Differentially Private Event Sequences over Infinite Streams

Georgios Kellaris[1] Stavros Papadopoulos[2]* Xiaokui Xiao[3] Dimitris Papadias[1]

[1]HKUST
{gkellaris, dimitris}@cse.ust.hk

[2]Intel Labs / MIT
stavrosp@csail.mit.edu

[3]Nanyang Technological University
xkxiao@ntu.edu.sg

ABSTRACT

Numerous applications require *continuous* publication of statistics for monitoring purposes, such as real-time traffic analysis, timely disease outbreak discovery, and social trends observation. These statistics may be derived from sensitive user data and, hence, necessitate privacy preservation. A notable paradigm for offering strong privacy guarantees in statistics publishing is ϵ-differential privacy. However, there is limited literature that adapts this concept to settings where the statistics are computed over an *infinite stream* of "events" (i.e., data items generated by the users), and published periodically. These works aim at hiding a *single* event over the entire stream. We argue that, in most practical scenarios, sensitive information is revealed from *multiple* events occurring at *contiguous* time instances. Towards this end, we put forth the novel notion of w-*event privacy* over infinite streams, which protects any *event sequence* occurring in w successive time instants. We first formulate our privacy concept, motivate its importance, and introduce a methodology for achieving it. We next design two instantiations, whose utility is independent of the stream length. Finally, we confirm the practicality of our solutions experimenting with real data.

1. INTRODUCTION

There is a large number of applications that benefit from the *continuous* monitoring of statistics. For example, traffic services publish the number of cars per area in real-time to allow for optimal route computation [2]. Moreover, hospitals periodically release the number of patients suffering from certain diseases [1], which may help in the timely discovery of disease outbreaks. In addition, social networks constantly report the number of users currently talking about a topic [3], which enables listing the hot topics for targeted advertising. However, the inadvertent disclosure of such statistics may compromise the privacy of the individuals, such as the locations a commuter visits, the type of illness a patient suffers from, and the political views a user communicates.

A popular paradigm for providing privacy in statistics publishing with strong theoretical guarantees is ϵ-*differential privacy* [10]. This framework entails perturbing the data prior to their release, in order to hide sensitive information about the individuals that participate in the statistical analysis. Recently, this notion has been applied to streaming scenarios [11, 16], where statistics are continuously published at certain timestamps. These statistics are computed over user *events*, i.e., "actions" taken by users at specific timestamps that contribute to the published data. For instance, assume that a traffic service periodically publishes the count of commuters per location. Then, the presence of a commuter at a specific location is an event occurring at a certain timestamp.

There exist two definitions of differential privacy in such settings; *event-level* and *user-level* [16]. The former protects any *single* event, whereas the latter hides *all* the events of any user throughout the entire stream. For example, event-level privacy protects a single location visit, whereas user-level privacy protects all the location visits of any commuter. The privacy level affects the amount of perturbation used, which is proportional to the contribution of the sensitive information to the statistics.

Prior work on differential privacy on streams mainly focuses on event-level privacy on *infinite* streams [11, 27, 6, 7, 4], and user-level privacy on *finite* streams [17]. The first category is not useful in realistic scenarios where users cause events in *contiguous* timestamps, which *collectively* disclose sensitive information. For example, although event-level privacy protects any single location visit, it does not protect a user *path* traversed in successive timestamps. The second category has limited applicability in most practical settings, where data must be published indefinitely. For instance, it is restrictive to assume that a traffic reporting service shuts down after an a priori known time interval. On the other hand, offering user-level privacy over infinite streams requires infinite amount of perturbation, which destroys the utility of the data in the long run.

