Privacy-Preserving Distributed Stream Monitoring (NDSS 2014)

Arik Friedman
NICTA, Australia
arik.friedman@nicta.com.au

Izchak Sharfman
Technion, Israel
tsachis@cs.technion.ac.il

Daniel Keren
Haifa University, Israel
dkeren@cs.haifa.ac.il

Assaf Schuster
Technion, Israel
assaf@cs.technion.ac.il

***Abstract*—Applications such as sensor network monitoring, distributed intrusion detection, and real-time analysis of financial data necessitate the processing of distributed data streams on the fly. While efficient data processing algorithms enable such applications, they require access to large amounts of often personal information, and could consequently create privacy risks. Previous works have studied how privacy risks could be mitigated through the application of differential privacy to continuous stream monitoring, focusing mostly on evaluating simple aggregates over the streams, such as counts and sums. However, many real world applications require monitoring a complex value derived from the streams, e.g., detecting that the correlation between the values of two stocks traded in different exchanges has crossed a threshold.**

In this paper we present a general framework that enables monitoring *arbitrary* functions over statistics derived from distributed data streams in a privacy-preserving manner. Our solution allows the monitoring of complex values derived from the streams, while preventing adversaries from learning about any particular element in the processed streams. We study the relationship between communication efficiency and privacy loss, and demonstrate that for given privacy constraints, our approach allows the system to be monitored over periods that are three orders of magnitude longer than would be possible with a naive approach. To the best of our knowledge, this work is the first to tackle privacy-preserving distributed monitoring of *arbitrary* functions, including non-linear functions, and to evaluate empirically the applicability of privacy-preserving stream monitoring in such settings.

I. Introduction

Distributed evaluation of functions is a fundamental problem in distributed computation, and monitoring queries constitute a significant portion of the tasks carried over distributed streams. In some cases, these queries can be as simple as monitoring the sum of a distributed set of variables against a predetermined threshold, or identifying frequent itemsets in a set of distributed streams. In other cases, the queries require more complicated computations, as in the case of non-linear scoring functions (e.g., information gain or χ^2) for the purpose of feature selection, or monitoring the sum of square errors with respect to some baseline to identify anomalous behavior.

NDSS '14, 23-26 February 2014, San Diego, CA, USA
Copyright 2014 Internet Society, ISBN 1-891562-35-5
http://dx.doi.org/doi-info-to-be-provided-later

While monitoring algorithms enable or improve applications such as fraud detection, early detection of disease outbreaks, and fast reaction to security-related incidents, they require access to large amounts of often personal information. As the collection of such information becomes easier and cheaper, there is growing awareness of the associated privacy risks. For example, analysis of the privacy implications of collaborative recommender systems [1] showed that even aggregative algorithms that process large amounts of information could leak sensitive information about particular individuals. Such works demonstrate the importance of incorporating formal and provable privacy guarantees into the design of algorithms.

The differential privacy framework [2], which we rely on in this paper, has been proposed to prevent an adversary from inferring private information from the output of a computation. Differential privacy requires that the probability distribution of the results of the computation be only marginally affected by each input record. In differential privacy, each information exchange that is derived from data on individuals incurs a cost in privacy. With any new information exchange, the cost accumulates. To restrict privacy leaks, information exchange should be stopped whenever the accumulated cost grows beyond a pre-determined bound (a *privacy budget*). Theoretical infeasibility results suggest that these constraints are inherent to privacy-preserving information processing [3]. However, the lifetime of a stream monitoring system can be greatly extended, without violating the privacy constraints. Moreover, even for systems in which utility trumps privacy (where the system should keep operating regardless of any privacy breaches), privacy risks and potential harm could be reduced by using algorithms that embed privacy protection in the monitoring process in a cost-effective manner.

With the goal of efficient use of the privacy budget in mind, much effort has been devoted to the application of differential privacy to aggregation over centralized or distributed data streams [3]–[8]. Those studies focused mostly on simple aggregates, such as counts and sums, or showed how specialized monitoring tasks, such as the heavy hitters problem, could be carried out in a privacy-preserving manner. However, many real world applications [9]–[13] require monitoring a complex value derived from the streams, e.g., detecting that the correlation between the values of two stocks traded in different exchanges has crossed a threshold.

Recently, Dwork et al. [3] studied continuous monitoring of a monotonically increasing or decreasing bounded range value derived from a stream (e.g., a counter) in a privacy-preserving manner. The proposed solution assumes that the monitored value changes by an amount of at least d at most k times, within a predetermined monitoring period T, and exploits this assumption to update the output only after the value changes "enough". Consequently, the privacy cost is incurred only due to update rounds, in contrast to a naive approach, which incurs a fixed privacy cost for each item received on the stream.

However, in practice, many real-world application require monitoring complex functions over statistics derived from streams. The values of these functions may not necessarily be monotonic, nor behave according to predetermined constraints. In addition, the assumption of a predetermined time period T for monitoring limits the ability to take advantage of circumstances that allow monitoring the system for longer periods of time (e.g., when the value of the function does not change much over a long period of time). Finally, many applications of interest are inherently distributed, and the monitoring requirements at each of the nodes may depend on the state of other nodes in the system.

In this work we study the problem of monitoring *arbitrary* threshold functions over statistics derived from distributed streams in a privacy-preserving manner. In this setup, data arrives at fixed time intervals, referred to as rounds, where at each round a new data item is received at each node. In addition, each node derives a vector of statistics from its local data stream. The goal is to determine when the value of an arbitrary scoring function, applied to the average of these vectors, exceeds a predetermined threshold.

We address the challenge of monitoring complex values in a privacy-preserving manner by employing communication minimization techniques to transform the monitored global condition into local constraints that can be monitored independently by each node in the system. These constraints are expressed in the form of *Safe Zones*, which are subsets of the input space. The safe zones are constructed such that as long as the local vectors are in their respective safe zones, the global condition is maintained. This reduces node synchronization, resulting in many silent (communication-free) rounds. We then leverage the reduction in communication costs towards fewer privacy leaks by applying privacy protection to a series of silent rounds simultaneously, thereby improving the privacy-accuracy trade-off provided by the system. Effectively, this protection is obtained by introducing noise to the safe zones. Our work makes the following contributions:

- We present a framework for privacy-preserving monitoring of general (possibly non-linear) functions over statistics derived from a set of distributed streams and conduct a theoretical analysis of the privacy and accuracy guarantees provided within this framework.
- We conduct an experimental evaluation of the proposed framework. We demonstrate that for given privacy constraints, our approach allows the system to be monitored over periods that are three orders of magnitude longer than would be possible with a naive approach, while maintaining remarkable accuracy.
- We discuss and evaluate the different privacy-accuracy trade-offs involved when monitoring distributed data streams, and highlight additional possible improvements of the proposed scheme.

To the best of our knowledge, our study is the first to tackle privacy-preserving distributed monitoring of *arbitrary* functions, and to evaluate empirically the applicability of privacy-preserving stream monitoring in such settings.

The paper is organized as follows. Section II discusses related work. Section III presents the problem statement and goals. Section IV follows with background on tools used in our solution. Section V presents our algorithm for privacy-preserving distributed stream monitoring, and Section VI describes some of our experimental results and performance analysis. Section VII concludes the work.

II. Related Work

Communication-efficient monitoring of distributed streams has been the subject of much research in recent years. Some research has focused on anomaly detection [9], [14], while other studies focused on monitoring specific types of functions, including sums [15], [16], Boolean predicates [10], inner products [11] and entropy [12]. Our work employs techniques presented in the context of geometric monitoring [13]. These techniques enable monitoring arbitrary threshold functions by interpreting the monitoring task as a geometric problem.

The practical implications of differentially private analysis were studied in many application domains, including network trace analysis [17], health data [18], intelligent transportation systems [19] and collaborative security mechanisms [20]. Monitoring of distributed data streams is an important scenario in many of these domains. The application of differential privacy to data stream processing was studied initially in [4], which introduced the concept of *pan-private* data stream algorithms – algorithms that retain their privacy properties even when intrusions expose the internal state of the system. Two independent works [3], [5] studied continuous release of differentially-private counts, optionally while ensuring pan-privacy. While we do not aim to obtain pan-privacy, our framework could be extended to support it through straightforward application of the technique of [3]. Dwork et al. [3] showed also how these techniques could be used to convert any single-output differentially-private algorithm to a T-round continual output algorithm, provided that the evaluated function is monotonically increasing or decreasing, or "close to" monotonic. Mir et al. [6] relied on sketches to track statistics such as distinct count, cropped first moment, and heavy hitters count over fully dynamic data while preserving pan-privacy. Fan and Xiong [7] addressed the dynamic nature of the data by adaptive sampling of the time-series data and use of Kalman filters for estimating the data in non-sampling points.

Early works on differential privacy in a distributed setting [21], [22] studied how differential privacy could be combined with cryptographic protocols to allow one-off computations to be carried out both securely and privately. Chen et al. [23] used the noise generation mechanism of [21] to allow analysts to pose histogram queries to a subset of distributed clients with the help of an honest but curious proxy. Rather than halting the system when the privacy budget is exhausted, the proxy merely tracks the overall consumed budget. Hsu et al. [8] proposed efficient differentially-private algorithms for solving the heavy hitters problem in the fully distributed local model, in which each peer has a single element.

Several works studied differentially private aggregation over distributed time-series data, focusing mostly on simple aggregates, such as counts and sums. Rastogi and Nath [24] relied on the Discrete Fourier Transform to compress historical time-series data, and leveraged threshold homomorphic encryption to run a distributed version of the Laplace mechanism on the compressed data. As the compression requires access to all the query results in advance, this method is not adequate for processing data streams on the fly. Shi et al. [25] applied cryptographic techniques to allow an untrusted aggregator to compute differentially-private sums over distributed peers without learning anything but the outcome. While the proposed scheme was designed to reduce the overhead of cryptographic operations in periodical communications, it did not address the cumulative privacy loss. Chan et al. [26] considered the problem of private tracking of heavy hitters over a sliding window, where each node maintains a small number of differentially-private counters for the most frequent items and notifies the aggregator when the approximate counts change significantly.

