]$ by definition of w-event privacy. Moreover, since the pairs $(\xi(a), \xi'(a))$ and $(\xi(b), \xi'(b))$ differ by at most 1, it follows that $|\xi'(i) - \xi(i)| \leqslant 1$ for every point $i \in [a, b]$. There are four cases to consider.

(*1*) If $\xi'(i) \geqslant x'_i$ and $\xi(i) \geqslant x_i$, then

$$\begin{aligned}\big||\xi'(i) - x'_i| - |\xi(i) - x_i|\big| &= \big|(\xi'(i) - x'_i) - (\xi(i) - x_i)\big| \\ &= \big|(\xi'(i) - \xi(i)) - (x'_i - x_i)\big| \\ &\leqslant 2.\end{aligned}$$

(*2*) The case $\xi'(i) < x'_i$ and $\xi(i) < x_i$ is symmetric. (*3*) The case $\xi'(i) \geqslant x'_i$ and $\xi(i) < x_i$ requires a further case analysis.

(*i*) If $\xi'(i) \leqslant \xi(i)$, we have $x'_i \leqslant \xi'(i) \leqslant \xi(i) < x_i$. Since $|x_i - x'_i| \leqslant 1$, it follows that $|\xi'(i) - x'_i| \leqslant 1$ and $|\xi(i) - x_i| \leqslant 1$. Therefore

$$\big||\xi'(i) - x'_i| - |\xi(i) - x_i|\big| \leqslant 1.$$

(ii) If $\xi'(i) > \xi(i)$, then $\xi(i) + 1 \geqslant \xi'(i) > \xi(i)$, since $|\xi(i) - \xi'(i)| \leqslant 1$. It follows that

$$x'_i \leqslant \xi'(i) \leqslant \xi(i) + 1 < x_i + 1.$$

Since $|x_i - x'_i| \leqslant 1$ and $|\xi(i) - \xi'(i)| \leqslant 1$, $|\xi'(i) - x'_i| \leqslant 1$ and $|\xi(i) - x_i| \leqslant 1$ and therefore

$$\big||\xi'(i) - x'_i| - |\xi(i) - x_i|\big| \leqslant 1.$$

(*4*) Finally, the last case, $\xi'(i) < x'_i$ and $\xi(i) \geqslant x_i$, is symmetric to previous one. Therefore

$$\Delta_{L_1}^{[a,b]} = \max_{\boldsymbol{x} \sim_w \boldsymbol{x}'} |L_1^{[a,b]}(\mathbf{D}') - L_1^{[a,b]}(\mathbf{D})| \leqslant 2(b-a).$$

□

Let $S = (\iota_1, \ldots, \iota_k)$ be a sequence of k indexes in $[w]$. Because of the stopping criteria, the largest contiguous interval $[\iota_i, \iota_j]$ for $i, j \in [k]$ has size $w - k$. The sensitivity of the L_1-score on such an interval is bounded by $2(w-k)$. Algorithm 3 thus uses

$$\Delta_L = 2(w-k).$$

Theorem 7 *Algorithm 3 is ϵ_s-differentially private.*

Proof. The proof is a direct consequence of the correctness of the SVT mechanism (Hardt & Rothblum, 2010), Theorem 6, and the definition of Δ_L. □

3.2 The Perturbation Procedure

Given the set S of k sampling indexes for a w-period, the perturbation process takes as input the k-ary vector of the data stream measurements $\boldsymbol{x}[S]$ and outputs a noisy version $\tilde{\boldsymbol{x}}_S$ of such vector satisfying ϵ_p-differential privacy. The process simply applies the Laplace mechanism with parameter $k\Delta_A/\epsilon_p$, where Δ_A is the sensitivity of the aggregation query.

Theorem 8 PERTURB *satisfy ϵ_p-differential privacy.*

The above result follows by straightforward application of the Laplace mechanism (Theorem 4) on a set of k points and parallel composition (Theorem 2).

Algorithm 4: POSTPROCESS – *Optimization-based post-processing*

input : $\boldsymbol{x}$: the data stream seen in the current w-period
$\tilde{\boldsymbol{x}}$: the privacy-preserving data stream seen in the current w-period
$Q_{\mathbf{F}}(\mathbf{D})$; the set of feature queries
ϵ_o: the privacy loss

output: $\hat{\boldsymbol{x}}$: a privacy-preserving post-processed data stream

1 $\tilde{\boldsymbol{c}}_1 = \tilde{\boldsymbol{x}}$

2 $\tilde{\boldsymbol{c}}_i = \mathcal{M}_{Lap}(\boldsymbol{x}; Q_{\mathbf{F}_i}, \epsilon/(p-1)) \quad \forall i \in \{2, \ldots, p\}$

3 $\boldsymbol{x}^* = \mathbf{argmin}_{\dot{\boldsymbol{x}}} \|\dot{\boldsymbol{x}} - \tilde{\boldsymbol{c}}\|^2_{2,\lambda} =$

$$\sum_{i=1}^{p} \lambda_i \sum_{j=1}^{m_i} (\dot{x}_{ij} - \tilde{c}_{ij})^2 \tag{O1}$$

subject to :

$$\forall i', i : \mathbf{F}_{i'} \prec \mathbf{F}_i, \quad j \in [m_i] : \dot{x}_{ij} = \sum_{l:\mathbf{d}_{i'l} \subseteq \mathbf{d}_{ij}} \dot{x}_{i'l} \tag{O2}$$

$$\forall i, j : \dot{x}_{ij} \geqslant 0. \tag{O3}$$

return $\hat{\boldsymbol{x}} = \dot{x}_{11}, \ldots, \dot{x}_{1w}$

3.3 The Reconstruction Procedure

The RECONSTRUCT procedure takes as input the noisy measurements $\tilde{\boldsymbol{x}}_S \in \mathbb{R}^k$ at the sample points S in $\boldsymbol{x}$ and outputs a vector $\tilde{\boldsymbol{x}} \in \mathbb{R}^w$ of privacy-preserving estimates for the sub-stream $\boldsymbol{x}$. Each value $\tilde{x}_i$ of $\tilde{\boldsymbol{x}}$ is obtained evaluating the function ξ^S at i. Section 4 analyzes how well the polynomial approximates the data stream at any point x_i. The reconstruction procedure is not required to query the real data stream and uses exclusively privacy-preserving information to compute its output. Hence, the output $\tilde{\boldsymbol{x}}$ remains $\epsilon_s + \epsilon_p$-differentially private by post-processing immunity of differential privacy (Theorem 3).

3.4 The Optimization-based Post-Processing

The noise introduced in the previous steps may substantially alter the values of the elements in the w-period, so that some global properties of interest, such as the total sum of elements in the w-period may differ from its original value. The goal of the post-processing step is to redistribute the noise introduced by the previous steps using noisy information on aggregated values of the w-period. It does so by casting the noise redistribution as an optimization problem whose solution guarantees the consistency of different estimates of identical quantities.

The POST-PROCESS step computes the final estimates $\hat{\boldsymbol{x}}$ of the w-period data $\boldsymbol{x}$ using the privacy-preserving data $\tilde{\boldsymbol{x}}$ and additional queries over $\boldsymbol{x}$. The procedure is summarized in Algorithm 4. It uses the concept of *features* to capture semantic properties of the application of interest and queries these features in addition to using the noisy input $\tilde{\boldsymbol{x}}$ of the original data stream. For example, two important features in the analysis of WiFi connections profiles are those periods when peaks typically occur, as well as the total amount of connections occurring in an w-period.

Formally, a *feature* is a partition of the w-period and the size of the feature is the number of elements in the partition. We say that 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* $Q_{\mathbf{F}}(\mathbf{D})$ on data stream $\mathbf{D}$ 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 of $\boldsymbol{x}$ for $j \in \mathbf{d}_i$.

Example3 *Consider a w-period $\boldsymbol{x} = (10, 15, 20, 23, 41, 72, 55, 50, 88, 72, 40, 18)$ of size 12 reporting the number of WiFi connections to an access point in a day-period. Consider a feature $\mathbf{F}_1 = \{1, \ldots, 12\}$ that includes the indexes for all the elements of the w-period. The associated feature query $Q_{\mathbf{F}_1}(\mathbf{D}) = (10, 15, 20, 23, 41, 72, 55, 50, 88, 72, 40, 18)$ returns each frequency value observed during the day. Next, we consider a sub-feature $\mathbf{F}_2$ of $\mathbf{F}_1$, defined as $\mathbf{F}_2 = \{\{1, 2, 3, 4\}, \{5, 6, 7, 8, 9\}, \{10, 11, 12\}\}$. Its associated feature query $Q_{\mathbf{F}_2}(\mathbf{D}) = (68, 306, 130)$ describes the sums of all connection frequencies associated with time steps in $\{1, 2, 3, 4\}$, $\{5, 6, 7, 8, 9\}$, and $\{10, 11, 12\}$, representing morning, afternoon, and evening hours respectively.*

The optimization-based post-processing takes as input the noisy data stream $\tilde{\boldsymbol{x}}$ from the reconstruction procedure and a collection of features queries $Q_{\mathbf{F}}(\mathbf{D})$ 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., $\mathbf{F}_1 = \{\{i\} : i \in [w]\}$. The noisy answer to this query is the output $\tilde{\boldsymbol{x}}$ of the perturbation procedure (line 8 of Algorithm 2). When viewed as queries, the inputs to the mechanism can be represented as a set of values $Q_{\mathbf{F}_i}(\mathbf{D}) = \mathbf{c}_i = (c_{i1}, \ldots, c_{im_i})$ $(1 \leqslant i \leqslant p)$ or, more concisely, as $\boldsymbol{c} = (c_{11}, \ldots, c_{pm_p})$. We assume that the partial ordering $\prec$ of features is given. Finally, notice that the feature queries $Q_{\mathbf{F}}(\mathbf{D})$ form a lattice on D.