A novel problem. In this paper we merge the important gap between event-level and user-level privacy in streaming settings. In particular, we propose w-*event privacy*, which protects any event sequence occurring within any window of w timestamps. This novel privacy definition is useful in numerous applications. For instance, it can protect any temporally constrained movement trajectory in traffic services, patient hospitalization, or succession of related topics discussed by a user.

w-event privacy captures a superset of the applications of event-level privacy, whereas for $w = 1$ the two definitions become equivalent. However, it is narrower than user-level privacy, since it does not hide multiple event sequences from the same user. Setting w to infinity, w-event privacy converges to user-level privacy, inheriting though the need for infinite perturbation and utility degradation over time. In that sense, our definition strikes a nice balance between practicality and privacy, which expands the machinery for differential privacy in infinite streams. The challenge lies in its rigorous formulation and the design of effective schemes to satisfy it.

*This work was done while the author was at HKUST.

Proceedings of the VLDB Endowment, Vol. 7, No. 12

Contributions. We first formulate w-event privacy, and explain that it gives rise to a new class of mechanisms. Next, we present a *sliding window* methodology that captures a wide range of w-event private mechanisms. A mechanism following our methodology constructs a separate sub mechanism per timestamp, each spending a certain *privacy budget* that controls the perturbation (the higher the budget, the lower the perturbation). Then, w-event privacy is attained when the sum of budgets of the mechanisms inside *any* window of w timestamps is at most the total privacy budget ϵ. In the absence of previous work on this problem, we devise three benchmarks adapting ideas from existing schemes. Their main weakness is that they poorly allocate the budget across the event sequences, which results in a low overall utility. This motivates the need for new, flexible, dynamic budget allocation schemes.

Towards this end, we introduce two novel schemes tailored to our proposed methodology, called *Budget Distribution* (BD) and *Budget Absorption* (BA). These mechanisms effectively allocate the budget relying on the fact that the statistics may not change significantly in successive timestamps. Specifically, based on a private sub mechanism that calculates the *dissimilarity* between statistics, they *skip* the publications that can be accurately *approximated* by the last publication. Skipping a publication implies that no budget is spent at that timestamp, which becomes available for a future publication. BD and BA differ in the way they *dynamically* allocate the available privacy budget over time. BD starts with the entire available budget and *distributes* it in an exponentially decreasing fashion as publications occur, while reusing the budget used at older timestamps. On the other hand, BA *uniformly* allocates the initial budget over the timestamps at first, but *absorbs* budgets that became available from previously skipped publications. We include a utility analysis for our schemes, and propose optimizations. Finally, we experimentally evaluate them on real datasets, and confirm their effectiveness.

2. BACKGROUND

This section surveys the related work, providing the preliminary definitions and theorems that are necessary for the rest of the presentation. Section 2.1 explains ϵ-differential privacy, whereas Section 2.2 describes this notion in scenarios where statistics are continuously published over time in the form of streams.

2.1 Differential Privacy

In the *ϵ-differential privacy* paradigm, a *trusted curator* gathers potentially sensitive data from a large number of respondents, creating a database D. The goal is to publish statistics computed on D, without compromising the privacy of the respondents. The curator achieves this goal by *randomly perturbing* the statistics, such that (i) the presence of any individual in D is hidden, and (ii) the published data have high *utility*, i.e., the perturbation does not greatly affect their accuracy with respect to the actual statistics.

Formally, let $\mathcal{D}$ denote a set of databases. Each $D \in \mathcal{D}$ is perceived as a set of rows. Two databases $D, D' \in \mathcal{D}$ are *neighboring* if we can obtain D' from D by removing or adding a single row. Let $\mathcal{M}$ be a randomized mechanism, which takes as input a database from $\mathcal{D}$ and outputs a *transcript* in some range $\mathcal{O}$. The curator runs $\mathcal{M}$ on $D \in \mathcal{D}$, and publishes $\mathbf{o} \in \mathcal{O}$, i.e., $\mathbf{o}$ constitutes the released statistics about D. The ϵ-differential privacy notion is formulated as follows.