III. Scenario and Goals

As a motivating example for our work, consider the following scenario: in order to improve their spam filtering capabilities, several e-mail service providers have agreed to report aggregated data about the patterns of spam messages they receive to an external spam filtering service. The spam filtering service is interested in monitoring a fixed list of terms (keywords) to determine how well they separate spam messages from benign messages. More technically, the spam filtering service would like to determine when the information gain score of a given term crosses a predetermined threshold. Information gain scores are between 0 and 1, where a score of 1 is received when the term perfectly separates spam messages from non-spam messages, i.e., all the spam messages contain the term and all the non-spam messages do not contain it, or vice versa. A score of 0 is received when the term is completely useless in separating spam from non-spam, i.e., the presence of the term is equally probable in both spam and non-spam messages. The information gain score is a function of the fraction of mail messages that contain the term for each category (spam or benign), and the fraction of messages that do not contain it for each category.

In general, we consider a system consisting of k nodes, $n_1, \ldots, n_k$. We assume that data arrives at fixed time intervals, referred to as rounds, where at each round a new data item (in our example, an e-mail message) is received at each node. Specifically, each node n_i processes a stream of elements $S_i = \{q_1, q_2, \ldots\}$ from some domain $\mathcal{D}$. In each round t, the node n_i can access the local stream prefix $S_i(t)$ seen so far (or a subset of it within a sliding window) and process the data to derive a vector $\vec{v}_i(t) \in \mathbb{R}^d$. We refer to these vectors as *local statistics vectors*. The *global vector* is then given by $\vec{v}_g(t) = \sum_i \vec{v}_i(t)/k$.[1] Our goal is to identify when $f(\vec{v}_g) > T$ for some predetermined function $f : \mathbb{R}^d \to \mathbb{R}$ and threshold $T \in \mathbb{R}$. When this condition is met, we say that a *global breach* of the threshold has occurred. In many important practical applications, a distributed monitoring problem can be expressed as the monitoring of a general function evaluated at the average vector, either directly [27], [28], or after augmenting it by various functions of the raw data [29].

We assume the existence of a coordinator, with whom the nodes communicate so that they do not need to communicate with each other. The coordinator can either be a central entity separate from the nodes (e.g., the spam filtering service in our motivating example), or one of the nodes could act as coordinator.

Example 3.1: Denote the k e-mail service providers as n_1,$n_2, \ldots, n_k$, and consider a monitored term t. Each provider n_i monitors a window consisting of the last w e-mail messages processed by n_i. Let W^i be the set of e-mail messages in the monitoring window of n_i, with W^i_{spam} denoting the subset of W^i comprising spam and W^i_{benign} denoting the subset comprising benign messages. We can similarly denote by W^i_t ($W^i_{\neg t}$) the messages that contain (do not contain) the term t. The local contingency table for the term t can then be defined by the four terms: $c^i_{1,1} = |W^i_{\text{spam}} \cap W^i_t|/w$, $c^i_{1,2} = |W^i_{\text{benign}} \cap W^i_t|/w$, $c^i_{2,1} = |W^i_{\text{spam}} \cap W^i_{\neg t}|/w$ and $c^i_{2,2} = |W^i_{\text{benign}} \cap W^i_{\neg t}|/w$. The vector $\vec{v}_i = [c^i_{1,1}; c^i_{1,2}; c^i_{2,1}; c^i_{2,2}]$, derived by processing the e-mail messages in the monitoring window of n_i, forms the *local statistics vector* of the node. The *global vector* $\vec{v}_g = \sum_i \vec{v}_i/k$ then provides the corresponding contingency table over the whole set of messages $W = \bigcup W^i$, where each element in the global vector is given by $c_{\alpha,\beta} = \sum_i c^i_{\alpha,\beta}/k$. The (global) information gain of the term t is then provided by

$$IG(t,W) = f(\vec{v}_g) = f([c_{1,1}; c_{1,2}; c_{2,1}; c_{2,2}]) = \qquad (1)$$
$$= \sum_{\substack{\alpha \in \{1,2\} \\ \beta \in \{1,2\}}} c_{\alpha,\beta} \cdot \log \frac{c_{\alpha,\beta}}{(c_{\alpha,1} + c_{\alpha,2}) \cdot (c_{1,\beta} + c_{2,\beta})} \ .$$

The goal is to detect when the information gain of a term crosses a given threshold T, indicating a new term that has been targeted by spammers.

Due to the dynamic nature of spam patterns, it is crucial that the spam filtering service detect on the fly that a term has come into or dropped out of use (i.e., its information gain score crossed the threshold). A naive approach for meeting this requirement is for each provider to notify the filtering service each time a new e-mail message is received. However, as discussed in the next section, this approach increases the privacy risks.

A. Threat Model and Privacy Goals

In this work we assume that each participating node is secure (e.g., each service provider's database is protected against intrusions), and all the communication channels are encrypted and authenticated. The public keys that are used to secure the communication channels are correct, and the corresponding private keys are secure. However, the communication channels may be monitored by an adversary, who may have arbitrary background knowledge. We do not assume secrecy of the distributed monitoring protocol (no security by obscurity), so the adversary might exploit traffic patterns to learn about the

[1] Our work can also be easily extended to weighted mean vectors; see, e.g., the discussion in [13].

decisions made by the protocol, and consequently about the inputs that affected them.

We rely on *differential privacy* [2] to formulate the desired privacy properties. Intuitively speaking, given a randomized algorithm Alg, we can say that differential privacy requires limiting the effect that each record in the input can have on the output distribution of the computation. In fact, we rely in this work on a variant of this concept, presented in [3] to account for privacy in continuous processing of streams. In this variant of differential privacy, replacing one element in the stream with another should have only marginal effect on the output stream, and the node should produce any possible output sequence with almost the same probability when any particular element changes. Formally:

Definition 3.1 (Adjacent Streams [3]): We say that two stream prefixes S and S' are *adjacent*, and write $S \approx S'$, if there exist $q, q' \in \mathcal{D}$ such that replacing an occurrence of q in S with q' will result in S'.

For example, replacing one e-mail message with another in a stream of processed messages, would result in an adjacent stream.

Definition 3.2 (Differential Privacy [3]): A randomized stream processing algorithm Alg provides ϵ-differential privacy if for any adjacent stream prefixes S and S' and any set of possible output sequences O,

$$Pr[Alg(S) \in O] \leq Pr[Alg(S') \in O] \times e^{\epsilon} \text{ .}$$

The probability is taken over the coin flips of Alg.

For example, a differentially private algorithm would provide the same output over a stream of e-mail messages with almost the same probability, even if one of the messages in the stream were to be replaced with another. The parameter ϵ controls the effect that any element in the stream might have on the outcome. A smaller value of ϵ means a lesser effect – and better privacy.

Differential privacy maintains composability, meaning that a series of b computations that are ϵ/b-differentially private is ϵ-differentially private. Composability gives rise to the concept of a *privacy budget*, in which a constraint on ϵ is determined in advance. A series of differentially private computations conducted by algorithms $A_1, A_2, \ldots$, each with a corresponding privacy parameter $\epsilon_1, \epsilon_2, \ldots$, can then be issued and processed as long as $\sum \epsilon_i \leq \epsilon$.

Example 3.2: This example follows up on Example 3.1. Recall that each e-mail service provider processes a stream consisting of w e-mail messages, $\{q_1, \ldots, q_w\}$. The service providers do not object to sending an aggregated statistics vector that reflects the appearance of terms in the messages they process. However, they would like to prevent any external entity, including other service providers and the spam filtering service, from inferring the presence of the term in any particular mail message. For example, consider a clinic that receives e-mail messages from patients. The clinic would obviously like to prevent an adversary from learning that a particular e-mail message contained a term such as "diabetes". An adversary who knows that a client sent an e-mail message at a certain time may learn that it contained that term directly from accurate statistics sent by the service provider (e.g., the adversary tracks the change in the value of the local statistics vector at the time the e-mail message was sent), or indirectly from the system behavior (e.g., if right after the client message was processed, the system identified that the monitored threshold was crossed). Moreover, the adversary could circumvent or weaken protections that rely on aggregates ("hiding in the crowd") by generating fake mail messages. The service provider can prevent such leaks by using a differentially private algorithm to process the data stream, since any resulting output sequence communicated to the coordinator (and consequently, any resulting system behavior) would be obtained with almost the same probability even if any particular e-mail message were to be replaced with a different e-mail message.

The privacy guarantee presented above considers only a single appearance of the elements q and q' (e.g., the effect of a single e-mail message), and consequently amounts to *event-level* privacy [4], where privacy is preserved with respect to each element in the stream, regardless of other elements that may be associated with the same individual. For example, in the context of the spam message scenario, this ensures that the contents of any particular mail message remain private, and have little effect on the probability of any output sequence. However, the system can still learn information concerning an aggregate of messages pertaining to the same user, for example, an analysis based on all the spam messages sent by the same spammer.

We note that no assumptions on the trustworthiness of the coordinator or other nodes are required to ensure that privacy is maintained. However, a rogue coordinator could lead to inefficient execution of the algorithm, and quick exhaustion of the privacy budget. In other words, privacy is guaranteed even if the coordinator and all other nodes are malicious, but correctness and efficiency guarantees hold only if the coordinator and nodes are honest (yet possibly curious).

B. Performance Goals

Differential privacy constrains the privacy risk inherent in information exchange. It is typically guaranteed by introducing noise to the computation process, and consequently incurs a cost in accuracy. To evaluate the trade-off between privacy and utility, we frame the following performance goals for the monitoring problem:

Recall We would like the system to identify global breaches of the threshold with as few misses as possible.[2]

Specificity We would like the system to give as few false alerts as possible.

Uptime When the differential privacy budget is exhausted, no further output updates are possible, and the system should be halted to maintain privacy. We would like to keep the monitoring process alive as long as possible within a given privacy budget.

IV. Background

In this section we present some tools that will be used in Section V.

[2]This property is also known as sensitivity; however, we avoid using this term in this context as it is also used in the differential privacy literature with a different meaning.

A. Differential Privacy Tools

We start by presenting two mechanisms that were used extensively in the differential privacy literature.