The first step of Algorithm 4 (line 1) associates the noisy data stream $\tilde{\boldsymbol{x}}$ to the vector $\boldsymbol{c}_1$, corresponding to the query answers associated to the first feature in $Q_{\mathbf{F}}$. This step is already privacy preserving and no additional noise is required. Line 2 applies the Laplace mechanism with privacy parameter $\frac{\epsilon}{p-1}$ to each remaining feature query, i.e.,

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

The resulting values $\tilde{\boldsymbol{c}} = (\tilde{c}_{11}, \ldots, \tilde{c}_{pm_p})$ are then post-processed by the optimization algorithm depicted in line 3 to obtain the values $\boldsymbol{x}^* = (x^*_{11}, \ldots, x^*_{pm_p})$. Finally, the mechanism outputs a data stream $\hat{\boldsymbol{x}} = (\dot{x}^*_{11}, \ldots, \dot{x}^*_{1w})$.

The essence of Algorithm 4 is the optimization model depicted in line 3. Its decision variables are the post-processed values $\dot{\boldsymbol{x}} = (\dot{x}_{11}, \ldots, \dot{x}_{pm_p})$, and $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_p) \in (0, 1]^p$ is a vector of reals representing weights for the terms of the objective function. The objective minimizes the squared weighted L_2-Norm of $\dot{\boldsymbol{x}} - \tilde{\boldsymbol{c}}$, where the weight λ_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 $\mathbf{d}_{ij}$.

Theorem 9 *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 count query with sensitivity 1. Thus, each $\tilde{c}_{ij}$ $(i > 1)$ obtained from the Laplace mechanism is ϵ_o-differentially private by Theorem 4 and the values $\tilde{\boldsymbol{x}} = \tilde{c}_{11}, \ldots, \tilde{c}_{1w}$ are differentially private (Theorems 7 and 8). Additionally, $(\tilde{c}_{11}, \ldots, \tilde{c}_{pm_p})$ is ϵ_o-differentially private by Theorem 1. Finally, the result follows from post-processing immunity (Theorem 3). □

Observe that the mechanisms considered in this paper all operate over the universe of the data stream in the 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 3 and hence the complexity of Algorithm 4. Recall that a δ-solution to an optimization problem is a solution whose objective value is within distance δ of the optimum.

Theorem 10 *A δ-solution to the optimization to the optimization model in line 3 (Algorithm 4) can be obtained in time polynomial in w, the number of features, and $\frac{1}{\delta}$.*

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 $\frac{1}{\delta}$ (Nemirovski, 2004). □

4. Error Analysis

This section analyzes how the sampling and reconstruction procedure can improve the accuracy of the output. It characterizes the error $\mathbb{E}\|\hat{\boldsymbol{x}} - \boldsymbol{x}\|_2^2$ of the results of Algorithm 2 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 $\tilde{\boldsymbol{x}}$ (line 8, Algorithm 2)).

4.1 Sample and Reconstruct

Consider a w-period of the data stream $\boldsymbol{x}$ using the same notational assumptions introduced in the previous period, and thus we focus in a w-period in $[1, w]$. Let $k = |S|$ 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 integer. Let L be the Lipschitz constant defined as $L := \sup_{t \in [w]} x_t - x_{t-1}$.

Theorem 11 *The error introduced by Algorithm 2 (ignoring post-processing) with the equally-spaced sampling procedure with parameter k is bounded by $O\left(m^2L^2w + 2\frac{w^2L}{\epsilon} + \frac{w^3}{m^2\epsilon^2}\right)$.*

Proof. For notational simplicity, consider the first interval $I = \{1, \ldots, m\}$ of the w-period, where the sample points 1 and m are selected. Thus, x_1 and x_m are measured privately. The reconstruct procedure uses linear interpolation between x_1 and x_m to recover the values of the non-sampling points $x_2, \ldots, x_{m-1}$.

For each point $i \in [m]$, there are two sources of error: the *perturbation error* e_p and the *reconstruction error* e_r. The worst-case reconstruction error is bounded by $e_r = mL$. The perturbation error e_p is the additive Laplace noise. There are k measurements taken, and hence the privacy loss ϵ must be divided by k, which results in $e_p = |\tilde{x}_i - x_i| = \text{Lap}(k/\epsilon)$. This error adds to the perturbation error at every point in the w-period. Therefore, for all $i \in [m]$, the expected error is:

$$\begin{aligned}\mathbb{E}\|\tilde{x}_i - x_i\|_2^2 &\leqslant \mathbb{E}\left[(e_r + e_p)^2\right] \\ &\leqslant \mathbb{E}_{Z\sim\text{Lap}(k/\epsilon)}\left[(mL + Z)^2\right] \\ &= m^2L^2 + 2mL\frac{k}{\epsilon} + (\frac{k}{\epsilon})^2.\end{aligned}$$

Multiplying this quantity by the number m of points in the interval gives

$$\mathbb{E}\|\tilde{\boldsymbol{x}}[I] - \boldsymbol{x}[I]\|_2^2 \leqslant m\mathbb{E}\|\tilde{x}_i - x_i\|_2^2 \leqslant m^3L^2 + 2m^2L\frac{k}{\epsilon} + m(\frac{k}{\epsilon})^2.$$

As a result, the error $\|\tilde{\boldsymbol{x}}[I] - \boldsymbol{x}[I]\|_2^2$ is bounded by $O(m^3L^2 + 2m^2L\frac{k}{\epsilon} + m(\frac{k}{\epsilon})^2)$. Multiplying the above by the number of intervals $\frac{w}{m}$ in the w-period gives the final error which is bounded by

$$O\left(m^2L^2w + 2\frac{w^2L}{\epsilon} + \frac{w^3}{m^2\epsilon^2}\right).$$

□

For $m = \sqrt{\frac{w}{\epsilon L}}$, the above expression generates an error of $O(w^2L/\epsilon)$.[2] In comparison, applying the Laplace mechanism to produce a privacy-preserving 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 2 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 $L1$-sampling procedure outperforms its equally-spaced counterpart.

4.2 Optimization-based Post-Processing

The following result is from Fioretto, Lee, and Van Hentenryck (2018). It bounds the error of the optimization-based post-processing. It proves that the post-processing step can accommodate any side-constraints without degrading the accuracy of the mechanism significantly. In the following $\|\cdot\|_{2,\boldsymbol{\lambda}}$ denotes the weighted 2-norm, where $\boldsymbol{\lambda}$ is a vector of real valued weights.

Theorem 12 *The optimal solution to the optimization model in line 9 of Algorithm 2 satisfies*

$$\|\boldsymbol{x}^* - \boldsymbol{x}\|_{2,\boldsymbol{\lambda}} \leqslant 2\|\tilde{\boldsymbol{x}} - \boldsymbol{x}\|_{2,\boldsymbol{\lambda}}.$$

Proof. It follows that:

$$\|\boldsymbol{x}^* - \boldsymbol{x}\|_{2,\boldsymbol{\lambda}} \leqslant \|\boldsymbol{x}^* - \tilde{\boldsymbol{x}}\|_{2,\boldsymbol{\lambda}} + \|\tilde{\boldsymbol{x}} - \boldsymbol{x}\|_{2,\boldsymbol{\lambda}}$$

2. Here, to simplify notation, we consider m as a real value.

$$\leqslant 2\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|_{2,\boldsymbol{\lambda}}.$$

where the first inequality follows from the triangle inequality on weighted L_2-norms and the second inequality follows from

$$\|\boldsymbol{x}^* - \tilde{\boldsymbol{x}}\|_{2,\boldsymbol{\lambda}} \leqslant \|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|_{2,\boldsymbol{\lambda}}$$

by optimality of $\boldsymbol{x}^*$ and the fact that $\boldsymbol{x}$ is a feasible solution to constraints (O2) and (O3). □

5. Privacy Model Extensions

This section first presents an extension of the privacy model that protects disclosure of arbitrary quantities within the w-event privacy model. It then generalizes the theoretical results of OPTSTREAM to this extended privacy model. Finally, it describes an extension of OPTSTREAM that supports hierarchical data streams.