DEFINITION 1. [10] *A mechanism $\mathcal{M} : \mathcal{D} \rightarrow \mathcal{O}$ satisfies* **ϵ-differential privacy**, *where $\epsilon \geq 0$, if for all sets $O \subseteq \mathcal{O}$, and every pair $D, D' \in \mathcal{D}$ of neighboring databases*

$$\Pr[\mathcal{M}(D) \in O] \leq e^{\epsilon} \cdot \Pr[\mathcal{M}(D') \in O]$$

The above definition states that the probability that $\mathcal{M}$ produces a transcript $\mathbf{o}$ when a user is in D must be roughly the same as in the case where this user is absent from D. Hence, the adversary infers no more information from $\mathbf{o}$ about the user than in the case where the individual's data were absent from the database. The smaller the value of ϵ, the stronger the privacy guarantees.

The first technique to satisfy ϵ-differential privacy is the Laplace Perturbation Algorithm (LPA) [14, 12]. Its main idea is to add *noise* drawn from a Laplace distribution into the statistics to be published. The scale of this distribution must be proportional to the effect that a single user can have on the statistical result. The latter is called *sensitivity* and formalized as follows. We view the statistics as the result of a query on D. For example, the query may ask for the count of each column of D. We model the query as a function $\mathbf{Q} : \mathcal{D} \rightarrow \mathbb{R}^d$, where d is the number of elements in the output. Let $\|\mathbf{Q}(D) - \mathbf{Q}(D')\|_1$ denote the L_1 norm of $\mathbf{Q}(D), \mathbf{Q}(D')$. The *sensitivity* of $\mathbf{Q}$ w.r.t. $\mathcal{D}$ is $\Delta(\mathbf{Q}) = \max_{D,D' \in \mathcal{D}} \|\mathbf{Q}(D) - \mathbf{Q}(D')\|_1$ for all neighboring $D, D' \in \mathcal{D}$ [14].

Let $Lap(\lambda)$ be a random value drawn from a Laplace distribution with mean zero and scale λ. LPA achieves ϵ-differential privacy through the mechanism outlined in the following theorem.

THEOREM 1. [14] *Let $\mathbf{Q} : \mathcal{D} \rightarrow \mathbb{R}^d$. A mechanism $\mathcal{M}$ that adds independently generated noise from a zero-mean Laplace distribution with scale $\lambda = \Delta(\mathbf{Q})/\epsilon$ to each of the d output values of $\mathbf{Q}(D)$, i.e., which produces $\mathbf{o} = \mathbf{Q}(D) + \langle Lap(\Delta(\mathbf{Q})/\epsilon)\rangle^d$ satisfies ϵ-differential privacy.*

The higher the $\Delta(\mathbf{Q})$ or the smaller the ϵ, the larger the required noise for achieving ϵ-differential privacy. This is justified since (i) the larger the effect of any user on the result, the larger the amount of noise needed to "hide" the user, and (ii) the stronger the privacy guarantees, the more independent the $\mathcal{M}$'s output should be with respect to the input, which mandates larger noise.

Although LPA is a general methodology for ϵ-differential privacy, there are scenarios in which the required noise is prohibitively large, and destroys the utility of the published statistics. Therefore, many approaches have been proposed in order to improve upon the LPA algorithm on specific settings. The Fourier Perturbation Algorithm (FPA) [29] focuses on time-series data. Roth and Roughgarden [30] propose the *median mechanism* for an interactive setting, where the adversary requests statistics about D multiple times in an adaptive fashion. There are also works that focus on other specialized scenarios, such as range-count queries [32, 8], query consistency [19], sparse data [9, 23], correlated queries [18, 21], frequent itemsets [33, 22], trajectories [20], minimization of relative error [31], and non-numerical query answers [26]. PINQ [25] is a system implementation that integrates differential privacy with data analysis. Finally, there are several relaxations of ϵ-differential privacy [13, 24, 28].

We include a *composition* theorem that is useful for our proofs. It concerns successive executions of differentially private mechanisms on the same input.