1) The Laplace Mechanism: To maintain differential privacy, the Laplace mechanism [2] adds noise sampled from the Laplace distribution when evaluating the value of a function. The noise is intended to mask the influence of any single element on the outcome, and is calibrated so as to hide this influence. Formally, this influence is bounded by the *global sensitivity* of the function, which is the largest possible change in the outcome for any pair of adjacent streams:

Definition 4.1 (Global Sensitivity [2]): The L_k-sensitivity of a function $g : S \to \mathbb{R}^d$ over a stream prefix S is

$$\Delta_k(g) = \max_{S \approx S'} \|g(S) - g(S')\|_k \ .$$

The Laplace distribution with mean 0 and variance $2z^2$ has probability density function $Pr(x|z) = \frac{1}{2z}\exp(-|x|/z)$. As was shown in [2], sampling noise from the Laplace distribution with scale $z = \Delta_1(g)/\epsilon$ and adding it to the value of the function results in a differentially-private computation:

Theorem 4.1 (Laplace Mechanism [2]): Given a function $g : S \to \mathbb{R}^d$ over a stream prefix S, the single-output computation $g'(S) = g(S) + \text{Laplace}(\Delta_1(g)/\epsilon)^d$ maintains ϵ-differential privacy.

Example 4.1: Consider the function $\text{count}_p(S)$, which returns the number of elements in the stream S that fulfill a predicate p. The L_1-sensitivity of $\text{count}_p(S)$ is 1, and the computation $\text{count}_p(S) + Laplace(1/\epsilon)$ maintains ϵ-differential privacy.

We also use in this paper the following property for the sum of independent Laplace distributions, which follows from [30, Theorem 6.5]:

Lemma 4.2 ([30]): Suppose γ_i's are n independent random variables, where each γ_i has Laplace distribution $Lap(z)$. Suppose $Y := \sum_i \gamma_i$. Then with probability of at least $1 - \delta$, the quantity $|Y|$ is at most $\sqrt{6n} \cdot z \log \frac{2}{\delta}$.

2) The Exponential Mechanism: The exponential mechanism [31] is useful for sampling one of several options in a differentially-private way, while taking into account the desirability of each option. A quality function q assigns a score to each of the options. This score is determined by the input of the algorithm, and higher scores signify more desirable outcomes. These scores, together with the privacy parameter ϵ, are then used to induce a probability distribution over the outcomes in a way that ensures differential privacy, while favoring outcomes with high scores.

Definition 4.2 (Exponential Mechanism [31]): Let $q : (\mathcal{D}^n \times O) \to \mathbb{R}$ be a quality function that, given a stream prefix S of length n, assigns a score to each outcome $r \in O$. Let $\Delta_1(q) = \max_{r, S \approx S'} |q(S,r) - q(S',r)|$. Let M be a mechanism for choosing an outcome $r \in O$ given a stream prefix S. Then the mechanism M, defined by

$$M(S,q) = \left\{ \text{return } r \text{ with probability} \propto \exp\left(\frac{\epsilon q(S,r)}{2\Delta_1(q)}\right) \right\} \ ,$$

maintains ϵ-differential privacy.

B. Communication-Efficient Monitoring and Safe Zones

The problem of monitoring a function over distributed streams in a communication-efficient way was studied by Sharfman et al. [13], [28]. One of the key steps in the proposed solution was to define the problem in terms of the input domain rather than the output range.

Definition 4.3 (Admissible Region): Given a function $f : \mathbb{R}^d \to \mathbb{R}$ and a threshold T, we define the admissible region $\mathcal{A}$ as the region where the value that f takes is at or below the threshold T:

$$\mathcal{A}_f(T) = \{\vec{v} \in \mathbb{R}^d | f(\vec{v}) \le T\} \ . \tag{2}$$

Given a set of local statistics vectors $\vec{v}_i(t)$ obtained from k nodes at time t, and the average vector $\vec{v}_g(t) = \sum_i \vec{v}_i(t)/k$, recall that a global breach of the threshold T occurs when $f(\vec{v}_g) > T$. As long as the average vector $\vec{v}_g(t)$ is within the admissible region, no global breach has occurred. To reduce communication costs, the global constraint imposed by the admissible region over the global vector is mapped to local constraints evaluated by each of the nodes over the local vectors. These constraints are expressed in the form of *Safe Zones*: they are constructed such that as long as all the local statistics vectors are within their respective safe zones, their average is guaranteed to be inside the admissible region, and thus no communication is required.

In general, safe zones can take any shape, though simple shapes, such as polygons with a small number of vertices, allow for more efficient algorithms. The techniques described in this paper are applicable to any chosen shape, but for simplicity we concentrate on ball-shaped safe zones. Specifically, we model each safe zone as a ball $B(\vec{c}_i, r)$, centered at $\vec{c}_i$ with radius r. The ball is chosen so that (a) it will be large, and (b) the local statistics vector will be far from its boundary. These two properties contribute to lower communication costs (and in our case, allow the privacy budget to be extended over a longer period, resulting in a longer lifetime of the system).

To assign safe zones to nodes, we start with an initial global vector $\vec{v}_g(0)$, which serves as a reference point, and use geometric techniques to fit a ball such that $B(\vec{c}, r) \subseteq \mathcal{A}_f(T)$ and $\vec{v}_g(0) \in B(\vec{c}, r)$. A description of the geometric methods used to fit the ball is out of the scope of this paper; we refer the reader to [28] for a full description of such techniques. Then we evaluate for each node i the *drift* vector $\vec{v}_i - \vec{v}_g$, and assign to the node the ball $B(\vec{c} + \vec{v}_i - \vec{v}_g, r)$. Essentially, this assignment aims to keep each local vector as far as possible from the boundary of the ball, while ensuring that the ball will function as a safe zone. The following theorem guarantees that as long as each of the local vectors $\vec{v}_i$ is within the respective assigned ball, the global average will remain in the admissible region.

Theorem 4.3: Let $\mathcal{A}_f(T)$ be the admissible region formed by the function f and a threshold T, and let $B(\vec{c}, r)$ be a ball with center $\vec{c}$ and radius r, such that $B(\vec{c}, r) \subseteq \mathcal{A}$. Given a set of k node-specific centers $\vec{c}_1, \ldots, \vec{c}_k$ such that $\sum_i \vec{c}_i/k = \vec{c}$, if for each node i, $\vec{v}_i \in B(\vec{c}_i, r)$, the mean vector $\vec{v}_g = \sum_i \vec{v}_i/k$ is within the admissible region $\mathcal{A}_f(T)$.

Proof: Denote by 0 the origin of $\mathbb{R}^n$. Then we can write $B(\vec{c}, r) = \{\, \vec{c} + \vec{x} |\, \vec{x} \in B(0, r)\}$ (i.e., $B(\vec{c}, r)$ is the translation of $B(0, r)$ by $\vec{c}$). Let $\vec{v}_i \in B(\vec{c}_i, r)$. Then

$$\frac{\vec{v}_1 + \cdots + \vec{v}_k}{k} = \frac{(\vec{c}_1 + \vec{x}_1) + \cdots + (\vec{c}_k + \vec{x}_k)}{k} \ ,$$

where $\vec{x}_i \in B(0, r)$, and then

$$\frac{\vec{c}_1 + \cdots + \vec{c}_k}{k} + \frac{\vec{x}_1 + \cdots + \vec{x}_k}{k} = \vec{c} + \vec{x} \ .$$

$B(0, r)$ is convex, hence closed under averaging. Since $\vec{x}_i \in B(0, r)$, and $\vec{x}$ is the average of $\vec{x}_i$, it follows that $\vec{x} \in B(0, r)$. Hence, the mean vector $\sum_i \vec{v}_i/k$ is in $B(\vec{c}, r)$, and therefore within the admissible region $\mathcal{A}_f(T)$. ■

Once the safe zones are assigned, each node can monitor its safe zone independently of the other nodes. If a node detects a local breach, it notifies the coordinator, who then collects the local statistics vectors from all the nodes in the system, and checks whether the new global vector breaches the admissible region. After the check, new safe zones can be assigned to the nodes on the basis of the new data. Figure 1 illustrates this process.

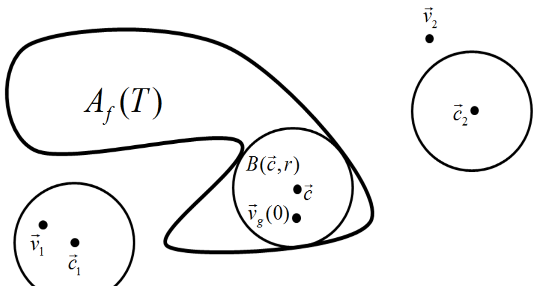

Fig. 1. A schematic description of the safe zones used in this paper, for two nodes. Depicted are the admissible region $A_f(T)$, the initial average vector $\vec{v}_g(0)$, and $B(\vec{c}, r)$, a maximal sphere containing $\vec{v}_g(0)$, which is contained in $A_f(T)$. The safe zones at the two nodes are spheres of the same size as $B(\vec{c}, r)$. As long as $\vec{v}_i$ is in its safe zone (left), no communication is initiated; if it wanders outside its safe zone (right), the coordinator initiates a "violation recovery" procedure.

Once the admissible region is breached, we can continue to monitor the streams to detect when the threshold is crossed back (i.e., the admissible region is "flipped"). Moreover, as discussed in [32], the threshold can be augmented with error margins to reduce communication costs due to thrashing when the global average is close to the threshold. The augmented threshold is used to set the admissible region and the safe zones, but once a local constraint is breached and the nodes synchronize to check for a global breach, the check is made against the original threshold. In other words, with a margin m, given the estimated global vector $\vec{v}_g(t)$, if the value of $f(\vec{v}_g(t))$ is below the original threshold T, the system uses $T' = (T + m)$ to set the admissible region and assign safe zones. When $f(\vec{v}_g(t))$ is above the threshold, the system uses $T' = (T - m)$ to set the new admissible region. For presentation purposes we assume without loss of generality that the monitored admissible region condition is always as in Equation 2; however, our results apply also when the threshold condition is flipped and when the threshold is augmented with error margins.

V. Distributed Stream Monitoring with Differential Privacy

In this section we present an algorithm for distributed stream monitoring with differential privacy. But first we describe a naive monitoring algorithm, which is easier to analyze.

