PrivStream: Differentially Private Event Detection on Data Streams

Maryam Fanaeepour

Duke University

Department of Computer Science

maryam@cs.duke.edu

Ashwin Machanavajjhala

Duke University

Department of Computer Science

ashwin@sc.duke.edu

ABSTRACT

Event monitoring and detection in real-time systems is crucial. Protecting users' data while reporting an event in almost real-time will increase the level of this challenge. In this work, we adopt the strong notion of differential privacy to private stream counting for event detection with the aim of minimizing false positive and false negative rates as our utility metrics.

CCS CONCEPTS

• **Information systems** → **Data streams**; • **Security and privacy** → **Privacy-preserving protocols**; **Data anonymization and sanitization**; • **Theory of computation** → *Theory of data-base privacy and security*.

KEYWORDS

Data Stream, Differential Privacy, Event Detection, Real-time Analytics

ACM Reference Format:

Maryam Fanaeepour and Ashwin Machanavajjhala. 2019. PrivStream: Differentially Private Event Detection on Data Streams. In *Ninth ACM Conference on Data and Application Security and Privacy (CODASPY '19)*, March 25–27, 2019, Richardson, TX, USA. ACM, New York, NY, USA, 3 pages. <https://doi.org/10.1145/3292006.3302379>

1 INTRODUCTION

Today is the era of fast data stream generation coming from various sources such as sensor data, the Internet of Things, telecommunication data, social media, health care, epidemics & disasters and so on. In the settings of data streaming which is continuously generated data, real-time event analytics and query response is vital, in scenarios such as real-time fraud analysis and detection such as in Uber¹, improving traffic safety and efficiency, e.g., TFL London Transport Management System², time sensitive health care monitoring. This indeed triggers privacy concerns for the individuals sharing their information. One of the governance challenges in these applications is the ownership of the data, and how rights and obligations will be respected and forced. According to an Accenture study, 57

¹<https://eng.uber.com/fraud-prevention-team-profile/>

²<https://wso2.com/blogs/thetechsource/2017/10/a-smarter-transport-management-system-for-london-with-the-help-of-wso2/>

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

CODASPY '19, March 25–27, 2019, Richardson, TX, USA

© 2019 Copyright held by the owner/author(s).

ACM ISBN 978-1-4503-6099-9/19/03.

<https://doi.org/10.1145/3292006.3302379>

Figure 1: Data flow in private streaming for event detection.

percent of respondents were willing to share information if they are assured their data would not be sold or shared. 56 percent of consumers wanted guarantees that data protection safeguards were in place before they would share information [7].

Consider a scenario for an event detection, where a system analyst wants to start collecting evidence, keep monitoring the system, and tracking some functions e.g., aggregates. At the time that aggregate satisfies some conditions, the analyst wants to be notified so that they can fix the system and start monitoring again. Which aggregates and what conditions does an analyst want to track? What is the window time that data can be processed? What is the delay measure? What is the privacy budget? These are the questions that need to be addressed.

One use-case to consider is the occupancy level for specific rooms in a building. An event of interest is when a room's occupancy meets a defined threshold, then an alert needs to be sent to the system. However, due to privacy reasons, individuals' data can not be touched. The goal in this paper is to design algorithms that not only analyze the data streams in real-time for the purpose of *event detection*, the events' privacy can also be guaranteed through the strong notion of differential privacy (DP) framework [4].

There are works in the setting of private streaming [1, 3, 5] outputting accumulated sum of counts for the continual monitoring stream data. There is also a work that reports counts using a data dependant approach that uses the data distribution to provide a better utility [2]. There has also been some attempts using local differential privacy [9] to sanitize the data on the user side before sharing it with a service provider, but the goal in this work is to design differentially private algorithms for an event detection system that receives the flow of data stream, and event queries. Figure 1 illustrates the data flow for event detection in a private stream analytics system.

We introduce a query type for event monitoring that can vary in determined time period for running queries and defined predicates to apply (see Section 2). Our proposed algorithms aim to minimize the *false positive* (FP) and *false negative* (FN) rates as our utility metrics depending on the error bounds, to guarantee that w.h.p. an event is detected. To address the challenge of utility in real-time analytics, we propose the notion of time delay to be able to have some level of certainty before reporting an event with privacy guarantee, minimizing FP and FN both at the same time. We also discuss approaches to run event queries in parallel reducing the privacy budget used.