5.1 α-Indistinguishability for Data Streams

The application studied in this paper requires the privacy-preserving release of energy load consumption streams. Unlike the previous setting, where the participation of users is revealed at each time step, the new privacy setting requires the *load consumption* of each customer to be revealed at each time step. To capture this requirement, the data stream setting presented in Section 2.2 is extended to an infinite sequence of elements in the *data universe* $\mathscr{U} = \mathcal{I} \times \mathcal{R} \times \mathcal{T}$, where $\mathcal{I}$ and $\mathcal{T}$ are defined as in Section 2.2, and $\mathcal{R}$ is the set of possible *events* (i.e., the data items generated by users). In other words, each tuple (i, r, t) describes an event that occurred at time t in which user i reported value r.

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 time period k. This setting implies that a tuple (i, r, t) describes the aggregate behavior of user i during the time period t. Here, a numerical count query $Q : \mathcal{D} \to \mathbb{R}^{|\mathcal{I}|}$ returns an $|\mathcal{I}|$-dimensional vector corresponding to the aggregated reports of each user in $\mathcal{I}$. The following example illustrates such behavior.

Example4 *Consider a data stream system that collects power consumption data from customers of an electric company distributed on a wide geographical region. Customers may 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 customer at time* t *is a real number denoting the average amount of power (in MegaWatts) the customer required during time interval* t*. The table below represents a the scenario in which user* $customer_1$ *consumes* 3.1 *MW at time* $t = 1$*,* 3.2 *MW at time* $t = 2$*, and* 2.8 *MW at time* $t = 3$*.*

$\mathcal{I}$	$Q(D_1)$	$Q(D_2)$	$Q(D_3)$	...
$customer_1$	*3.1*	*3.2*	*2.8*	
$customer_2$	*1.0*	*1.5*	*1.6*	
$customer_2$	*0.7*	*1.1*	*1.2*	

Customer identities are assumed to be a public information, as every facility consumes power. However, their load fluctuations are considered to be highly sensitive. Indeed. changes in power consumption may indirectly reveal production levels and hence strategic investments, decreases in

sales, and other similar information. These 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_u^t and r'^t_u to be two distinct values that may be reported by user u at time t and that satisfy $|r_u^t - r'^t_u| \leqslant \alpha$ for some positive real value α. An attacker should not be able to confidently determine that u reported value r_u^t instead of value r'^t_u and vice-versa.

This goal can be achieved by using a more general adjacency relation between datasets. This relation needs to capture the distance between reported quantities whose magnitudes must be protected up to some given value $\alpha > 0$. The value α is called the *indistinguishability level*. This generalized definition of differential privacy has been adopted in several applications and has sound theoretical foundations (e.g., Koufogiannis, Han, & Pappas, 2015; Andrés et al., 2013; Chatzikokolakis et al., 2013)[3]x. Consider a dataset X to which n individuals contribute their real-valued data $r_i \in \mathbb{R}$, i.e., $X = (r_1, \ldots, r_n) \in \mathbb{R}^n$. For $\alpha > 0$, an adjacency relation that captures the indistinguishability of a single individual to the aggregating scheme is:

$$X \sim_\alpha X' \;\Leftrightarrow\; \exists i \text{ s.t. } |r_i - r'_i| \leqslant \alpha \text{ and } r_j = r'_j, \forall j \neq i. \tag{5}$$

Such adjacency definition, in our target domain, is useful to hide increases or decreases of loads up to some quantity α. The generalization of differential privacy that uses the above adjacency relation is referred to as α-indistinguishability (Chatzikokolakis et al., 2013).

The w-privacy definition, introduced in Definition 2, can be extended to use α-indistinguishability by replacing the standard neighboring relation between datasets with the adjacency relation defined in Equation (5). More precisely, for a given $\alpha > 0$, two data streams prefixes $D[t], D'[t]$ are said *(w, α)-neighbors*, written $D[t] \sim_w^\alpha D'[t]$, 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 $j - i + 1 \leqslant w$ holds.

Definition 3 ((w, α)-indistinguishability) *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 under α-indistinguishability ((w, α)-indistinguishability for short) if, for all sets $O \subseteq \mathcal{O}$, all (w, α)-neighboring stream prefixes $D[t], D'[t]$, and all t:*

$$\frac{Pr[\mathcal{A}(D[t]) \in O]}{Pr[\mathcal{A}(D'[t]) \in O]} \leqslant \exp(\epsilon). \tag{6}$$

This privacy definition combines w-event privacy (Kellaris et al., 2014) and a metric-based generalization of differential privacy (Chatzikokolakis et al., 2013) in order to meet the requirements of the motivating application. Notice that (w, α)-indistinguishability is a generalization of w-privacy, and it reduces to w-privacy when $\alpha = 1$.

OPTSTREAM can be naturally extended to satisfy this definition, by calibrating the noise introduced by the Laplace mechanism, used during the sampling (Theorem 6), perturbation (Theorem 8), and post-processing (Theorem 9) steps, to satisfy the above definition. More practically, it is achieved by updating the definition of query sensitivity as:

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

3. We refer the interested reader to (Chatzikokolakis et al., 2013) for a broad analysis of indistinguishability when differential privacy uses different notions of distance.

Thus, in the (w, α)-indistinguishability model, the sensitivity of each query Q used during the *perturb* (see Section 3.2) and the *post-process* (see Section 3.4) procedures is $\Delta_Q = \alpha$, and the sensitivity Δ_L of of the queries adopted in the $L1$-based *reconstruction* procedure (see Section 3.3) is $\Delta_L = 2\alpha(\mathrm{w}-k)$. The behavior of OPTSTREAM for different indistinguishability levels is presented in Section 6.

5.2 Hierarchical Data Streams

OPTSTREAM can also be generalized 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, either $S_{a_1} \subset S_{a_2}$ or $S_{a_2} \subset S_{a_1}$ holds or $S_{a_1} \cap S_{a_2} = \varnothing$ is true.

A set of aggregation entries $\mathcal{S}$ can be represented hierarchically through a tree (called an *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 *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 whose 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 is 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 13 *For a hierarchy of height h,* OPTSTREAM*, when using a privacy loss of ϵ/h for each level of the hierarchy, satisfies ϵ-differential privacy.*

The result follows from the privacy guarantee of OPTSTREAM (Theorem 1), parallel composition of differential privacy across each level of the hierarchy (Theorem 2), and sequential composition of differential privacy, for each of the h hierarchical queries (Theorem 1).

6. Evaluation

This section evaluates OPTSTREAM on real data streams for a number of tasks. It first evaluates the accuracy of the privacy-preserving release of a stream of data within the w-privacy model. Then, it studies the accuracy of each of the four individual components of the algorithm. Finally, it evaluates the algorithm within the (w, α)-indistinguishability model.

Stream Data Regions ($\mathscr{R}$)	Daily Average (MW)	Stream Data Regions ($\mathscr{R}$)	Daily Average (MW)
Auvergne-Rhône-Alpes	7717.58	Ile de France (Paris)	8315.13
Bretagne	2554.23	Nouvelle Aquitaine	4985.68
Bourgogne - Franche-Comtè	2498.23	Normandie	3267.22
Centre - Val de Loire	2157.97	Occitanie	4314.70
Grand Est	5286.24	Pays de la Loire	3174.17
Hauts de France	5832.26	Provence - Cote d'Azur	4782.40

Table 2: 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. The number of time steps available for all regions is 17,520.

Dataset The source data was obtained through a collaboration with *Réseau de Transport d'Électricité*, the largest energy transmission system operator in Europe. It consists 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 $\boldsymbol{x}(R) = x_1(R), x_2(R), \ldots$ 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 analyze the data stream release. For the evaluation of hierarchical data streams, the experiments consider a hierarchical aggregation tree of height 2, where the root node is the national level and each of the leaves corresponds to one region. Table 2 lists an overview of the data streams derived from the real energy load consumption data for each French region in 2016. Each data stream contains 17,520 entires.

Algorithms The following sections evaluate two versions of OPTSTREAM, with equally-spaced sampling and with the $L1$-sampling. They are compared against the *Laplace mechanism* and the *Discrete Fourier Transform (DFT)* algorithm (Rastogi & Nath, 2010). All the algorithms release privacy-preserving data associated with the sub-streams $\boldsymbol{x}$ for each w-period.