THEOREM 2. [25] *Let $\mathcal{M}_1, \ldots, \mathcal{M}_r$ be a set of mechanisms, where $\mathcal{M}_i$ provides ϵ_i-differential privacy. Let $\mathcal{M}$ be another mechanism that executes $\mathcal{M}_1(D), \ldots, \mathcal{M}_r(D)$ using independent randomness for each $\mathcal{M}_i$, and returns the vector of the outputs of these mechanisms. Then, $\mathcal{M}$ satisfies $\left(\sum_{i=1}^{r} \epsilon_i\right)$-differential privacy.*

The above theorem allows us to view ϵ as a *privacy budget* that is distributed among the r mechanisms. Moreover, note that the theorem holds even when $\mathcal{M}_i$ receives as input the private outputs of $\mathcal{M}_1, \ldots, \mathcal{M}_{i-1}$ [25].

2.2 Differential Privacy on Streams

So far we have focused on statistics that are derived from a *static* dataset. In this section, we describe how ϵ-differential privacy applies to settings where the dataset is *dynamically* updated by a stream, and the curator continuously publishes new statistics capturing the updates. In such a streaming scenario, every user is associated with *events*, i.e., "actions" taken by the user, which are modeled as update items in the stream that contribute to the new statistics. There are typically two different privacy goals in the literature, namely *event-level* and *user-level*. The former hides a *single* event, whereas the latter hides *all* the events of any user (thus concealing completely the presence of the user in the analysis).

The two works that initiated the study on this setting are [11, 16]. They view a data stream as a sequence of symbols drawn from a domain X, where each symbol represents a user's event (e.g., $x \in X$ may refer to event "*user u visited location j*"). The stream is essentially an array S, where element $S[1]$ corresponds to the first event that occurred, and new events/symbols are appended at the end of S as time elapses. If the stream S is infinite, we use S_t to denote the *prefix* of S after t updates, i.e., $S_t = (S[1], \ldots, S[t])$.

To model privacy, [16] defines the notion of adjacency between stream prefixes, which is similar to the concept of neighboring databases in the static case. Specifically, stream prefixes S_t, S'_t are *event-level adjacent*, if there exists *at most one* i such that $S_t[i] \neq S'_t[i]$, i.e., if they differ in at most one symbol. Moreover, S_t, S'_t are *user-level adjacent* if, for any user u, it holds that $S_t[i] \neq S'_t[i]$ for *any number of* $S_t[i]$'s corresponding to u's events, i.e., if they differ in any number of symbols belonging to u. *Differential privacy under continual observation* is defined as follows.

DEFINITION 2. [11, 16] *Let $\mathcal{M}$ be a mechanism that takes as input a stream prefix of arbitrary size. Let $\mathcal{O}$ be the set of all possible outputs of $\mathcal{M}$. Then, $\mathcal{M}$ is event-level (user-level) ϵ-differentially private if for all sets $O \subseteq \mathcal{O}$, all event-level (resp. user-level) adjacent stream prefixes S_t, S'_t, and all t, it holds that*

$$\Pr[\mathcal{M}(S_t) \in O] \leq e^{\epsilon} \cdot \Pr[\mathcal{M}(S'_t) \in O]$$

Most existing schemes focus on event-level privacy in publishing *counters*, i.e., in reporting at every timestamp the number of event occurrences since the commencement of the system. In such scenarios, the effect of a single event spans over all subsequent publications. The schemes view the data stream as a bitstring, and at each timestamp they publish the number of 1's seen so far. Upon the arrival of an update, Dwork [11] adds to the counter the update value along with Laplace noise of scale $1/\epsilon$. The author states that this works well in dense (in terms of 1's) streams, but it may be ineffective in sparse cases. To remedy this, she also proposes to postpone publishing a new counter value, until a predefined number of 1's has been seen. [11] also deals with other counting problems, such as density estimation, cropped mean, and heavy hitters.