2 PRIVATE EVENT DETECTION OVER DATA STREAMS

2.1 Definitions

The goal in this work to preserve *event-level DP* defined as follows.

Definition 2.1 (*event-level ϵ -differential privacy* [3, 5]). A randomized mechanism $\mathcal{M}$, that takes as input an arbitrary size of stream prefix, is considered to preserve *event-level ϵ -differential privacy* for $\epsilon > 0$, if for all adjacent stream prefixes S_t, S'_t , all timestamps t and all possible set of outputs $\mathcal{O} \subseteq \text{Range}(\mathcal{M})$, the following holds $P(\mathcal{M}(S_t) \in \mathcal{O}) \leq \exp(\epsilon) \cdot P(\mathcal{M}(S'_t) \in \mathcal{O})$.

A mechanism that protects any single event preserves event-level privacy.

2.2 Event Detection

As shown in Figure 1, there is a set of event query monitoring the data stream for certain events, A_i , and our system is outputting Yes/No. A query takes stream of input over a window of time and computes a function, considering a specific event, e.g., whether $q(A_i) \geq \theta$ to then output True or False based on the threshold. The input to our event detection algorithms are as follows (cf. Figure 1).

- A query q or a set of queries Q , which includes:
  - Events of interest, A_i
  - Processing window of time $w = (t_{start}, t_{end})$
- Condition threshold θ
- Privacy budget ϵ
- Time Delay τ

In the setting of *event detection*, various queries could be considered based on the defined events and determined time window: $q(A_i, w)$. The focus of this paper is on queries in the form of Query 1 as follows. They are counting how many times the defined predicate becomes True on an incoming stream, and once the number of met condition reach the threshold report an event.

Query 1: Aggregate function (e.g., Count) on a time window starting from time zero, up until now or a specified end time $w = (0, t)$ and with predicates $q(A_i) \geq \text{or } \leq \theta$;

Event Time vs Report Time (Time Delay τ): The time an event occurs vs the time that is reported is crucial in real-time analytics. To this end, we define a delay parameter to measure the accuracy of a reported event, whether it is reported within a time delay.

What are the mistakes that a system can make? A system outputs an alert even though the condition has not been met yet (system reports the condition is true before an event occurs); or a system

notifies an analyst after the condition has been met; or the condition is true but the system does not notify at all.

2.3 Evaluation Metrics

We consider the following metrics to evaluate solutions to our problem.

- (1) **Utility:** The two error measures that we consider are *false positive* (FP) and *false negative* (FN) rates with considering time delay τ . $\text{FP}(\tau)$ happens when the time of detection is τ time units before the actual time t^* the event occurs. $\text{FN}(\tau)$ happens when the time of detection is τ time units after the actual time the event occurs (or does not report the event). We use FP and FN to denote $\text{FP}(0)$ and $\text{FN}(0)$ respectively.
- (2) **Privacy:** Mechanisms should achieve *event-level ϵ -differential privacy*, at some ϵ in their release of output.

2.4 Problem Statement

We wish to address Problem 1 based on Query 1 (cf. Section 2.2).

PROBLEM 1. Given an input of data streams, let A be a set of events, $\{A_1, A_2, \dots, A_d\}$, that our system is monitoring over time window $(0, t)$. Consider an event query or a sequence of queries $(q(A_1), (0, t)), (q(A_2), (0, t)), \dots$, that are outputting Yes/No for queries in the type of Query 1: $q(A_i) \geq \theta$; Given a bounded privacy budget ϵ , and a defined threshold θ , our goal is to design an algorithm that privately releases $(A_1, t'_1), (A_2, t'_2), \dots$ [within a specified time delay τ], that w.h.p. detects an event or a series of events in real-time with regard to the evaluation metrics.

3 BINARY MECHANISM FOR EVENT DETECTION

Our baseline approach is adopted from Binary Mechanism [1, 5] proposed to save the amount of privacy budget used. We call this algorithm as *Binary Mechanism for Event Detection*. The input to Binary Mechanism (BM) is a time upper bound T ; a privacy parameter ϵ ; and a stream $\sigma\{0, 1\}^T$ (i.e., sequence of $q_i(t)$ in the time period of $(0, T)$). BM outputs estimate of count at each point of time. Here, we call BM and then check if a condition for an event query meets a threshold given the computed error bound for BM. Algorithm 1 runs a single event query to notify an analyst when an event occurs.