To ensure ϵ-differential privacy for each w-period, the Laplace mechanism applies Laplace noise with parameter ϵ/w to each value in the period. To ensure ϵ-differential privacy, the DFT algorithm works as follows. For each w-period, and given a value $k < w$, it first computes the *Fourier coefficients* $DFT(\boldsymbol{x})_j = \sum_{i=1}^{w} \exp(\frac{2\pi\sqrt{-1}}{w} jix_i)$ of each sub-stream $\boldsymbol{x}$ and each $j \in [w]$. It then considers the k coefficients of lowest frequencies – which represent the high-level trends in $\boldsymbol{x}$ – and perturbs them using Laplace noise with parameter $\sqrt{k}\Delta_{2Q}/\epsilon$, where Δ_{2Q} is the L_2 sensitivity of the count query. It then pads the vector of noisy coefficients with a $(\mathrm{w}-k)$-dimensional vector of 0's to obtain a w-dimensional vector. Finally, it applies the *Inverse Discrete Fourier Transform* (IDFT) to this w-dimensional vector to obtain a noisy estimate of the elements in the w-period. In more details, if $\hat{f}_j$ is the j-th perturbed coefficient of the series, the noisy estimate for the value x_j is obtained through the IDFT function as: $\frac{1}{w}\sum_{i=1}^{w} \exp(\frac{2\pi\sqrt{-1}}{w} ji)\hat{f}_j$.

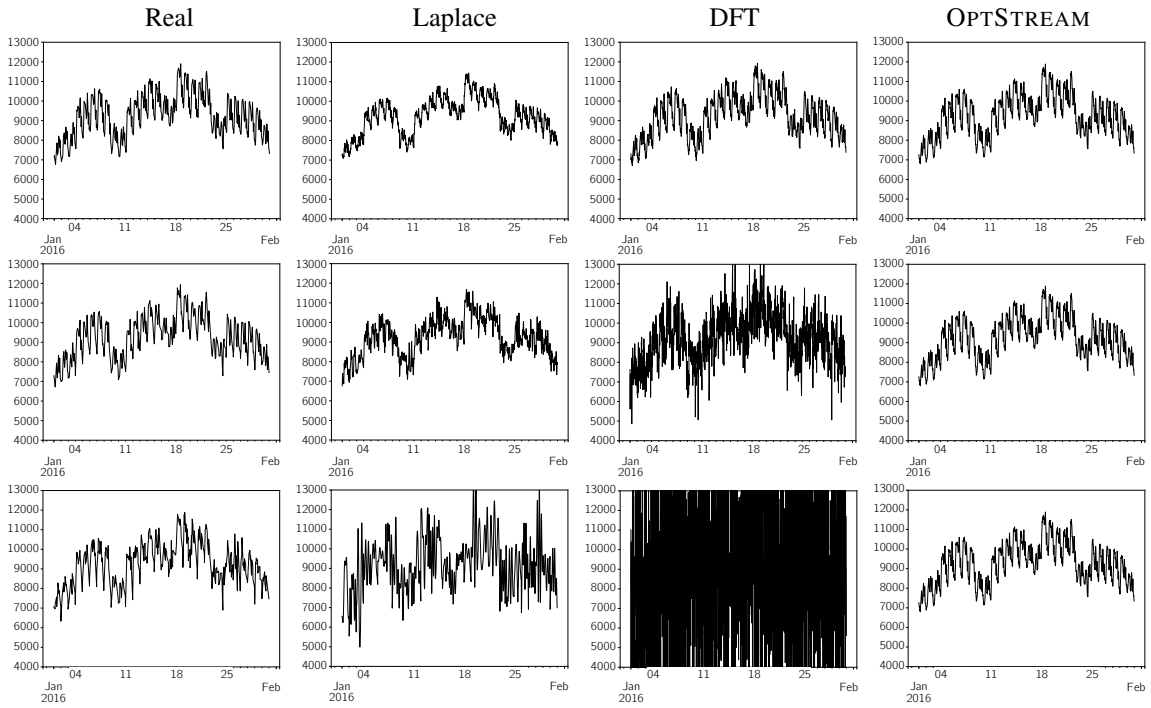


Figure 2: Real load consumption data, in MW, for the Auvergne-Rhône-Alpes region in January 2016 (first column) and its privacy-preserving versions obtained using Laplace (second column), DFT (third column), and OPTSTREAM (fourth column) with privacy loss $\epsilon = 1$ (top), $\epsilon = 0.1$ (middle), and $\epsilon = 0.01$ (bottom).

6.1 Stream Data-Release in the w-Privacy Model

This section evaluates the accuracy of privately releasing a data stream within the w-privacy model. It first studies the prediction error of the algorithms. It then analyzes the accuracy of privately releasing hierarchical streams on aggregated queries. Finally, it analyzes the accuracy of forecasting tasks from the released privacy-preserving data streams.

Answering queries over contiguous w-periods corresponds to releasing the privacy-preserving stream over the entire available duration. Figure 2 illustrates the real and privacy-preserving versions of the data-stream for the Auvergne-Rhône-Alpes region in January, 2016. It uses w-periods of size 48 for given privacy losses $\epsilon = 1.0, 0.1$, and 0.01, shown in the top, middle, and bottom rows respectively. Recall that the choice for the w-period allows the data curator to ensure the protection of the observed power consumptions within each period. Thus, the released stream protects loads in each entire day.

The real loads are illustrated in the first column. The figure compares our proposed OPTSTREAM algorithm (fourth column) against the *Laplace mechanism* (second column), and the DFT algorithm (Rastogi & Nath, 2010) (third column).

The experiments set the number of Fourier coefficients in the DFT and sampling steps in OPTSTREAM to 10. The privacy loss allocated to perform each measurement is split equally. Additionally, for OPTSTREAM $\epsilon_s = \epsilon_p = \epsilon_o = \frac{1}{3}\epsilon$, and the $L1$-sampling procedure uses a thresh-

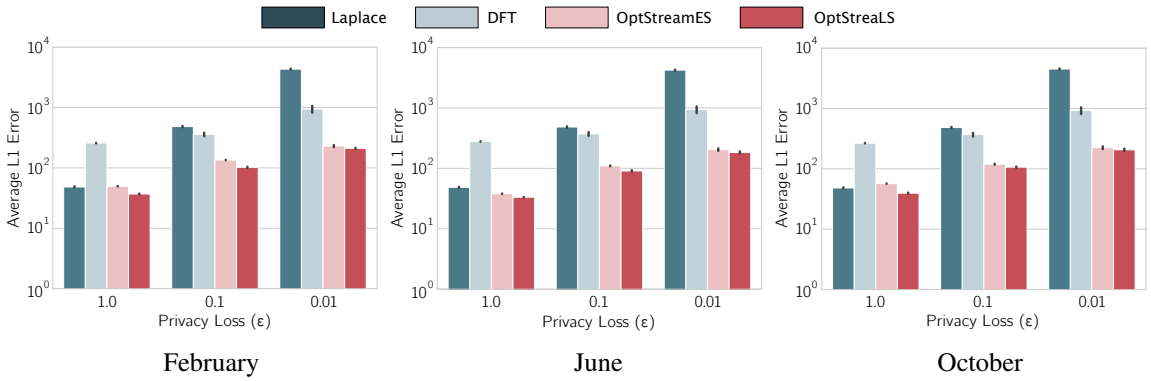


Figure 3: L_1-error analysis: Load stream data for the months of February (left), June (middle), and October (right). The y-axis reports $\log_{10}$ of the average L_1-error for all the stream data streams $R \in \mathscr{R}$.

old value θ of 1000 (which is about one tenth of the average load consumption in each region). This information was publicly revealed. 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 $[0, 14)$, $[14, 24)$, $[24, 36)$, and $[36, 48)$ 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. 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 privacy-preserving answers, the privacy is guaranteed by the post-processing immunity of DP (see Theorem 3). Finally, if an algorithm reports negative noisy value for a stream point, we truncate it to zero.

Figure 2 clearly illustrates that, for a given privacy disclosure level, OPTSTREAM produces privacy-preserving streams that are more accurate than its competitors when visualized. The next paragraph quantifies the error reported by the algorithms.

6.1.1 AVERAGE L_1-ERROR ANALYSIS

This section reports the average L_1-error for each w-period produced by the algorithms. For each input data stream $\boldsymbol{x}_R$ associated to a region $R \in \mathscr{R}$ (illustrated in Table 2) and each reported privacy-preserving stream $\hat{\boldsymbol{x}}_R$, we compare the average L_1-error defined as: $\frac{1}{N_R}\|\hat{\boldsymbol{x}}_R - \boldsymbol{x}_R\|_1$, where N_R is the length of the data stream associated with region R. Figure 3 reports the average errors across all streaming regions $r \in \mathscr{R}$ for the months of February (left), June (middle), and October (right). The consumptions in these three months capture different customers load profile behaviors due to different weather patterns and durations 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 OPTSTREAMLS) are presented and correspond to the qqually-spaced sampling, and the $L1$-sampling procedures, respectively. While all algorithms induce a notable L_1-error which increases as the privacy loss decreases, the figure highlights that OPTSTREAM consistently outperforms competitor