A. Naive Algorithm

In a simple monitoring algorithm, each node releases in each round a noisy version of the local statistics vector, using the Laplace mechanism. The coordinator averages all these vectors, and checks whether the function of the global average has crossed the threshold T, in which case a global breach is identified. Because a new output is shared by each node every round, the number of rounds b that the process will run should be determined in advance and the differential privacy noise calibrated accordingly. In each round t, given the stream prefix $S_i(t)$ of node n_i, an aggregation function g is applied to derive the local statistics vector $\vec{v}_i(t) = g(S_i(t))$. The ϵ/b-differentially private output of each node is then given by $\vec{o}_i(t) = \vec{v}_i(t) + \text{Laplace}(b \cdot \Delta_1(g)/\epsilon)^d$.

Privacy guarantees of the naive algorithm follow from the Laplace mechanism and the composability property of differential privacy. The naive algorithm implies a direct trade-off between the monitoring period and the accuracy of the outcome, controlled by the parameter b. The following theorems state the recall and specificity guarantees for the naive algorithm:[3]

Theorem 5.1 (Naive algorithm – recall): If the global vector exceeds the admissible region $\mathcal{A}_f(T)$ by more than $(\log \frac{2d}{\delta} \cdot \sqrt{\frac{6d}{k}} \cdot b \cdot \Delta_1(g)/\epsilon)$, the naive algorithm will identify the global breach with probability of at least $(1 - \delta)$.

Proof: According to Lemma 4.2, with probability $1-\delta/d$, the sum of k Laplace variables across each dimension is at most $(\sqrt{6k} \log \frac{2d}{\delta} \cdot b \cdot \Delta_1(g)/\epsilon)$, and their average is at most $(\sqrt{\frac{6}{k}} \cdot \log \frac{2d}{\delta} \cdot b \cdot \Delta_1(g)/\epsilon)$. By the union bound, with probability $1 - \delta$ this holds across all d dimensions. Therefore the noisy global vector is within Euclidean distance $(\log \frac{2d}{\delta} \sqrt{\frac{6d}{k}} \cdot b \cdot \Delta_1(g)/\epsilon)$ from the real global vector. If the distance of the global vector from the admissible region is more than that, then the noisy global vector will stray out of the admissible region as well, and a breach will be detected. ■

Using a similar proof, a specificity guarantee follows:

Theorem 5.2 (Naive algorithm – specificity): If the global vector is more than $(\log \frac{2d}{\delta} \sqrt{\frac{6d}{k}} \cdot b \cdot \Delta_1(g)/\epsilon)$ inside the interior of the admissible region $\mathcal{A}_f(T)$, the naive algorithm will report a (false) global breach with probability of at most δ.

B. Safe-Zone-Based Algorithm Outline

Our goal is to monitor the local statistics vectors derived from the distributed input streams, and detect when a given

[3]The accuracy guarantees refer to the input domain and to the admissible region. When the monitored function f is a Lipschitz function, i.e., $\exists C$ s.t. $|f(x) - f(y)| \leq C|x - y|$ for all x and y, these guarantees also map to bounds that apply to the output range and the distance of $f(\vec{v}_g)$ from the threshold T.

function of the mean of these vectors crosses a given threshold. When monitoring streams in the distributed system, each node applies a local monitoring algorithm against a local safe zone.

In the local monitoring process we distinguish between two kinds of rounds. *Silent rounds* are ones in which the local statistics vector is within the safe zone, and the node does not produce any output. *Violation rounds* are ones in which the local statistics vector breaches the safe zone, and consequently the node notifies the coordinator. The coordinator then initiates a process of *violation recovery* to determine whether the global threshold was crossed. While violation rounds require an explicit exchange of information, privacy leaks should also be accounted for in silent rounds, as the fact that the local statistics vector is within the safe zone also conveys information to an adversary.

A change in a single element in the stream could affect the value of the local statistics vector in a sequence of rounds. To preserve differential privacy, the algorithm should produce any possible output sequence with almost the same probability when any particular element changes. Specifically, silent rounds should remain silent, violation rounds should still result in an alert, and the violation recovery process should produce similar outputs.

The algorithm starts with an initialization phase, detailed in Section V-C, where the nodes establish their initial local statistics vectors and send noisy versions of them to the coordinator. The coordinator then uses these vectors to assign a safe zone to each node. Each node then monitors the local safe zone in a privacy-preserving way, as detailed in Section V-D, and alerts the coordinator if a local safe zone breach is detected. In that case, the coordinator launches a violation recovery process, which is described in Section V-E. In the recovery process the coordinator collects noisy statistics vectors from the nodes to check for a global violation (i.e., to evaluate whether $f(\vec{v}_g(t)) > T$), and then re-assigns safe zones to the nodes.

C. Initialization phase

The initialization phase takes place during the first time period ($t = 0$), and its goal is to assign safe zones to each of the nodes. Each of the nodes (Algorithm 1) establishes the initial local statistics vector and sends a noisy version of it to the coordinator. The coordinator then verifies that the initial global vector is within the admissible region (a global breach is declared otherwise). To assign safe zones, the coordinator (Algorithm 2) finds the largest ball $B(\vec{c}, r)$ that can fit within the admissible region while containing the global vector. Then, in accordance with Theorem 4.3, each node is assigned a center $\vec{c}_i$ such that the centers average at $\vec{c}$.

D. Local Monitoring of Safe Zones

The local monitoring process is described in Algorithm 3.

Once a local node obtains a safe zone from the coordinator, it monitors the local statistics vector against that safe zone. To maintain privacy, three noise elements are introduced in the algorithm. First, the safe zone is perturbed by adding noise to the radius to obtain the ball $B(\vec{c}_i, \hat{r})$. The same perturbed safe zone is used until a new safe zone is assigned by the coordinator, and it protects privacy during a sequence of silent rounds. Second, in each round, the node checks whether its local statistics vector is within the perturbed safe zone, using the exponential mechanism. Fresh noise is used in each inclusion check, and it protects privacy when local violations occur. Finally, when the coordinator initiates violation recovery following a local breach in one or more of the nodes, each node uses the Laplace mechanism to send the coordinator a noisy version of its local statistics vector. The Laplace mechanism maintains privacy throughout violation recovery.

Fig. 2. Algorithm 1: NodeInitialization(S_i, g, b, ϵ)

```
Input:
    $S_i$ – a local input stream
    $g$ – a function that generates a $d$-dimensional local statistics vector from $S_i$
    $b$ – bound on the number of violation rounds
    $\epsilon$ – privacy parameter
1: $\vec{v}_i(0) \leftarrow g(S_i(0))$
2: Sample $\vec{n}_{i,0} \sim \text{Laplace}\left(\frac{3(b+1)\cdot\Delta_1(g)}{\epsilon}\right)^d$
3: Send to the coordinator $\vec{o}_i(0) = \vec{v}_i(0) + \vec{n}_{i,0}$
```

Fig. 3. Algorithm 2: CoordinatorInitialization(f, T, k)

```
Input:
    $f$ – a global function to monitor
    $T$ – a threshold for $f$
    $k$ – number of nodes
1: Obtain $\vec{o}_i(0)$ from the nodes
2: $\vec{v}_g(0) \leftarrow \sum \vec{o}_i(0)/k$
3: if $f(\vec{v}_g(0)) > T$ then report a global breach
4: $\vec{c}, r \leftarrow \arg\max_{\vec{c},r}(Vol(B(\vec{c}, r)))$ subject to $B(\vec{c}, r) \subseteq \mathcal{A}_f(T)$ and $\vec{v}_g(0) \in B(\vec{c}, r)$
5: $\forall i : \vec{c}_i \leftarrow \vec{c} + \vec{o}_i(0) - \vec{v}_g(0)$
6: Assign to each node $i$ the safe zone $B(\vec{c}_i, r)$
```

Algorithm 4 details how the exponential mechanism is applied to evaluate whether a given local statistics vector $\vec{v}_i(t)$ is within the (noisy) safe zone $B(\vec{c}_i, \hat{r})$. The quality function is set to $q(\textbf{true}) = \hat{r} - \text{dist}(\vec{v}_i(t), \vec{c}_i)$ (the distance from the boundary of the safe zone) for a result indicating inclusion in the safe zone and $q(\textbf{false}) = -(\hat{r} - \text{dist}(\vec{v}_i(t), \vec{c}_i))$ otherwise. The L_1-sensitivity of q is $\Delta_2(g)$.

E. The Coordinator Algorithm and Violation Recovery

The global monitoring algorithm is orchestrated by the coordinator, as shown in Algorithm 5. To check whether the global threshold was crossed following a local breach, the coordinator gathers the noisy local statistics vectors from all the nodes in the system, and evaluates whether their average is within the admissible region. The global average vector is used also to reassign safe zones to the nodes in the system, regardless of whether a global breach is detected.

F. Accuracy and Privacy Guarantees

In this section we state the accuracy and privacy guarantees provided by the system.

Fig. 4. Algorithm 3: LocalMonitoring(S_i, g, b, ϵ)

```
Input:
    $S_i$ – a local input stream
    $g$ – a function that generates a $d$-dimensional local statistics vector from $S_i$
    $b$ – bound on the number of violation rounds
    $\epsilon$ – privacy parameter
1: $m \leftarrow 1$
2: Acquire new safe zone $B(\vec{c}_i, r)$ from the coordinator
3: Sample $\alpha_i \sim$ Laplace $\left(\frac{3b \cdot \Delta_2(g)}{\epsilon}\right)$
4: for each round $t$ do
5:     $\vec{v}_i(t) = g(S_i(t))$
6:     if Evaluate($\vec{v}_i(t) \in_\epsilon B(\vec{c}_i, r + \alpha_i)$) returns false then report a local breach
7:     if the coordinator initiates violation recovery then
8:         Sample $\vec{n}_{i,t} \sim$ Laplace$(\frac{3(b+1) \cdot \Delta_1(g)}{\epsilon})^d$
9:         Send $\vec{o}_i(t) = \vec{v}_i(t) + \vec{n}_{i,t}$ to the coordinator
10:        if $m < b$ then $m \leftarrow m + 1$ else HALT
11:        Continue from step 2
12:    end if
13: end for
```

Fig. 5. Algorithm 4: Evaluate($\vec{v}_i(t) \in_\epsilon B(\vec{c}_i, \hat{r})$)

```
Input:
    $\vec{v}_i(t) = g(S_i(t))$ – a local statistics vector
    $B(\vec{c}_i, \hat{r})$ – a ball with center $\vec{c}_i$ and radius $\hat{r}$ denoting a (perturbed) safe zone
    $\epsilon$ – privacy parameter
1: $\mu = \frac{\epsilon}{6b} \cdot \frac{\hat{r} - \text{dist}(\vec{v}_i(t), \vec{c}_i)}{2\Delta_2(g)}$
2: Sample $u_{i,t} \sim U[0, 1]$
3: return true if $u_{i,t} \leq \frac{\exp(2\mu)}{1+\exp(2\mu)}$, and false otherwise
```

1) Accuracy Guarantees:

Theorem 5.3 (Accuracy – recall): With probability of at least $(1 - 2\delta)$, if a local node has not halted, and its local statistics vector exceeds the safe zone assigned by the coordinator by more than $6b \cdot \Delta_2(g) \log \frac{1-\delta}{\delta^{1.5}}/\epsilon$, Algorithm 3 will identify the breach.