Algorithm 1: Binary Mechanism for Event Detection

Input : Data Streams S ; Query of interest $q(A, (0, T))$;
Threshold θ ; Privacy budget ϵ ; Error bound α

Output: Detected time for an event of interest

/* Estimate the count of q calling **BM**, that estimates $\hat{q}$ at each point of time $t \in (0, T)$ as $\hat{q}(t)$ */

```

1 BM ( $T, \epsilon, q$ )
2 if  $\hat{q}(t) + \alpha \geq \theta$  then
3    $t_0 \leftarrow t // t_0$  as the time that event is
   detected using noisy counts
4   Report event  $A$  at time  $t_0$ 

```

THEOREM 3.1. Algorithm 1 achieves $T\epsilon$ -differential privacy.

THEOREM 3.2 (UTILITY). For confidence level $\delta \in (0, 1)$ and $t \in [T]$, Algorithm 1 is (α, δ) -accurate w.h.p. at least $1 - \delta$: when $\alpha = -(\sqrt{8 \log 2/\delta} \cdot \log T \sqrt{\log t \log 1/\delta})/\epsilon$ then it ensures that $P(FP) < \delta$; when $\alpha = +(\sqrt{8 \log 2/\delta} \cdot \log T \sqrt{\log t \log 1/\delta})/\epsilon$ then $P(FN) < \delta$.

PROOF. The final error bound follows by applying Theorem 3.6 [1], and the fact that the scale of the Laplace mechanism is the same. Then according to Corollary 2.9 [1], the sum of our random variables is at most $(\sqrt{\log t} \cdot \lambda \cdot \log 1/\delta)/\epsilon$: $\alpha = O(1/\epsilon \cdot \log T \cdot \sqrt{\log t \log 1/\delta})$, where the constant is $\sqrt{8 \log 2/\delta}$. $\square$

4 PRIVATE STREAM COUNTING MECHANISM FOR EVENT DETECTION USING SVT

Algorithm 1 reports estimated count at each point of time, and as a result consumes unnecessary privacy budget. In the case of event detection, the only time that an estimated count needs to be considered but not reported is when it meets a certain threshold. To improve Algorithm 1, we use sparse vector technique (SVT) [8] as described in Algorithm 2. For the case of event monitoring and detection, we only need to report the event once, and then the privacy budget gets refreshed as the system analyst needs to take an action. Therefore, the cut-off point parameter c is 1, and no need to keep track of it. As a result, the Laplace parameter for the next step to add noise to c will be $2\Delta/\epsilon_2$ instead of $2c\Delta/\epsilon_2$. In addition, it is suggested to apply Δ/ϵ_2 instead of $2\Delta/\epsilon_2$, when the original query function is monotonic. In terms of budget allocation for each separate privacy budget, we could do $1 : 1$ or as suggested by authors in [8] $1 : (c)^{2/3}$ for monotonic case, which will be again $1 : 1$ in our case as c is 1.

Algorithm 2: Private Stream Counting Mechanism for Event Detection using SVT

Input : Data Streams S ; Query of interest $q(A, (0, T))$; Threshold θ ; Privacy budget ϵ ; Error Bound α
Output: Detected time for an event of interest

// Initialize the privacy budget

```

1  $\epsilon_1 = \epsilon_2 = \epsilon/2$ 
2  $\hat{\theta} = \theta + \text{Lap}(1/\epsilon_1)$ 
3 for  $t \in (0, T)$  do
4   if  $q(t) + \alpha + \text{Lap}(1/\epsilon_2) \geq \hat{\theta}$  then
5      $t_0 \leftarrow t$  //  $t_0$  as the time that event is
      detected using noisy counts
6   Report event  $A$  at time  $t_0$ 
```

THEOREM 4.1. Algorithm 2 achieves ϵ -differential privacy [8].

THEOREM 4.2 (UTILITY). For confidence level $\delta \in (0, 1)$, Algorithm 2 is (α, δ) -accurate, w.h.p. at least $1 - \delta$, when $\alpha = -(\frac{8(\log t + \log(2/\delta))}{\epsilon})$ then Algorithm 2 guarantees that $P(FP) < \delta$; when $\alpha = +(\frac{8(\log t + \log(2/\delta))}{\epsilon})$ then $P(FN) < \delta$.