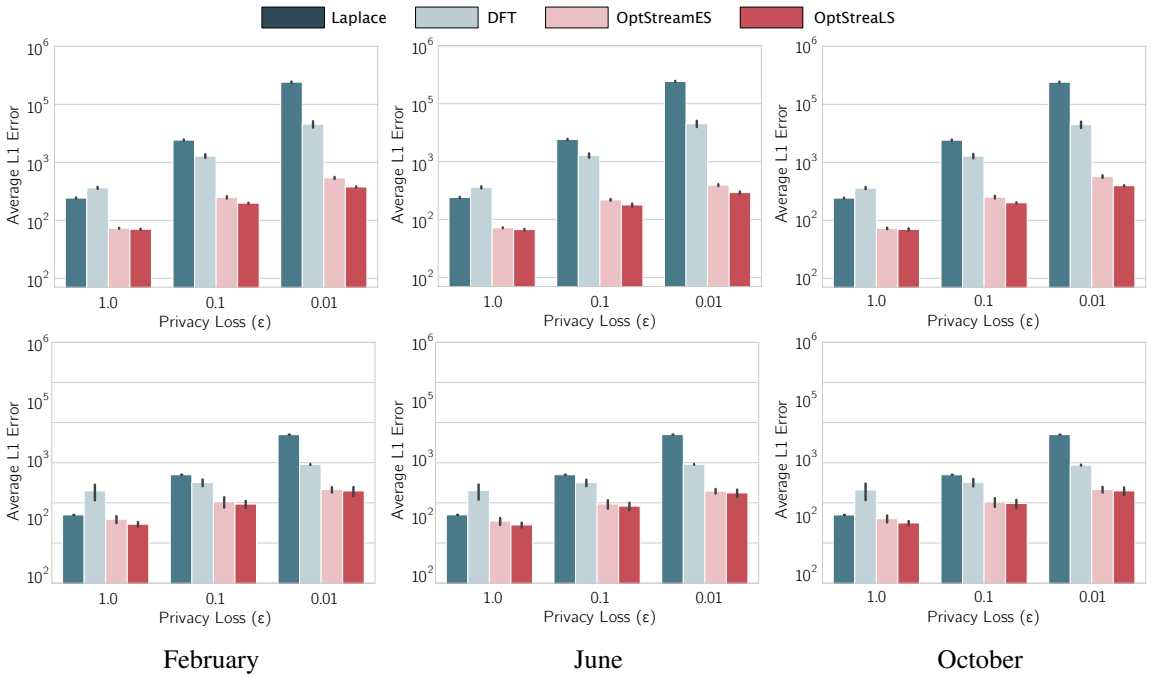


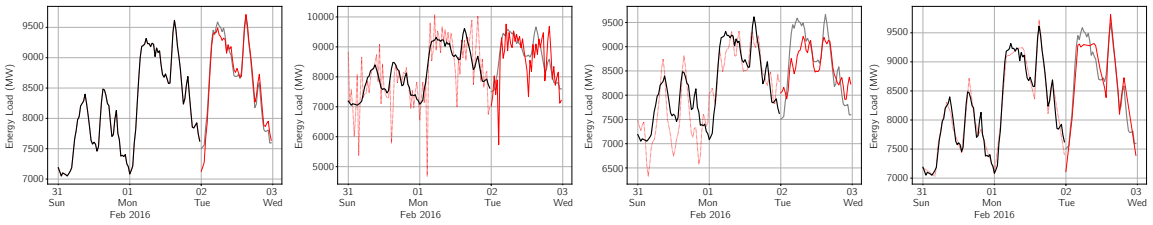
Figure 4: L_1-error analysis: Hierarchical energy load stream data for the months of February, June, and October. The y-axis reports $\log_{10}$ of the average L_1-error at national level (top) and at the regional level (bottom).

algorithms. Additionally, OPTSTREAM with $L1$-sampling is found to outperform its equally-spaced sampling counterpart, especially for large privacy losses. For small privacy losses, the two versions of the algorithm tend to perform similarly. This is due to the fact that the $L1$-score becomes less accurate as the amount of noise increases.

6.1.2 HIERARCHICAL PRIVACY-PRESERVING DATA-STREAM RELEASE

This section evaluates the extensions proposed in Section 5.2 for releasing aggregated queries over hierarchical data streams. We answer the following queries over contiguous w-periods for the whole duration of the stream: count queries over the data stream $\boldsymbol{x}(R) = x_1(R), x_2(R), \ldots$ for each region $R \in \mathscr{R}$ listed in Table 2, as well as count queries $\boldsymbol{x} = x_1, x_2, \ldots$, 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 loss of $\frac{\epsilon}{2}$ to each level of the hierarchy. OPTSTREAM is compared against the Laplace mechanism applied to each stream data, using a uniform allocation of the privacy loss ($\frac{\epsilon}{2}$) for each query in a different level of the stream hierarchy and the DFT algorithm which answers queries over each data stream by uniformly allocating a portion of the privacy loss at each level of the hierarchy.

Figure 4 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 loss $\epsilon = 1.0$. The top row of the figure reports the average L_1-errors when releasing stream data $\boldsymbol{x}(R)$ associated with each region R, while the bottom row gives the L_1-errors when releasing the stream data $\boldsymbol{x}$



Real Laplace DFT OPTSTREAM

Figure 5: Prediction error: Forecast for a one day load consumption through an ARMA model on the real load consumption data (Real) and its privacy-preserving versions obtained using Laplace, DFT, and OPTSTREAM with privacy loss $\epsilon = 0.1$.

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 $L1$-sampling produces stream data with the lowest average L_1-errors for both levels of the stream hierarchy. In addition to the improved error it is important to note that OPTSTREAM ensures the consistency of the values of the privacy-preserving stream in the hierarchy, i.e., for each time step, the reported sum of the loads across all regions equals the reported load at national level. Neither the Laplace mechanism nor DFT ensure such property.

6.1.3 IMPACT OF PRIVACY ON FORECASTING DEMAND

The final results evaluate the capability of the released privacy-preserving streams to accurately predict future consumptions. To do so, we adopt the Autoregressive Moving Average (ARMA) model (Alwan & Roberts, 1988; Zhang, 2003; Cochrane, 2005; Hipel & McLeod, 1994). ARMA is a popular stochastic time series model used for predicting future points in a time series (forecast). It combines an Autoregressive (AR) model (Alwan & Roberts, 1988) and a Moving Average (MA) model (Alwan & Roberts, 1988; Cochrane, 2005), 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 the past p 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. A 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 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 privacy-preserving data stream estimated using Laplace, DFT, and OPTSTREAM with $L1$-sampling. All models use the same parameters adopted in the previous sections.

Figure 5 visualizes the forecast for the load consumptions in the Auvergne-Rhône-Alpes region for February 2, 2016. The black and gray solid lines illustrates, respectively, the real load values observed so far and those of the day to be forecasted. The dotted red lines illustrates the privacy-preserving stream data estimated so far (and used as input to the prediction model) and the solid

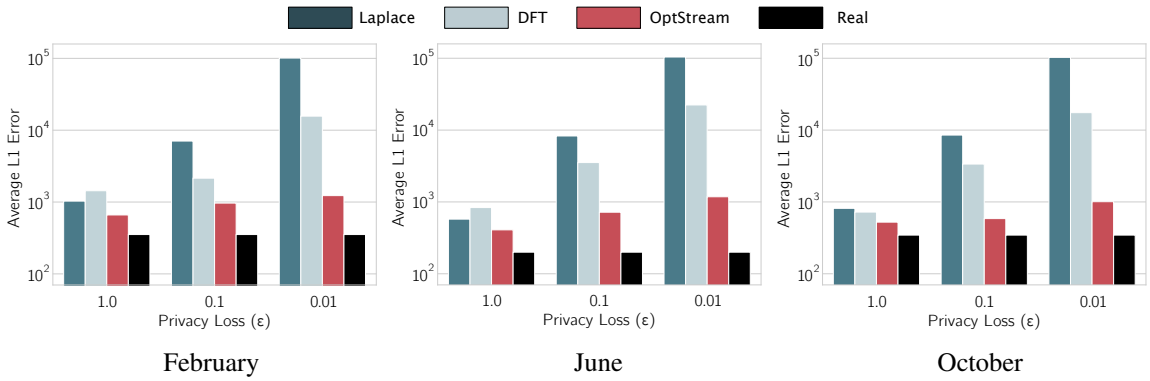


Figure 6: L_1 error analysis: ARMA forecasting model on stream data for the energy loads of the months of February, June, and October for the Auvergne-Rhône-Alpes region.

red lines depict the prediction obtained using the ARMA model. Figure 5 shows the forecast results using the real data (Real) and the privacy-preserving stream obtained through Laplace, DFT, and OPTSTREAM, respectively. The figure clearly shows that OPTSTREAM is able to visually produce better estimates for the next day forecast.

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 6, the average L_1-error for predicting each day in the month of February, June, and October 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.

6.2 Evaluation of OPTSTREAM Individual Components and Hyperparameters

We also evaluate the effect of the sampling step, the benefits of the post-processing procedure, and the hyperparameters of OPTSTREAM. Figure 7 visualizes the real (first column) and OPTSTREAM-based privacy-preserving version (other columns) load consumption data for the Auvergne-Rhône-Alpes region in January, 2016. The top row and bottom rows illustrate results for privacy losses of $\epsilon = 0.1$ and 0.01 respectively.