Proof: Assume that for a given node Algorithm 3 has not halted by time t. The error in the evaluation step in Line 6 stems from the randomness $u_{i,t}$ in Algorithm 4, as well as from the perturbation of the safe zone radius with α_i. With probability of at least $1 - \delta$, the value of $|\alpha_i|$ is at most $3b \cdot \Delta_2(g) \log \frac{1}{\delta}/\epsilon$. Moreover, with probability of at least $1 - \delta$, if the local statistics vector exceeds the perturbed safe zone by more than $6b \cdot \Delta_2(g) \log \frac{1-\delta}{\delta}/\epsilon$, Algorithm 4 will identify the breach. Taking a union bound, we obtain that with probability of at least $1 - 2\delta$, Algorithm 3 will declare a local breach if the local vector's distance from the original safe zone is more than $6b \cdot \Delta_2(g) \log \frac{1-\delta}{\delta^{1.5}}/\epsilon$. ■

A similar proof provides the specificity guarantees.

Theorem 5.4 (Accuracy – specificity): If a local node has not halted, and its local vector is inside the safe zone assigned by the coordinator, with distance of at least $6b \cdot \Delta_2(g) \log \frac{1-\delta}{\delta^{1.5}}/\epsilon$ from its boundary, Algorithm 3 will report a breach with probability of at most 2δ.

Fig. 6. Algorithm 5: Coordinator(f, T, b, k)

```
Input:
    $f$ – a global function to monitor
    $T$ – a threshold for $f$
    $b$ – bound on the number of violation rounds
    $k$ – number of nodes
1: $m \leftarrow 1$
2: while $m \leq b$ do
3:     if any node reports a local violation in round $t$ then
4:         Announce violation recovery and collect $\vec{o}_i(t)$ from the nodes
5:         $\vec{v}_g(t) \leftarrow \sum \vec{o}_i(t)/k$
6:         if $f(\vec{v}_g(t)) > T$ then report a global breach
7:         $m \leftarrow m + 1$
8:         $\vec{c}, r \leftarrow \arg\max_{\vec{c}, r}(Vol(B(\vec{c}, r)))$ subject to $B(\vec{c}, r) \subseteq \mathcal{A}_f(T)$ and $\vec{v}_g(t) \in B(\vec{c}, r)$
9:         $\forall i : \vec{c}_i \leftarrow \vec{c} + \vec{o}_i(t) - \vec{v}(t)$
10:        Assign to each node $i$ the safe zone $B(\vec{c}_i, r)$
11:    end if
12: end while
```

Once a local breach is detected and the coordinator has checked for a global breach, similar accuracy guarantees to those stated in Theorems 5.1 and 5.2 for the naive algorithm apply also to Algorithm 5 (albeit with larger noise magnitude), as the same mechanism is used to evaluate the global vector and check for a breach.

2) Privacy Guarantees:

Theorem 5.5: Algorithm 3 maintains ϵ-differential privacy.

Proof:

To prove that Algorithm 3 maintains ϵ-differential privacy, we follow the proof technique of [3, Theorem 5.2], which incorporates several noise components to obtain privacy and accuracy guarantees when a single-output differentially-private algorithm is transformed to a T-round continual output algorithm. We focus on a single node i and fix the execution of the coordinator and all the other nodes in the system. Given two adjacent streams $S_i \approx S'_i$ and an output sequence O, consider an execution E_{S_i} of node i over input stream S_i, and denote by R the series of random variables sampled throughout this execution, such that $Alg_R(S_i) = O$ (If no such R exists, then $Pr[Alg(S_i) = O] = 0$). Given R, we will describe a corresponding execution $E_{S'_i}$ over stream S'_i with randomness R' such that $Alg_{R'}(S'_i) = O$, and the probability density of R' differs by a factor of at most $\exp(\epsilon)$ from that of R. As this holds for any choice of adjacent streams $S_i \approx S'_i$, and any possible output O, this proves ϵ-differential privacy for Algorithm 3.

We consider the output sequence O as a concatenation of several output sequences of the form $[\vec{o}_i(0), O_1^{(S)}, O_1^{(V)}, \ldots, O_{b'}^{(S)}, O_{b'}^{(V)}]$, as illustrated in Figure 7, where $\vec{o}_i(0)$ is the output of the initialization round, each $O_j^{(S)}$ is a (possibly empty) sequence of silent rounds, and this sequence of silent rounds is followed by a violation round $O_j^{(V)}$ (which includes also violation recovery). We have $b' \leq b$ due to the condition in line 10 of the algorithm. Any output stream that does not conform to this format would have probability 0 regardless of the input stream.

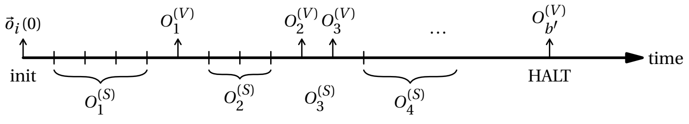

Fig. 7. Breaking an output sequence into multiple sequences

We next focus on each of the components that comprise the stream O, and show how the noise values sampled in the executions E_{S_i} and $E_{S'_i}$ should be different from each other if they are to result in the same output.

Initialization round In the initialization round of the executions E_{S_i} and $E_{S'_i}$, the node generates the local statistics vectors $\vec{v}_i(0) = g(S_i(0))$ and $\vec{v}'_i(0) = g(S'_i(0))$ respectively. To obtain the output $\vec{o}_i(0)$ in both executions, we have

$$|\vec{n}_{i,0} - \vec{n}'_{i,0}|_1 = |\vec{v}_i(0) - \vec{v}'_i(0)|_1 \leq \Delta_1(g) \quad .$$

Since the noise vectors are sampled from the Laplace distribution with scale $\frac{3(b+1)\cdot\Delta_1(g)}{\epsilon}$ on each dimension, the probability density of generating $\vec{n}'_{i,0}$ differs by a factor of at most $\exp(\frac{\epsilon}{3(b+1)})$ from that of generating $\vec{n}_{i,0}$.

Silent rounds Consider an uninterrupted sequence of l silent rounds $O_j^{(S)}$ over time periods $t_m, t_{m+1}, \ldots, t_{m+l-1}$.

We keep the noise elements $u_{i,t}$ the same for rounds $[t_m, t_{m+l-1}]$ in executions E_{S_i} and $E_{S'_i}$, and show how the same outcome (Algorithm 4 returns true) would be obtained in both executions throughout those rounds.

The safe zone perturbation noise α_i is generated in line 3 following the initialization round and any violation round, and remains the same throughout any uninterrupted sequence of silent rounds. Given the noise α_i in time periods $[t_m, t_{m+l-1}]$ for execution E_{S_i}, we set $\alpha'_i = \alpha_i + \Delta_2(g)$ for the same time periods in execution $E_{S'_i}$ (note that the same α'_i applies also to the violation recovery round at t_{m+l}, which will be addressed in the next paragraph). For each of the silent rounds $t \in [t_m, t_{m+l-1}]$ on execution E_{S_i}, the check $Evaluate(\vec{v}_i(t) \in_\epsilon B(\vec{c}_i, \hat{r}))$ returns true, i.e., $u_{i,t} \leq \frac{\exp(2\mu)}{1+\exp(2\mu)}$, where

$$\mu = \frac{\epsilon}{6b} \cdot \frac{\hat{r} - \text{dist}(\vec{v}_i(t), \vec{c}_i)}{2\Delta_2(g)} \quad .$$

Since $\hat{r}' = r + \alpha'_i = r + \alpha_i + \Delta_2(g)$, and

$$\begin{aligned}\text{dist}(\vec{v}'_i, \vec{c}_i) &\leq \text{dist}(\vec{v}'_i, \vec{v}_i) + \text{dist}(\vec{v}_i, \vec{c}_i) \leq \\ &\leq \Delta_2(g) + dist(\vec{v}_i, \vec{c}_i) \quad ,\end{aligned}$$

we get that $\hat{r} - \text{dist}(\vec{v}_i(t), \vec{c}_i) \leq \hat{r}' - \text{dist}(\vec{v}'_i(t), \vec{c}_i)$, and therefore $\mu \leq \mu'$, so that $Evaluate(\vec{v}'_i(t) \in_\epsilon B(\vec{c}_i, \hat{r}')$ also returns true for all $t \in [t_m, t_{m+l-1}]$.

Because $\alpha_i \sim \text{Laplace}\,(3b \cdot \Delta_2(g)/\epsilon)$, the probability to obtain $\alpha'_i = \alpha_i + \Delta_2(g)$ differs by a factor of at most $\exp(\frac{\epsilon}{3b})$ from that of obtaining α_i.

Violation rounds Consider a time period t on execution E_{S_i}, in which violation recovery took place. The violation recovery process could be triggered by the coordinator, due to a local breach detected on another node, or it could be the result of a safe zone breach on the local node.