Dwork et al. [16] improve upon the counter publishing schemes of [11]. They focus on a stream of *fixed* length T, and build a full binary tree over the updates of the stream. Every node stores a noise value with scale logarithmic in T. Then, at the i^{th} update, they identify the subtrees it belongs to, and report the current counter after adding the noise values stored in the roots of these subtrees. A similar result appeared independently in [6]. The proposed scheme constructs a full binary tree on the updates, where each node contains the sum of the updates in its subtree, plus noise with scale logarithmic in T. At each update i, it identifies the maximal subtrees covering updates 1 to i, and reports the sum of the values stored in their roots. The authors extend their work to infinite streams, by constructing a new binary tree after seeing 2^{κ} updates, for $\kappa \in \mathbb{N}$.

Bolot et al. [4] extend [6] by proposing decayed sums, which emphasize on recent data more than the data of the past. Moreover, they present the new notion of decayed privacy, where past data are not viewed as private anymore and, thus, they can be used in their raw form. Mir et al. [27] consider counters that may also decrease. They improve upon the counting schemes of [11] by utilizing sketches. Chan et al. [7] report the heavy hitters instead of the counter values in various settings, such as sliding windows and multiple streams with untrusted aggregators.

There are also two schemes that depart from the counter-based setting. Cao et al. [5] specify a set of range queries on the time domain before the system starts. Each query requests the sum of the updates therein. The goal is to determine the optimal strategy for answering all the queries, by responding to larger ranges using the noisy answers of smaller subranges. Fan and Xiong [17] report counts at every timestamp, focusing on user-level privacy. They assume a finite stream, and use sampling to reduce the noise; given a pre-specified number of samples, they choose a subset of timestamps to publish the data, in a way that reduces the total error.

3. PROBLEM DEFINITION

We introduce a new privacy definition, motivate its importance, and propose a novel methodology for achieving it. In the absence of direct competitors in the literature, we also devise benchmark schemes by adapting work proposed in different scenarios.

Setting and privacy goal. The curator receives updates from an *infinite* data stream S at discrete timestamps. At every timestamp i, the curator collects a database D_i with d columns and an arbitrary number of rows, where every row corresponds to a *unique* user, and every column represents an event. The value in row u and column j is 1 if event j of user u has occurred at i, and 0 otherwise. Every row contains *at most one* 1. We model the stream as an infinite tuple $S = (D_1, D_2, \ldots)$, where $S[i]$ is the i^{th} element of S. We define a stream prefix of S at t as tuple $S_t = (D_1, D_2, \ldots, D_t)$. At each timestamp i, the curator wishes to publish the *count* of every column in D_i (i.e., the number of non-zero elements in each column). Let a count query be $\mathbf{Q} : \mathcal{D} \rightarrow \mathbb{R}^d$, where $\mathcal{D}$ is the set of all databases with d columns. Then, $\mathbf{Q}(S[i]) = \mathbf{Q}(D_i) = \mathbf{c}_i$ is the data to be released at i, where $\mathbf{c}_i[j]$ is the count of column j of D_i. Hence, the curator generates infinite stream $(\mathbf{c}_1, \mathbf{c}_2, \ldots)$ while collecting S. Note that the sensitivity of $\mathbf{Q}$ is $\Delta(\mathbf{Q}) = 1$.

We justify our model with an example. Suppose that a curator collects at timestamp i a database D_i from a traffic reporting service. This database contains one row for every commuter, and a set of columns corresponding to locations in a road network. The value in row u and column j is 1 if commuter u is at location j at timestamp i, and 0 otherwise. Since any commuter can be at a single location at each timestamp, every row can have *at most a single* 1. The curator publishes at i the count of every column of D_i, which represents the number of commuters per location at i. This system runs indefinitely. We next explain our privacy goal.