PROOF. Proof follows by similar ways in Theorem 3.24 [6]. $\square$

Comparing the error bounds for Algorithms 1 and 2, Algorithm 2 provides a smaller bound and as a result a better utility.

FP or FN with time delay τ . Ideally we wish to have a bound on both FP and FN at the same time, therefore we consider the utility metric with the notion of time delay.

THEOREM 4.3 (UTILITY WITH TIME DELAY τ). For confidence level $\delta \in (0, 1)$ and $t \in [T]$, a DP algorithm is (α, δ) -accurate w.h.p. at least $1 - \delta$, such that we can guarantee that $P(FP) < \delta$ when an event is detected at time $t_0 < t^* - \tau$, given the true event occurs at t^* , and $P(FN) < \delta$ when an event is detected at time $t_0 > t^* + \tau$. If we define a function of time for data stream rate, $\Gamma(\tau)$ for the increases in $q(t)$, i.e., in τ time, $q(t)$ increases by $\Gamma(\tau)$ w.h.p. δ . Then we will have the following changes in our decision rules:

- (a) if $\hat{q}(t) - (\alpha - \Gamma(\tau)) \geq \theta \Rightarrow P(\text{FP with delay}) < 2\delta$
- (b) if $\hat{q}(t) + (\alpha - \Gamma(\tau)) \geq \theta \Rightarrow P(\text{FN with delay}) < 2\delta$

If we set α to be equal to $\Gamma(\tau)$ then both rules become the same and we will have a bound on both FP and FN at the same time.

REMARK 1. If we are to run a set of event queries, in the worst case, it will follow the sequential mechanism and provides d -DP, d as the number of queries. However, our algorithm could perform better if event queries are in the form of vector presentation. Then, for the BM we can assume that if we have a set of queries represented as a workload matrix w , then the sensitivity of a workload, which is represented as a matrix of 0 and 1's, is the maximum column norm, each column is one entry for every possible value for the domain. A column with the largest number of 1's determines the largest sensitivity, then our algorithm achieves $\max(\Delta_i)\epsilon$ -DP. Therefore, we only need to scale the BM to the max sensitivity of the workload.

5 CONCLUDING REMARKS

In this work, we propose private stream counting algorithms for event detection using binary mechanism and sparse vector technique. This is an initiation for differentially private event detection aiming for minimizing false positive rate and/or false negative rate, depending on which metrics users decide to apply.

Acknowledgements: This work was supported by the National Science Foundation under grants 1253327, 1408982; and by DARPA and SPAWAR under contract N66001-15-C-4067.

REFERENCES

- [1] T.-H. Hubert Chan, Elaine Shi, and Dawn Song. 2010. Private and Continual Release of Statistics. In *ICALP*. 405–417.
- [2] Yan Chen, Ashwin Machanavajjhala, Michael Hay, and Gerome Miklau. 2017. PeGaSus: Data-Adaptive Differentially Private Stream Processing. In *CCS*. 1375–1388.
- [3] Cynthia Dwork. 2010. Differential Privacy in New Settings. In *SODA*. 174–183.
- [4] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating Noise to Sensitivity in Private Data Analysis. In *TCC'06*. 265–284.
- [5] Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N. Rothblum. 2010. Differential privacy under continual observation. In *STOC*. 715–724.
- [6] Cynthia Dwork and Aaron Roth. 2014. The Algorithmic Foundations of Differential Privacy. *Foundations and Trends in Theoretical Computer Science* 9, 3-4 (2014), 211–407.
- [7] Priya Keshav and Jason R. Baron. 2018. INSIGHT: The Internet of Things and Information Governance: What You Need To Know To Manage Data Streaming From Everywhere. <https://www.bna.com/insight-internet-things-n73014476349/>
- [8] Min Lyu, Dong Su, and Ninghui Li. 2017. Understanding the Sparse Vector Technique for Differential Privacy. *PVLDB* 10, 6 (2017), 637–648.
- [9] Do Le Quoc, Martin Beck, Pramod Bhatotia, Ruichuan Chen, Christof Fetzer, and Thorsten Strufe. 2017. PrivApprox: Privacy-Preserving Stream Analytics. In *USENIX*. 659–672.