In the first case, the choice of $\alpha'_i = \alpha_i + \Delta_2(g)$ discussed in the previous paragraph ensures that the violation recovery event will not be triggered by the local node also for execution $E_{S'_i}$. In the second case, privacy protection is ensured by the differentially-private comparison (Algorithm 4), which relies on the exponential mechanism. Since

$$\hat{r}' - \text{dist}(\vec{v}'_i(t), \vec{c}_i) \leq \hat{r} - \text{dist}(\vec{v}_i(t), \vec{c}_i) + \Delta_2(g) \quad ,$$

we have $\mu' \leq \mu + \frac{\epsilon}{6b}$. It follows that the probability that the algorithm will return false in $E_{S'_i}$ differs by a factor of at most $\exp(\frac{\epsilon}{3b})$ from that for E_{S_i}. Example 5.1 below demonstrates why a separate noise element is needed to maintain ϵ-differential privacy in violation rounds.

It remains to show how the same output $\vec{o}_i(t)$ for the violation recovery can be maintained for execution $E_{S'_i}$. As it does in the initialization round, the Laplace mechanism ensures that the probability of obtaining the same output $\vec{o}_i(t)$ on each violation round in $E_{S'_i}$ differs by a factor of at most $\exp(\frac{\epsilon}{3(b+1)})$ from that of execution E_{S_i}.

Bringing it all together All the noise elements are sampled independently, so the ratio of probabilities of obtaining the output sequence O for the executions E_{S_i} and $E_{S'_i}$ can be bounded by multiplying the bounds of the individual steps described above. As the initialization round occurs once, and each of the other cases occurs at most b times, the overall probability of obtaining the output sequence in execution $E_{S'_i}$ differs by a factor of at most $\exp(\epsilon)$ from that of execution E_{S_i}. ■

In the described algorithm, the privacy budget is distributed evenly between the different components of the algorithm (safe zone perturbation, the safe zone inclusion check, and local statistics vector perturbation) – a privacy budget of $\frac{\epsilon}{3b}$ is assigned to each check ($\frac{\epsilon}{3b+1}$ when perturbing the local statistics vector, to account also for the initialization round). However, it is also possible to assign a different portion of the budget to each of components. For example, we could introduce less noise when perturbing the safe zones, in exchange for increased noise in the inclusion check, by assigning a budget of $\frac{4\epsilon}{9b}$ to the first and a budget of $\frac{2\epsilon}{9b}$ to the second. However, after experimenting with different privacy budget distributions, we found that an even distribution of the budget performed better than the alternatives for our experimental setup.

The following example illustrates why protection of privacy in violation rounds requires the use of a differentially-private comparison (Algorithm 4), beyond safe zone perturbation.

Example 5.1: Consider a node that monitors locally a (one-dimensional) counter over its data stream, and checks it against a safe zone with center c and radius r (i.e., the safe zone will be breached when the counter goes above $c + r$ or below $c - r$). Figure 8 shows the value of the counter p_i in three consecutive rounds in a certain execution. The counter was within the safe zone in the first two rounds, resulting in silent rounds, and outside the safe zone in the third round, resulting in a breach of the safe zone. Now consider an alternative execution, in which one of the counted events did not occur, resulting in lower counts p'_1, p'_2 and p'_3. To obtain the same output as in the former execution, the first round should still be silent, and the third round should still result in a breach. However, using only safe zone perturbation, there is no perturbed safe zone radius r' such that $p'_1 \in B(c_i, r')$ (requiring $r' > r$) while $p'_3 \notin B(c_i, r')$ (requiring $r' < r$) simultaneously.

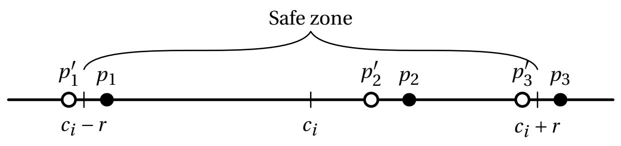

Fig. 8. A local node using a (one-dimensional) safe zone to monitor a counter

VI. Experimental Evaluation

To evaluate the performance of the privacy-preserving distributed monitoring algorithm, we used the Reuters Corpus (RCV1-v2) [33] as a source of input streams. The test set in the corpus contains 781,265 news stories, produced by Reuters between August 20, 1996 and August 19, 1997. The stories were processed to extract feature vectors, and each document was also labeled as belonging to several categories. In the experimental setup the stories were distributed by round robin between 10 nodes, where each node processes in each round a window containing the last 10,000 stories.

In the experiments our goals were to simulate monitoring of the number of spam messages, and to simulate feature selection for spam identification. To this end, we followed the experimental setup described in [13], where a similar feature selection scenario was evaluated without any privacy constraints. We chose the "CCAT" (Corporate/Industrial) category, which is the most frequent category label in the dataset, as denoting a spam message. In one experiment we monitored the number of spam messages, and in another we monitored the information gain of one of the extracted features ("febru") to determine its value in identifying a spam message. Figure 9 illustrates the global average values of the "CCAT" count and the "febru" information gain in the processed rounds.

Monitoring the spam message count requires aggregating 1-dimensional vectors, $g_1(S_i(t)) = |q \in S_i(t) \wedge \texttt{CCAT}|$, with

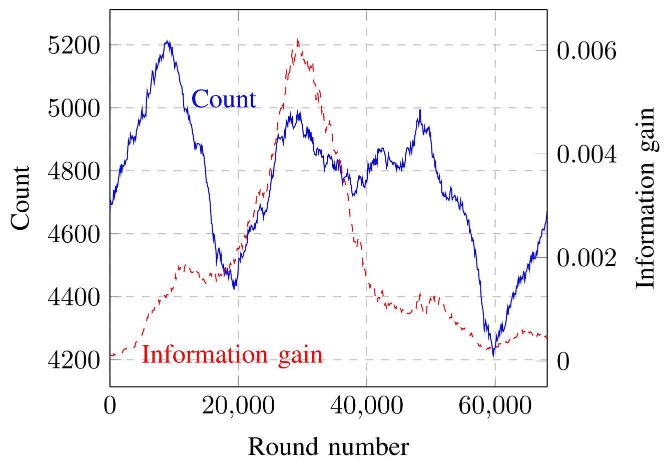

Fig. 9. The monitored global average values of the "CCAT" category (count) and the "febru" feature (information gain) in the Reuters dataset

global sensitivities $\Delta_1(g_1) = \Delta_2(g_1) = 1$. Monitoring the information gain function requires aggregating 3-dimensional vectors $g_2(S_i(t)) = |S_i(t)| \cdot (c_{1,1}, c_{1,2}, c_{2,1})$ (see Example 3.1; the 4th coordinate can be obtained from the other three because of the fixed window size), with global sensitivities $\Delta_1(g_2) = 2$ and $\Delta_2(g_2) = \sqrt{2}$. We conducted experiments with different values for the bound b on the number of violation rounds, and report below results for $b = 5$. All the results reported below were obtained by averaging over 10 executions of each experiment with different random seeds.

As a baseline, we also conducted experiments where the same data streams are consumed and processed by a single node. These experiments simulate an "ideal" setting in which a trusted third party aggregates all the local vectors and determines whether the threshold was crossed. To ensure that the output maintains privacy, the single node employs the same (safe-zone-based) mechanism to evaluate whether the threshold was breached.

Figure 10 shows the trade-off between the privacy parameter ϵ and the system lifetime in rounds when monitoring the number of spam messages, for two different threshold values, $T = 4800$ and $T = 5000$. Similar trade-offs were observed also for other threshold values that we tested. When the global vector is close to the threshold, the likelihood of local breaches is much higher, requiring frequent communication between the nodes, and faster depletion of the violation round limit. Consequently, when monitoring for $T = 4800$ without error margins, the system lifetime is much shorter than when monitoring for $T = 5000$. Allowing for error margins (Figure 10 reports the results for a margin of 100) considerably mitigates this problem. The margin reduces the likelihood of local breaches in the absence of a global breach, and allows global breaches to be detected with fewer violation rounds. Thus the margin allows us to trade off monitoring accuracy for longer system lifetime within the given privacy constraint. Figure 10 also shows the trade-off for the baseline, in which the streams are aggregated by a single trusted entity (while still ensuring a differentially private output). In this setting, the system can operate with the same privacy constraint over longer periods of time than in the distributed setting, due to two contributing factors: (i) since the privacy mechanism is employed by a single node rather than by multiple nodes, a local breach due to an "unlucky" coin toss that produces a high level of noise is less likely to occur in any given round; and (ii) regardless of the privacy protection mechanism, direct monitoring of the global vector with a single safe zone results in fewer breaches of the safe zone constraint than in the case of monitoring ten different local statistics vectors with separate local safe zones – changes in a local statistics vector that would have caused a safe zone breach in the distributed case may be offset by changes in other nodes when aggregated by a trusted third party, thereby precluding a safe zone breach in the baseline scenario.

Figure 11 illustrates the results for monitoring the information gain with different threshold values (and the same error margin 0.0005), and shows the three-way trade-off between the privacy parameter ϵ, the system lifetime in rounds, and the monitored threshold T. Note that the signal in the information gain experiment is much weaker than in the count experiment: about half of the processed messages in the dataset

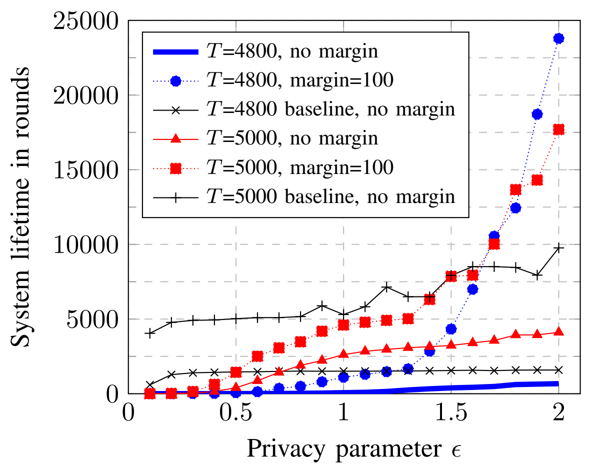

Fig. 10. Monitoring the number of spam messages (count)