We call two databases D_i, D'_i at timestamp i as *neighboring* if we can obtain D'_i from D_i by removing or adding a row (i.e., we use the term in the same manner as in Section 2.1). We also require a new notion of neighboring stream prefixes, defined below.

DEFINITION 3. *Let w be a positive integer. Two stream prefixes S_t, S'_t are **w-neighboring**, if*

(i) for each $S_t[i], S'_t[i]$ such that $i \in [t]$ and $S_t[i] \neq S'_t[i]$, it holds that $S_t[i], S'_t[i]$ are neighboring, and

(ii) for each $S_t[i_1], S_t[i_2], S'_t[i_1], S'_t[i_2]$ with $i_1 < i_2$, $S_t[i_1] \neq S'_t[i_1]$ and $S_t[i_2] \neq S'_t[i_2]$, it holds that $i_2 - i_1 + 1 \leq w$.

In other words, if S_t, S'_t are w-neighboring, then (i) their elements are *pairwise* the same or neighboring, and (ii) all their neighboring databases can fit in a window of *up to* w timestamps.

DEFINITION 4. *Let $\mathcal{M}$ be a mechanism that takes as input a stream prefix of arbitrary size. Also let $\mathcal{O}$ be the set of all possible outputs of $\mathcal{M}$. We say that $\mathcal{M}$ satisfies **w-event ϵ-differential privacy** (or, simply, **w-event privacy**) if for all sets $O \subseteq \mathcal{O}$, all w-neighboring stream prefixes S_t, S'_t, and all t, it holds that*

$$\Pr[\mathcal{M}(S_t) \in O] \leq e^{\epsilon} \cdot \Pr[\mathcal{M}(S'_t) \in O]$$

This definition captures settings where sensitive information is disclosed from an *event sequence* of some *finite* length w, i.e., from a set of events where (i) each event occurs at a different timestamp, and (ii) all the events occur within w consecutive timestamps. Revisiting our traffic reporting service example, an event sequence of length w may be a trajectory over a set of locations traversed within w consecutive timestamps. In this scenario, w-event privacy protects any such trajectory of any commuter anywhere in the stream.

w-event privacy expands event-level privacy [16]. If we set $w = 1$ and flatten element $S_t[i]$ so that it corresponds to a sequence of symbols, our w-neighboring definition becomes equivalent to event-level adjacency and, thus, w-event privacy degenerates to event-level privacy. For $w > 1$, w-event privacy offers stronger guarantees, at the cost of an error increase. However, we shall see that this error depends on w, but *not* on the stream length, which guarantees that utility does not degrade over time. Setting $w = t$, w-event privacy converges to user-level privacy. Hence, w-event privacy strikes a nice balance between privacy and utility.

One may argue that w-event privacy can be satisfied by any user-level private mechanism as follows. We decompose the stream prefix S_t into *disjoint* sub sequences, each of length w. We then break mechanism $\mathcal{M}$ into sub mechanisms $\mathcal{M}_i$, each corresponding to a different such sub sequence. Every $\mathcal{M}_i$ is any user-level private mechanism for *finite streams* (e.g., [17]), since we know a priori the length of its sub sequence. By Definition 3, any w-neighboring prefix S'_t will differ from S_t in at most two adjacent sub sequences. If we assign budget $\epsilon/2$ to each $\mathcal{M}_i$, the condition in Definition 4 holds and, thus, w-event privacy is satisfied.

Nevertheless, this adaptation of user-level privacy to achieve w-event privacy substantially constrains the design and effectiveness of the $\mathcal{M}_i$ instantiations. Specifically, every $\mathcal{M}_i$ acts independently and allocates budget $\epsilon/2$ to a specific sub sequence of fixed length w. This approach increases the error by a fixed factor of two for *any* sub sequence of length w in the stream. Furthermore, it hinders budget allocation optimizations for sub sequences that *span over two* sub mechanisms. In particular, the sub sequences that fall entirely in the scope of a sub mechanism may be more benefited (e.g., from adaptive sampling as in [17]) than those that are split between two sub mechanisms.