In order from the second to the last row, Figure 7 illustrates the results of a version of OPTSTREAM that enables only the perturbation step ($\langle P, \circ, \circ \rangle$), the perturbation and the optimization steps ($\langle P, \circ, O \rangle$), the perturbation and the sampling steps ($\langle P, S, \circ \rangle$), and all steps ($\langle P, S, O \rangle$). The privacy loss is divided equally among all active components of the algorithm. When only the perturbation step is enabled (second column), OPTSTREAM has the same behavior of the Laplace mechanism. The addition of the optimization step (third column) helps increasing the fidelity of the data stream reported (for instance, the heavy peaks are less pronounced), although the overall signal is still noisy. When the perturbation step is combined with the sampling step under the $L1$ reconstruction procedure (fourth column), the resulting privacy-preserving stream captures with much higher fidelity the salient load *peaks* and the streams looks much more similar to the real one. However,

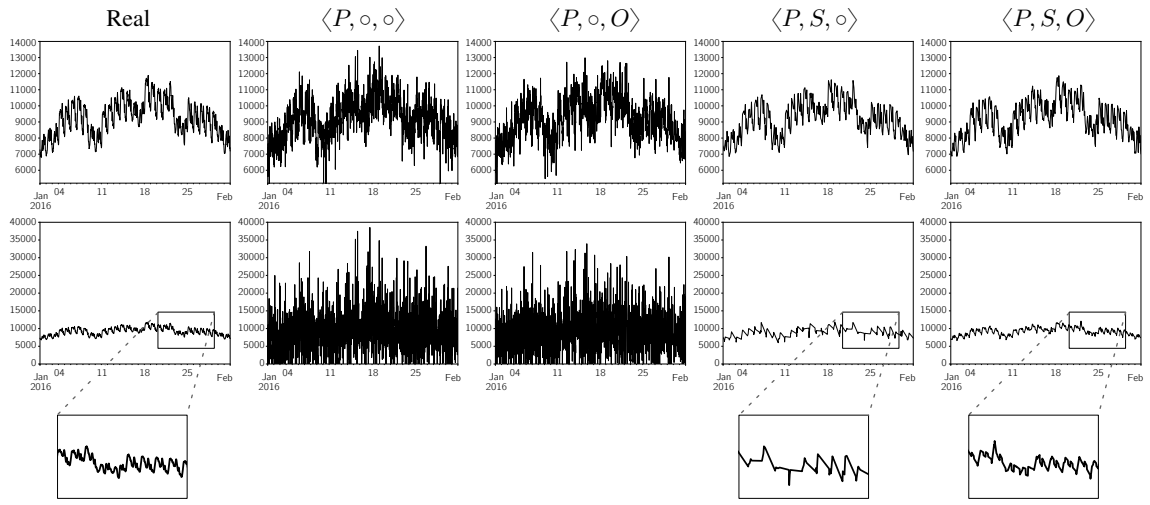


Figure 7: Real load consumption data for the Auvergne-Rhône-Alpes region in January, 2016 (Real) and OPTSTREAM activating: exclusively the perturbation step ($\langle P, \circ, \circ \rangle$), perturbation and optimization steps ($\langle P, \circ, O \rangle$), perturbation and sampling steps ($\langle P, S, \circ \rangle$), and all steps ($\langle P, S, O \rangle$). The top row, illustrates the results for privacy loss $\epsilon = 0.1$, while the bottom row uses $\epsilon = 0.01$. The boxes in the last two quadrants provide a zoom to illustrate in more details the reconstructed time series.

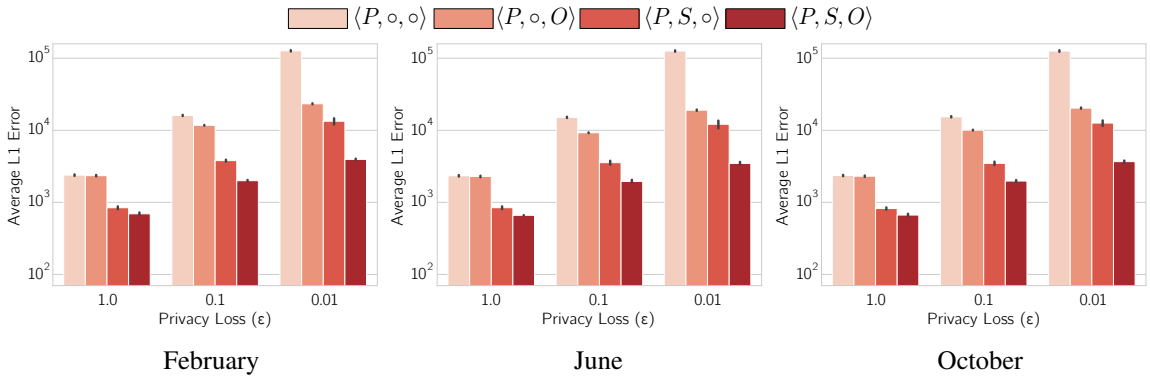


Figure 8: OptStream components analysis: Average L_1-error associated with the energy loads of the months of February, June, and October.

a careful inspection reveals several shortcomings: Several load peaks are not captured accurately, and the loads are often distributed uniformly between sampled points, making the resulting stream unrealistic. Finally, the optimization-based post-processing shines when integrated with the perturbation and sampling steps (5-th column). The salient features of the data are effectively captured and the noise redistribution provides higher resemblance to the real data stream.

Figure 8 details the L1-errors associated with the privacy-preserving energy loads for the months of February, June, and October. This quantitative analysis further highlights the importance of the

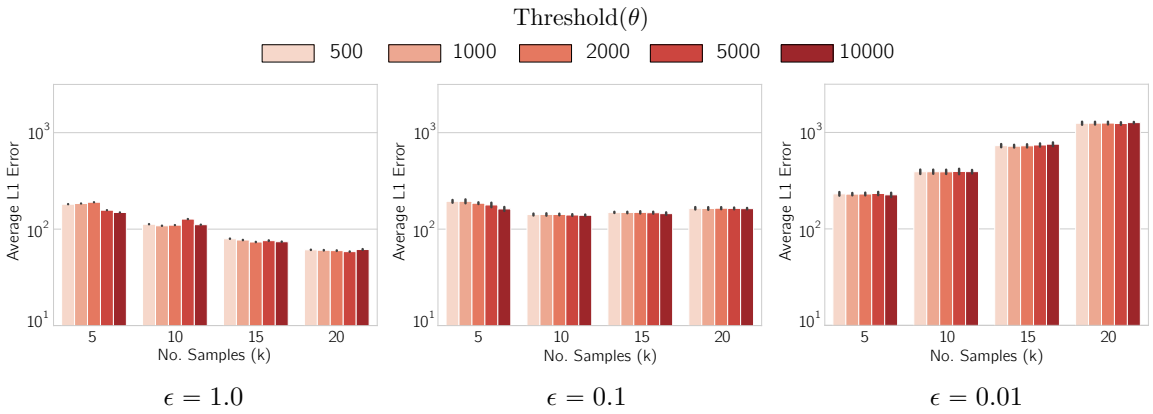


Figure 9: OptStream components analysis: Average L_1-error associated with the energy loads at varying of the threshold θ and number of samples k of Algorithm 3, for different privacy losses ϵ.

four components of the proposed OPTSTREAM algorithm. It illustrates a gradual error reduction of the framework when it adopts, in order, the perturbation step only, the perturbations and the optimization steps, the perturbation and the sampling with linear interpolation steps, and finally, all four steps.

The final results evaluate the effect of the hyperparameters k and θ adopted in the $L1$-sampling procedure of OPTSTREAM (Algorithm 3). Here, k denotes the number of samples the algorithm selects within each w-period and θ is a threshold used to adaptively select the next point to sample. These parameters are responsible for the perturbation and reconstruction errors discussed in Section 3.1. Figure 9 illustrates the effect of changing the values of θ from 500 to $10,000$, and k from 5 to 20, for different privacy losses $\epsilon \in \{1.0, 0.1, 0.01\}$. The figure shows the average $L1$-errors associated with the year-long data streams of each region $r \in \mathscr{R}$.

For a given pair of ϵ and k values, the effect of modifying the threshold θ results in negligible error differences, except when the number of samples k is very small (e.g., 5). In such a case, larger thresholds θ result in more precise reconstructions. This is because larger thresholds allow the sampling procedure to subsample within the whole w-period, while small θ values force the subsamples to be drawn in smaller portions of the w-period. The results also show that, for different privacy levels, the number of samples has a large effect on the produced error. In particular, for large privacy losses ($\epsilon = 1.0$), the error reduces as the number of samples increase while, for small privacy losses ($\epsilon = 0.01$), the error increases as the number of samples increase. For moderate privacy losses ($\epsilon = 0.1$) the effect of changing k is less noticeable. Therefore, the best number of sampling steps depends on the privacy loss adopted. This behavior illustrates the dichotomy between the reconstruction and the sampling errors and highlights the accuracy results of Theorem 11.