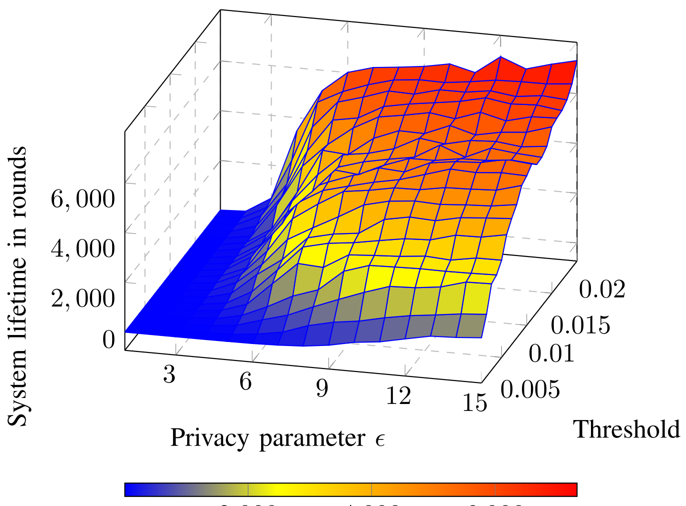

Fig. 11. Monitoring the information gain of the "febru" feature with error margin 0.0005

are categorized as spam, but there are an order of magnitude fewer messages that have the "febru" feature. Monitoring for the "febru" feature is thus much more sensitive to noise than monitoring the number of messages. Generally, the effect of the weaker signal could be mitigated by processing larger amounts of data. In the experiment we used a larger value of ϵ instead, to obtain a similar effect. The information gain experiment reflects similar trade-offs between privacy and system lifetime as those observed in the count experiment. In addition, we varied the value of the threshold to evaluate its impact on the system. The farther the monitored threshold is from the actual values measured throughout typical system operation, the less the likelihood of local breaches that require communication between the nodes and further loss of privacy.

Figure 12 shows the three-way trade-off for the trusted-third-party baseline. As before, this setting allows for longer system lifetime within the same privacy constraint. Moreover, since the aggregated signal is stronger, the effect of the privacy parameter ϵ in the evaluated range is negligible with respect to that of the monitored threshold.

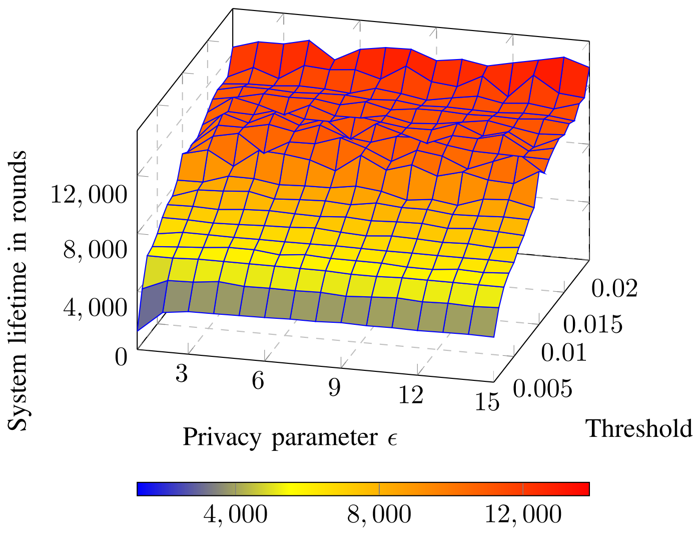

Fig. 12. Trusted-third-party baseline for monitoring the information gain of the "febru" feature with error margin 0.0005

The outcome of the monitoring process was highly accurate within the given error margins. Over all the conducted experiments, there were on average 2.9 ± 1.2 false positives and 8 ± 2.6 false negatives in the evaluation of the condition on the global count, and 0.2 ± 0.6 false positives and 3.6 ± 10.8 false negatives in the evaluation of the condition on the global information gain, out of hundreds and thousands of monitoring rounds. The output of the privacy-preserving system reflected the actual system state in more than 99.5% of the monitored rounds.

The naive algorithm: we note that for any value of ϵ, running the naive algorithm with the same accuracy guarantees as in the setup described above (per Theorems 5.1 and 5.2) would allow only for 18 rounds (the noise $\vec{n}_{i,t}$ in Algorithm 3 is sampled from the distribution Laplace$(\frac{18\Delta_1(g)}{\epsilon})$).

A. Additional Trade-offs

Beyond the direct trade-off between privacy and system lifetime, additional factors affect the balance between the two. In this section we explore how setting different values for the error margins and the number of violation rounds affects system performance.

1) Varying Error Margins: As mentioned in Section IV-B, the threshold can be augmented with error margins to mitigate excessive communications when the global average vector is close to the threshold. The augmented threshold is used to set the admissible region and the safe zones, but once a local constraint is breached and the nodes synchronize to check for a global breach, the check is made against the original threshold.

Beyond the smaller communication costs, augmenting the threshold with error margins also makes the monitoring algorithm more resilient to the noise introduced for enhancing privacy. The accuracy in evaluating the monitored global condition can thereby be traded-off for increased system lifetime or a stricter privacy constraint.

Figures 13 and 14 show the privacy accuracy trade-offs when monitoring the spam messages count with thresholds

$T = 4800$ and $T = 5000$ respectively. Similarly, figure 15 shows the trade-off when monitoring the information gain of the "febru" feature with the threshold $T = 0.002$. Higher error margins mean that the admissible region is larger, and consequently the safe zones assigned to the nodes are larger. The larger safe zones reduce the number of false positives incurred due to the noisy monitoring process, such that there are fewer synchronization rounds. Consequently, the system can be monitored for longer periods of time with the same privacy constraints.

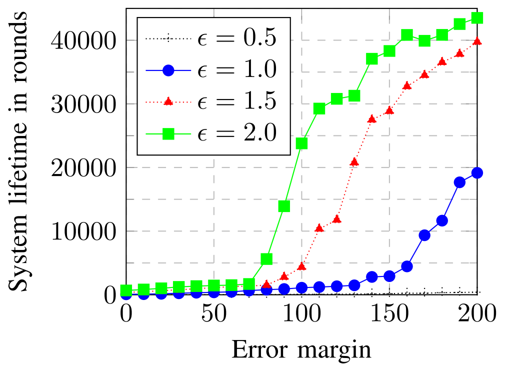

Fig. 13. Different error margins when monitoring the number of spam messages against a threshold of $T = 4800$ with 5 violation rounds

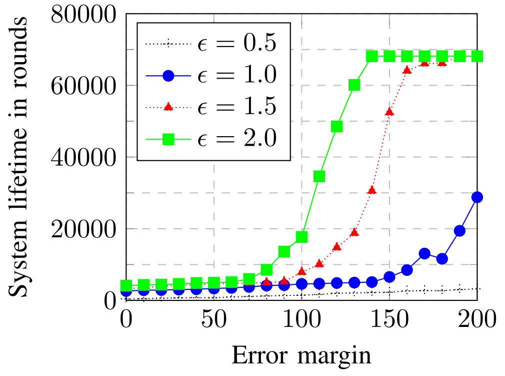

Fig. 14. Different error margins when monitoring the number of spam messages against a threshold of $T = 5000$ with 5 violation rounds

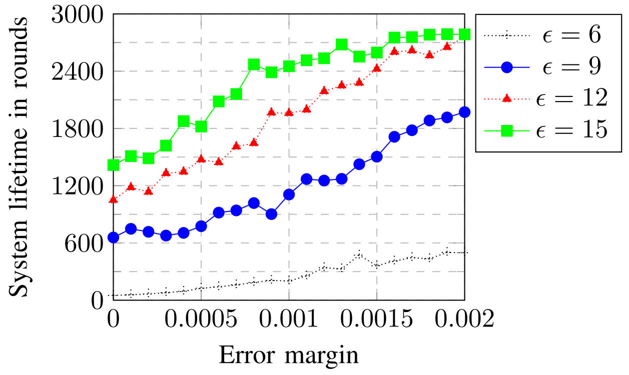

Fig. 15. Different error margins when monitoring the information gain of the "febru" feature against a threshold of $T = 0.002$ with 5 violation rounds

2) Varying Number of Violation Rounds: The bound b on the number of violation rounds has two opposing effects on the system lifetime. Increasing the bound allows the system to sustain more local breaches (and the resulting synchronization rounds) before the privacy budget is exhausted. However, at the same time, the privacy budget assigned for monitoring each sequence of silent rounds that ends with a violation round will be smaller. This in turn will introduce more noise into the monitoring process, increasing the likelihood (and therefore the frequency) of violation rounds.

Figures 16 and 17 show the effect of different bounds on the number of violation rounds when monitoring the number of spam messages with thresholds $T = 4800$ and $T = 5000$ respectively. Similarly, Figure 18 shows that effect when monitoring the information gain of the "febru" feature with the threshold $T = 0.002$. The aforementioned opposing effects of the number of violation rounds are evident in the decrease in system lifetime when the number of violation rounds is too low or too high. The system lifetime peaks in the "sweet spot" where these opposing effects are balanced. Furthermore, when the privacy constraints are weaker and ϵ is higher, it is possible to sustain more violation rounds without a large increase in false positives, resulting in longer system lifetime.