Ideally, a w-event private mechanism should achieve two goals: for *every* sub sequence of length w *anywhere* in stream S_t, it should (i) allocate up to ϵ budget, and (ii) take budget allocation decisions considering the *entirety* of the sub sequence. From the above discussion, the adaptation of user-level privacy fails to meet these goals. This suggests that w-event privacy gives rise to novel mechanisms that depart from the user-level privacy literature. Towards this end, we next propose a framework that allows the design of a large class of w-event private mechanisms, achieving the two goals.

A sliding window methodology. The main idea is to view a mechanism $\mathcal{M}$ on a stream prefix S_t as the *composition* of t mechanisms $\mathcal{M}_1, \ldots, \mathcal{M}_t$, where every $\mathcal{M}_i$ operates on $S_t[i] = D_i$ at timestamp i and uses *independent randomness*. We analyze the privacy level of every $\mathcal{M}_i$ *separately*. Let $\mathcal{M}_i$ be ϵ_i-differentially private, for some ϵ_i. In order for $\mathcal{M}$ to satisfy w-event privacy, it suffices to ensure the following condition. Let the *active window* of size w of timestamp i span from $i - w + 1$ to i. Then, the sum of $\epsilon_{i-w+1}, \ldots, \epsilon_i$ must be smaller than or equal to ϵ. Moreover, this should hold for *any* timestamp, i.e., as the active window *slides* over time. We formulate this observation in the theorem below.

THEOREM 3. *Let $\mathcal{M}$ be a mechanism that takes as input stream prefix S_t, where $S_t[i] = D_i \in \mathcal{D}$, and outputs a transcript $\mathbf{o} = (\mathbf{o}_1, \ldots, \mathbf{o}_t) \in \mathcal{O}$. Suppose that we can decompose $\mathcal{M}$ into t mechanisms $\mathcal{M}_1, \ldots, \mathcal{M}_t$, such that $\mathcal{M}_i(D_i) = \mathbf{o}_i$, each $\mathcal{M}_i$ generates independent randomness and achieves ϵ_i-differential privacy. Then, $\mathcal{M}$ satisfies w-event privacy if*

$$\forall i \in [t], \sum_{k=i-w+1}^{i} \epsilon_k \leq \epsilon \tag{1}$$

PROOF. Since all the mechanisms use independent randomness, the following holds for stream prefix S_t and any mechanism output $(\mathbf{o}_1, \ldots, \mathbf{o}_t) \in O \subseteq \mathcal{O}$:

$$\Pr[\mathcal{M}(S_t) = (\mathbf{o}_1, \ldots, \mathbf{o}_t)] = \prod_{k=1}^{t} \Pr[\mathcal{M}_k(D_k) = \mathbf{o}_k]$$

Similarly, for any w-neighboring stream prefix S'_t of S_t and the same $(\mathbf{o}_1, \ldots, \mathbf{o}_t)$, it holds:

$$\Pr[\mathcal{M}(S'_t) = (\mathbf{o}_1, \ldots, \mathbf{o}_t)] = \prod_{k=1}^{t} \Pr[\mathcal{M}_k(D'_k) = \mathbf{o}_k]$$

By Definition 3, there exists $i \in [t]$, such that $D_k = D'_k$ for $1 \leq k \leq i - w$ and $i + 1 \leq k \leq t$. Combining this with the above two equations, we get

$$\frac{\Pr[\mathcal{M}(S_t) = (\mathbf{o}_1, \ldots, \mathbf{o}_t)]}{\Pr[\mathcal{M}(S'_t) = (\mathbf{o}_1, \ldots, \mathbf{o}_t)]} = \prod_{k=i-w+1}^{i} \frac{\Pr[\mathcal{M}_k(D_k) = \mathbf{o}_k]}{\Pr[\mathcal{M}_k(D'_k) = \mathbf{o}_k]}$$