The evaluation considers the average daily demanded by each region to be publicly available. When this is not the case, one can allocate a very small portion of the overall privacy budget to estimate such a quantity. Hence, one can choose a threshold value θ that is approximately proportional to the average daily consumption divided by the number of samples k.

6.3 Stream Data-Release in the (w, α)-Indistinguishability Model

Having analyzed the algorithms in the w-privacy setting, this section illustrates the results of the privacy-preserving data streams released under the combined w-event and α-indistinguishability models (see Section 5.1). The motivating application requires releasing the privacy-preserving data streams without disclosing chosen amounts of load changes within a time window. We thus analyze the results of privately releasing a stream of data under the (w, α)-indistinguishability model and use an adapted version of OPTSTREAM, the Laplace mechanism, and the DFT algorithm. These versions recalibrate the noise required by the mechanisms to satisfy the (w, α)-indistinguishability definition as detailed in Section 5.1.

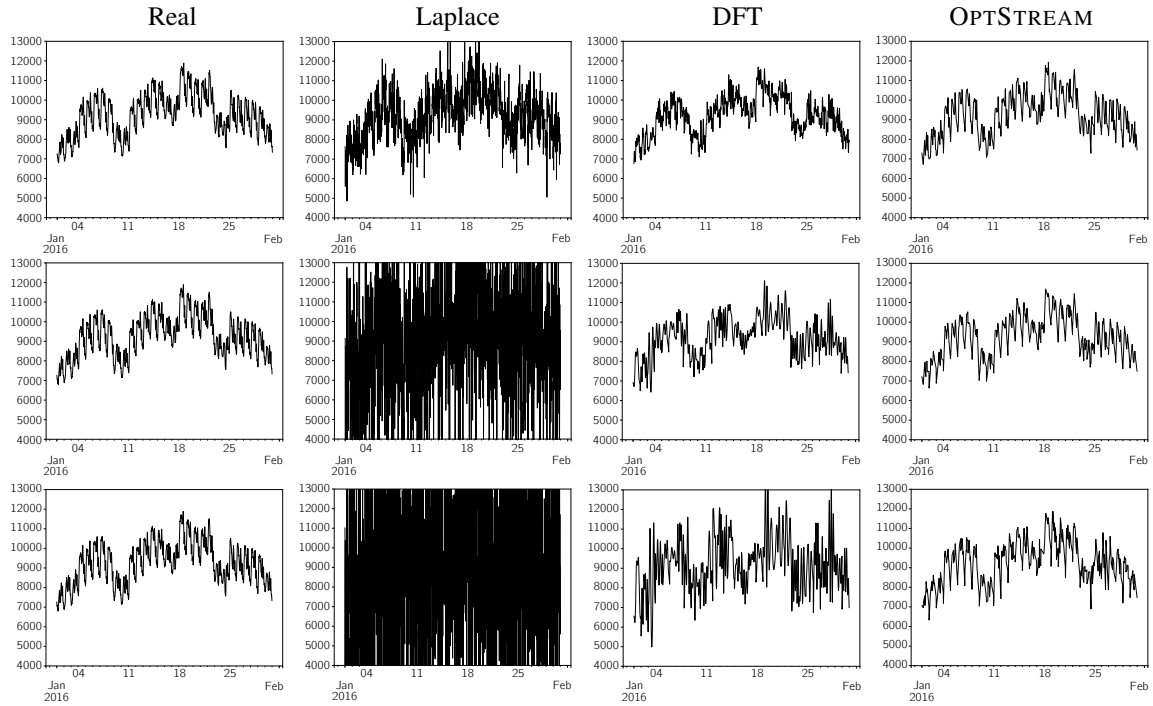


Figure 10: Real load consumption data for the Auvergne-Rhône-Alpes region in January 2016, and its privacy-preserving versions obtained using Laplace, DFT, and OPTSTREAM with privacy loss $\epsilon = 1$ and $\alpha = 10$ (top), 50 (middle), and 100 (bottom).

Figure 10 illustrates the real and privacy-preserving versions of the data stream for w-periods of size 48 (i.e., one day) given privacy loss of $\epsilon = 1$ and indistinguishability parameter $\alpha = 10$ (top row), $\alpha = 50$ (middle row) and $\alpha = 100$ (bottom row). 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 w-period specifies the length of the obfuscation period. The figure compares our proposed OPTSTREAM algorithm against the Laplace mechanism and DFT .

For DFT and OPTSTREAM, the experiments set the number of Fourier coefficients and sampling steps to 10 (when $\alpha \leqslant 50$) and 5 (when $\alpha = 100$). As in the previous experiments, the privacy loss allocated to perform each measurement is divided equally. Finally, OPTSTREAM uses the same

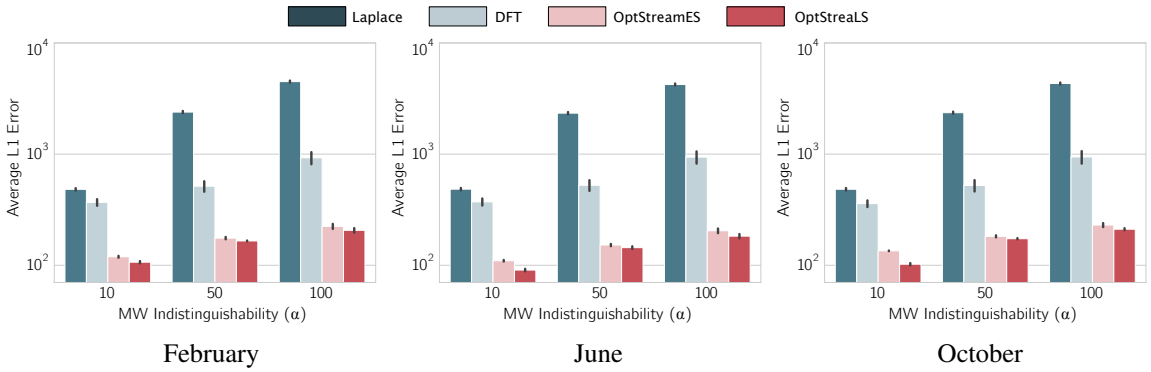


Figure 11: L_1-error analysis: Energy load stream data for the months of February, June, and October. The y-axis reports $\log_{10}$ errors averaged across the L_1 errors for each stream data $R \in \mathscr{R}$.

feature query set $\mathscr{F}$ used in the previous experiments. Figure 10 clearly illustrates that OPTSTREAM produces privacy-preserving streams that are more accurate than its competitors when visualized. This is especially evident for high indistinguishability parameters, when the Laplace mechanism can barely preserve any signal from the original data.

We now report the L_1-errors averaged for each w-period produced by the algorithms. Figure 11 reports the average results for each streaming region for the months of February, June, and October, at varying of the indistinguishability parameter $\alpha = \{10, 50, 100\}$ given a privacy loss $\epsilon = 1.0$. Each histogram reports the $\log_{10}$ value of the average error of 30 random trials. While all algorithms induce an L_1-error which increases as the indistinguishability level α increases, the figure highlights that OPTSTREAM consistently outperforms the other algorithms.

7. Discussion

One of the core advantages of the w-privacy model is that it provides flexibility regarding the size of the time frame to protect. OPTSTREAM exploits this flexibility to select a small set of representative points to sample and performs additional optimization over such values to redistribute noise and enhance accuracy. Practically, this means that the data stream needs to be batched in periods of w times steps, each of which is privately released at once.

Recall that the privacy model adopted in this paper combines the w-privacy model (Kellaris et al., 2014) with the definition of α-indistinguishability (Chatzikokolakis et al., 2013). The resulting (w, α)-indistinguishability model allow us to protect events related to quantities which are not directly related to user identities but can be used as a proxy to disclose a *secret* (e.g., a sensitive information) about a user. For instance, in our application domain, it is important to both protect the given stream from inferences about load increases or decreases–which are typically related to sensitive user activities–while also protecting the stream in an entire day.