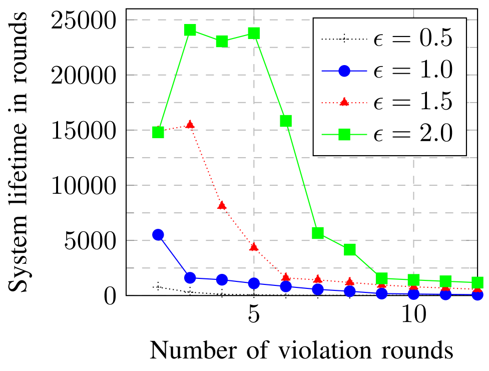

Fig. 16. Different number of violation rounds when monitoring the number of spam messages. $T = 4800$ with an error margin of 100

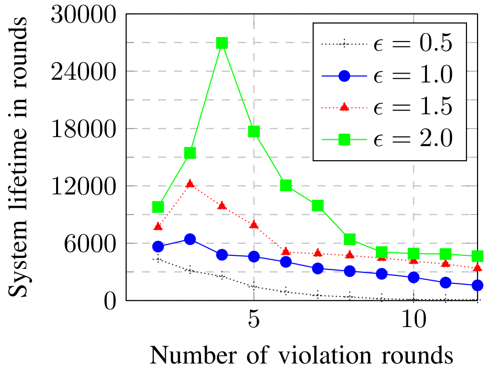

Fig. 17. Different number of violation rounds when monitoring the number of spam messages. $T = 5000$ with an error margin of 100

VII. Discussion and Future Work

We demonstrated in this paper how communication-efficient distributed monitoring algorithms can be leveraged

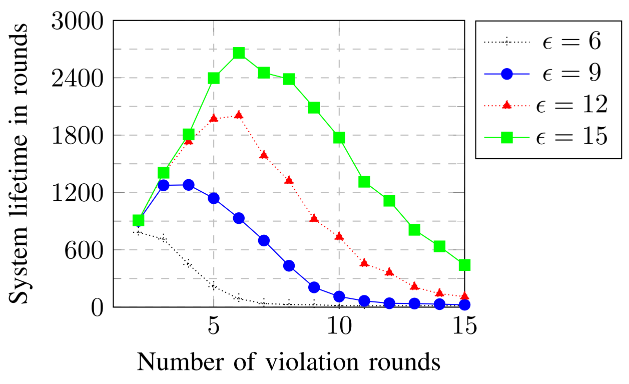

Fig. 18. Different number of violation rounds when monitoring the information gain of the "febru" feature. $T = 0.002$ with an error margin of 0.001

towards privacy-preserving monitoring of a global condition over long periods of time. We provided theoretical analysis of the proposed algorithm, and evaluated experimentally some of the trade-offs between the privacy constraint, the system lifetime, and the monitored threshold. Beyond these direct trade-offs, we studied additional factors that affect the balance between privacy and system performance: error margins that augment the monitored threshold, and the bound on the number of violation rounds.

For future research, we note that more sophisticated methods for violation recovery, such as local communication between nodes rather than global synchronization, could allow, in some cases, further mitigation of privacy loss while monitoring the system. Prediction models that tailor safe zones to nodes [34], [35] also show promise in reducing the probability of local safe zone breaches, allowing further increase in the system lifetime. In addition, once the window of processed elements advances beyond elements that contributed to violation rounds, the privacy loss induced by those rounds can be discounted. This gives rise to the option of replenishing the violation round limit, allowing further extension of the system lifetime within the given privacy constraint.

VIII. Acknowledgments

This work was partially supported by the European Commission under ICT-FP7-LIFT-255951 (Local Inference in Massively Distributed Systems).

References

[1] J. A. Calandrino, A. Kilzer, A. Narayanan, E. W. Felten, and V. Shmatikov, ""you might also like:" privacy risks of collaborative filtering," in *IEEE Symposium on Security and Privacy*, 2011, pp. 231–246.

[2] C. Dwork, F. Mcsherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Proceedings of the 3rd Theory of Cryptography Conference*. Springer, 2006, pp. 265–284.

[3] C. Dwork, T. Pitassi, M. Naor, and G. N. Rothblum, "Differential privacy under continual observation," in *STOC*, 2010, pp. 715–724.

[4] C. Dwork, M. Naor, T. Pitassi, G. N. Rothblum, and S. Yekhanin, "Pan-private streaming algorithms," in *ICS*, 2010, pp. 66–80.

[5] T. H. H. Chan, E. Shi, and D. Song, "Private and continual release of statistics," in *ICALP*, 2010, pp. 405–417. [Online]. Available: http://dl.acm.org/citation.cfm?id=1880999.1881044

[6] D. J. Mir, S. Muthukrishnan, A. Nikolov, and R. N. Wright, "Pan-private algorithms via statistics on sketches," in *PODS*, 2011, pp. 37–48.

[7] L. Fan and L. Xiong, "Real-time aggregate monitoring with differential privacy," in *CIKM*, 2012, pp. 2169–2173.

[8] J. Hsu, S. Khanna, and A. Roth, "Distributed private heavy hitters," in *ICALP (1)*, 2012, pp. 461–472.

[9] L. Huang, M. N. Garofalakis, J. M. Hellerstein, A. D. Joseph, and N. Taft, "Toward sophisticated detection with distributed triggers," in *MineNet*, 2006, pp. 311–316.

[10] S. Agrawal, S. Deb, K. V. M. Naidu, and R. Rastogi, "Efficient detection of distributed constraint violations," in *ICDE*, 2007, pp. 1320–1324.

[11] G. Cormode and M. N. Garofalakis, "Approximate continuous querying over distributed streams," *ACM Transactions on Database Systems*, vol. 33, no. 2, 2008.

[12] C. Arackaparambil, J. Brody, and A. Chakrabarti, "Functional monitoring without monotonicity," in *ICALP (1)*, 2009, pp. 95–106.

[13] I. Sharfman, A. Schuster, and D. Keren, "A geometric approach to monitoring threshold functions over distributed data streams," in *SIGMOD*, 2006, pp. 301–312.

[14] L. Huang, X. L. Nguyen, M. Garofalakis, J. M. Hellerstein, M. I. Jordan, A. D. Joseph, and N. Taft, "Communication-efficient online detection of network-wide anomalies," in *INFOCOM*, 2007.

[15] R. Keralapura, G. Cormode, and J. Ramamirtham, "Communication-efficient distributed monitoring of thresholded counts," in *SIGMOD Conference*, 2006, pp. 289–300.

[16] C. Olston, J. Jiang, and J. Widom, "Adaptive filters for continuous queries over distributed data streams," in *SIGMOD Conference*, 2003, pp. 563–574.

[17] F. McSherry and R. Mahajan, "Differentially-private network trace analysis," *SIGCOMM Computer Communication Review*, vol. 40, no. 4, pp. 123–134, Aug. 2010. [Online]. Available: http://doi.acm.org/10.1145/1851275.1851199

[18] F. K. Dankar and K. El Emam, "The application of differential privacy to health data," in *Proceedings of the 2012 Joint EDBT/ICDT Workshops*, ser. EDBT-ICDT '12. New York, NY, USA: ACM, 2012, pp. 158–166. [Online]. Available: http://doi.acm.org/10.1145/2320765.2320816

[19] F. Kargl, A. Friedman, and R. Boreli, "Differential privacy in intelligent transportation systems," in *Proceedings of the Sixth ACM Conference on Security and Privacy in Wireless and Mobile Networks*, ser. WiSec '13. New York, NY, USA: ACM, 2013, pp. 107–112. [Online]. Available: http://doi.acm.org/10.1145/2462096.2462114

[20] J. Reed, A. J. Aviv, D. Wagner, A. Haeberlen, B. C. Pierce, and J. M. Smith, "Differential privacy for collaborative security," in *Proceedings of the Third European Workshop on System Security*, ser. EUROSEC '10. New York, NY, USA: ACM, 2010, pp. 1–7. [Online]. Available: http://doi.acm.org/10.1145/1752046.1752047

[21] C. Dwork, K. Kenthapadi, F. Mcsherry, I. Mironov, and M. Naor, "Our data, ourselves: Privacy via distributed noise generation," in *EUROCRYPT*. Springer, 2006, pp. 486–503.

[22] A. Beimel, K. Nissim, and E. Omri, "Distributed private data analysis: Simultaneously solving how and what," in *Proceedings of the 28th Annual Conference on Cryptology: Advances in Cryptology*, ser. CRYPTO 2008. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 451–468.

[23] R. Chen, A. Reznichenko, P. Francis, and J. Gehrke, "Towards statistical queries over distributed private user data," in *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, ser. NSDI'12. Berkeley, CA, USA: USENIX Association, 2012, pp. 13–13. [Online]. Available: http://dl.acm.org/citation.cfm?id=2228298.2228316

[24] V. Rastogi and S. Nath, "Differentially private aggregation of distributed time-series with transformation and encryption," in *SIGMOD*, 2010, pp. 735–746.

[25] E. Shi, T.-H. H. Chan, E. G. Rieffel, R. Chow, and D. Song, "Privacy-preserving aggregation of time-series data," in *NDSS*, 2011.

[26] T. H. H. Chan, M. Li, E. Shi, and W. Xu, "Differentially private continual monitoring of heavy hitters from distributed streams," in *Privacy Enhancing Technologies*, 2012, pp. 140–159.

[27] I. Sharfman, A. Schuster, and D. Keren, "Shape sensitive geometric monitoring," in *Proceedings of the Twenty-seventh ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, ser.

PODS '08. New York, NY, USA: ACM, 2008, pp. 301–310. [Online]. Available: http://doi.acm.org/10.1145/1376916.1376958

[28] D. Keren, I. Sharfman, A. Schuster, and A. Livne, "Shape sensitive geometric monitoring," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 8, pp. 1520–1535, 2012.

[29] S. Burdakis and A. Deligiannakis, "Detecting outliers in sensor networks using the geometric approach," in *ICDE*, 2012, pp. 1108–1119.

[30] A. Gupta, A. Roth, and J. Ullman, "Iterative constructions and private data release," in *TCC*, 2012, pp. 339–356.

[31] F. McSherry and K. Talwar, "Mechanism design via differential privacy," in *FOCS*, 2007, pp. 94–103.

[32] I. Sharfman, A. Schuster, and D. Keren, "A geometric approach to monitoring threshold functions over distributed data streams," *ACM Transactions on Database Systems (TODS)*, vol. 32, no. 4, Nov. 2007. [Online]. Available: http://doi.acm.org/10.1145/1292609.1292613

[33] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li, "Rcv1: A new benchmark collection for text categorization research," *Journal of Machine Learning Research*, vol. 5, pp. 361–397, Dec. 2004. [Online]. Available: http://dl.acm.org/citation.cfm?id=1005332.1005345

[34] N. Giatrakos, A. Deligiannakis, M. Garofalakis, I. Sharfman, and A. Schuster, "Prediction-based geometric monitoring over distributed data streams," in *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '12. New York, NY, USA: ACM, 2012, pp. 265–276. [Online]. Available: http://doi.acm.org/10.1145/2213836.2213867

[35] D. Keren, G. Sagy, A. Abboud, D. Ben-David, A. Schuster, I. Sharfman, and A. Deligiannakis, "Geometric monitoring of heterogeneous streams," *IEEE Transactions on Knowledge and Data Engineering*, 2014.