While setting $w = 1$ would allow to use OPTSTREAM to release the stream data online – without requiring batching –, this setting offers no room for OPTSTREAM to improve accuracy, since it would be similar to the Laplace mechanism. To the best of our knowledge, the only available algorithm that can release a privacy-preserving data stream online is *PeGaSus* (Chen et al., 2017). It

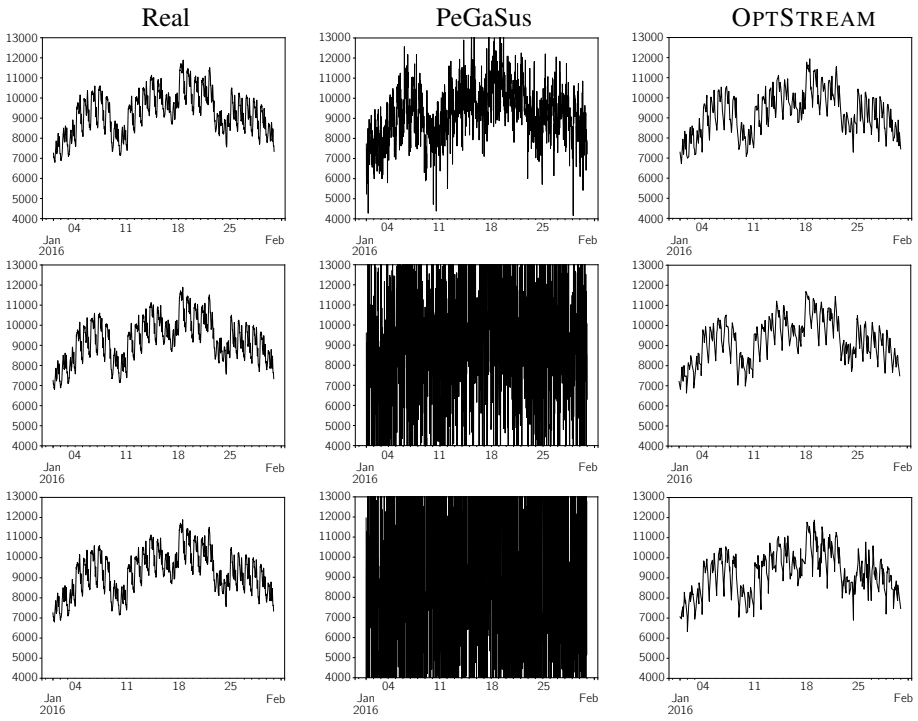


Figure 12: Real load consumption data for the Auvergne-Rhône-Alpes region in January 2016 (left), and its privacy-preserving versions obtained using PeGaSus (center), and OPTSTREAM (right) with privacy loss $\epsilon = 1$ (top-row), $\epsilon = 0.1$ (middle-row), and $\epsilon = 0.01$ (bottom-row).

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. PeGaSus, was not introduced in the w-event model as it focuses on protecting data in a single time-step.

Despite the fact that the two algorithms were designed for solving different problems, we adapted PeGaSus to the w-privacy model and to satisfy the (w, α)-indistinguishability and compare it against OPTSTREAM for completeness. Figure 12 illustrates a comparison of the algorithms on privately releasing load streams for the Auvergne-Rhône-Alpes region in January 2016. The real loads are highlighted in the first column, while their privacy-preserving counterparts are shown in the second and third columns. The figure compares our proposed OPTSTREAM (third column) against PeGaSus (second column) for indistinguishability parameters $\alpha = 10$ (top rows) $\alpha = 50$ (middle row) and $\alpha = 100$ (bottom row), and fixed a privacy loss $\epsilon = 1$. For PeGaSus, the privacy loss 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 (Chen et al., 2017). The settings for OPTSTREAM are the same as in the previous sections. Figure 12 clearly illustrates that OPTSTREAM produces privacy-preserving streams that are more accurate than those produced by PeGaSus, when the (w, α)-indistinguishability model is adopted. It is an interesting research avenue to combine PeGaSus and OPTSTREAM.

8. Related Work

Continuous release of aggregated real-time data has been studied in previous work including by Dwork (2010), Dwork et al. (2010). Most of the state-of-the-art either focuses on event-level privacy on infinite streams (Rastogi & Nath, 2010) or on user-level privacy on finite streams (Dwork et al., 2010). Dwork proposed an adaptation of differential privacy to a continuous observation setting (Dwork et al., 2010). 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, Muthukrishnan, Nikolov, and Wright (2011) proposed *pan-privacy* for estimating 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. (2014) 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 applies to the last model.

Within the differential privacy proposal for data streams that fit such model, Wang, Zhang, Lu, Wang, Qin, and Ren (2016) proposed *Rescue DP*, which is designed explicitly for spatiotemporal traces. PeGaSus (Chen et al., 2017) is another seminal work that allows to protect 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 privacy-preserving 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 privacy-preserving data-stream release and has been showed to be effective in several settings.

Rastogi and Nath (2010) 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 (Fan & Xiong, 2014). We also compare OPTSTREAM against DFT.

Fan, Xiong, and Sunderam (2013) proposed Fast, an adaptive algorithm for privacy-preserving 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 (Rastogi & Nath, 2010) 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 one order of magnitude when compared to DFT.

Finally, Chan et al. (2011) focused on releasing prefix-sums of the streaming data counts while adopting an event-based privacy model. Bolot et al. (2013) 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.

9. Conclusions

This paper presented OPTSTREAM, a novel algorithm for privately releasing stream data in the w-event privacy model. 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 privacy-preserving output of the previous modules, along with the privacy-preserving answers of additional queries on the data stream, to ensure consistency of quantities associated with salient features of the data. OPTSTREAM was evaluated on a real dataset from the largest transmission operator in Europe. Experimental results on multiple test cases show that OPTSTREAM improves the accuracy of the state-of-the-art by at least one order of magnitude in 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 privacy-preserving data sub-streams. 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. An important direction of future work is to generalize these results to the streaming setting where a data element is emitted at each time step. Another direction is to develop models to privately learn temporal patterns that can further improve the data release of streams in certain domains. Future work will also focus on ensuring that salient properties of an optimization problem of interest hold, when the problem relies on inputs that include the load consumption data, e.g., as in Fioretto and Van Hentenryck (2018), Fioretto, Mak, and Van Hentenryck (2019). Finally, it is interesting to integrate recent studies on the correlation of the series over time (e.g., see Liu, Chakraborty, & Mittal, 2016) in order to take into account their effect on the privacy loss.

Acknowledgments

The authors are thankful to Kory Hedman for extensive discussions and to the anonymous reviewers for their valuable comments and suggestions to improve the quality of the paper. This research is partly funded by the ARPA-E Grid Data Program under Grant 1357- 1530.

References

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

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

Bolot, J., Fawaz, N., Muthukrishnan, S., Nikolov, A., & Taft, N. (2013). Private decayed predicate sums on streams. In *Proceedings of the 16th International Conference on Database Theory*, pp. 284–295. ACM.

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

Chatzikokolakis, K., Andrés, M. E., Bordenabe, N. E., & Palamidessi, C. (2013). Broadening the scope of differential privacy using metrics. In *International Symposium on Privacy Enhancing Technologies Symposium*, pp. 82–102. Springer.

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

Cochrane, J. H. (2005). Time series for macroeconomics and finance. *Manuscript, University of Chicago*.

Ding, B., Kulkarni, J., & Yekhanin, S. (2017). Collecting telemetry data privately. In *Advances in Neural Information Processing Systems*, pp. 3574–3583.

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

Dwork, C., McSherry, F., Nissim, K., & Smith, A. (2006). Calibrating noise to sensitivity in private data analysis. In *TCC*, Vol. 3876, pp. 265–284. Springer.

Dwork, C., Naor, M., Pitassi, T., & Rothblum, G. N. (2010). Differential privacy under continual observation. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pp. 715–724. ACM.

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

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

Fan, L., Xiong, L., & Sunderam, V. (2013). 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*, pp. 1065–1068. ACM.

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

Fawaz, K., & Shin, K. G. (2014). Location privacy protection for smartphone users. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pp. 239–250. ACM.

Fioretto, F., Lee, C., & Van Hentenryck, P. (2018). Constrained-based differential privacy for mobility services. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS*, pp. 1405–1413.

Fioretto, F., Mak, T. W. K., & Van Hentenryck, P. (2019). Differential privacy for power grid obfuscation. *CoRR*, *abs/1901.06949*.

Fioretto, F., & Van Hentenryck, P. (2018). Constrained-based differential privacy: Releasing optimal power flow benchmarks privately - releasing optimal power flow benchmarks privately. In *Proceedings of the Integration of Constraint Programming, Artificial Intelligence, and Operations Research (CPAIOR)*, pp. 215–231.

Hardt, M., & Rothblum, G. N. (2010). A multiplicative weights mechanism for privacy-preserving data analysis. In *Foundations of Computer Science*. IEEE.

Hipel, K. W., & McLeod, A. I. (1994). *Time series modelling of water resources and environmental systems*, Vol. 45. Elsevier.

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

Koufogiannis, F., Han, S., & Pappas, G. J. (2015). Optimality of the laplace mechanism in differential privacy. *arXiv preprint arXiv:1504.00065*.

Liu, C., Chakraborty, S., & Mittal, P. (2016). Dependence makes you vulnberable: Differential privacy under dependent tuples.. In *NDSS*, Vol. 16, pp. 21–24.

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

Nemirovski, A. (2004). Interior point polynomial time methods in convex programming. Tech. rep. ISYE 8813, Georgia Institute of Technology, Atlanta, GA.

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

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

Rastogi, V., & Nath, S. (2010). Differentially private aggregation of distributed time-series with transformation and encryption. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pp. 735–746. ACM.

Wang, Q., Zhang, Y., Lu, X., Wang, Z., Qin, Z., & Ren, K. (2016). 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*, pp. 1–9. IEEE.

Zhang, G. P. (2003). Time series forecasting using a hybrid arima and neural network model. *Neurocomputing*, *50*, 159–175.