Due to Definition 3, it holds that database pairs D_k, D'_k are neighboring for $i - w + 1 \leq k \leq i$. Since $\mathcal{M}_k$ is ϵ_k-differentially private and due to Definition 1, it holds that $\frac{\Pr[\mathcal{M}_k(D_k) = \mathbf{o}_k]}{\Pr[\mathcal{M}_k(D'_k) = \mathbf{o}_k]} \leq e^{\epsilon_k}$. Thus, we derive that

$$\frac{\Pr[\mathcal{M}(S_t) = (\mathbf{o}_1, \ldots, \mathbf{o}_t)]}{\Pr[\mathcal{M}(S'_t) = (\mathbf{o}_1, \ldots, \mathbf{o}_t)]} \leq \prod_{k=i-w+1}^{i} e^{\epsilon_k} = \exp\left(\sum_{k=i-w+1}^{i} \epsilon_k\right)$$

Adding probabilities $\Pr[\mathcal{M}(S_t) = (\mathbf{o}_1, \ldots, \mathbf{o}_t)]$ over any $O \in \mathcal{O}$, we get $\frac{\Pr[\mathcal{M}(S_t) \in O]}{\Pr[\mathcal{M}(S'_t) \in O]} \leq \exp\left(\sum_{k=i-w+1}^{i} \epsilon_k\right)$. Hence, if Formula (1) holds, then we get $\frac{\Pr[\mathcal{M}(S_t) \in O]}{\Pr[\mathcal{M}(S'_t) \in O]} \leq e^{\epsilon}$ which, due to Definition 4, concludes our proof. □

Note that our theorem holds even if we provide the previous private outputs $\mathbf{o}_1, \ldots, \mathbf{o}_{i-1}$ as input to $\mathcal{M}_i$ for every i [25], as well as the previously allocated budgets $\epsilon_1, \ldots, \epsilon_{i-1}$ [12].

This theorem enables a w-event private scheme to view ϵ as the total available privacy budget in any *sliding window* of size w, and appropriately allocate portions of it across the timestamps therein. Observe that, contrary to the user-level privacy adaptation discussed before, this methodology (i) allows *any* sub sequence of length w in the stream to enjoy up to ϵ budget, and (ii) enables each $\mathcal{M}_i$ to decide on budget ϵ_i considering the budgets allocated to sub sequence $S_t[i - w + 1], \ldots, S_t[i - 1]$, and optimize budget allocation for the *entire* sub sequence $S_t[i - w + 1], \ldots, S_t[i]$.

The challenge in designing mechanisms that follow this methodology lies in the budget allocation technique, which must respect the condition of the theorem for *any* sliding window over time. This is because (i) the ϵ_i values determine the noise scale and, hence, have a direct effect on the accuracy of the outputs, and (ii) in a streaming setting, ϵ_i may need to be specified *on-the-fly* as the data arrive at the curator.

Benchmark methods. No existing scheme is directly applicable to our targeted setting. The works that aim at event-level privacy in infinite streams [11, 27, 6, 7] capture counter-based scenarios, whereas in our model each event contributes to a *single* published statistic. Cao et al. [5] exploit overlapping query ranges, whereas we publish non-overlapping counts at each timestamp. Furthermore, all the schemes in the static scenario (described in Section 2.1) require a priori knowledge of all data, whereas we process the data on-the-fly as they arrive from the stream. The only ideas that can be adapted to our model are from [17, 4], noting though that these target at significantly different settings from ours. Specifically, FAST [17] supports *finite streams*, whereas Bolot et al. [4] study a more relaxed privacy concept called decayed privacy. We next design three competitors based on [17, 4]. The first is an instantiation of the user-level privacy adaptation to w-event privacy we discussed above, using FAST, which consists of sub mechanisms, each operating on a disjoint window of w timestamps. On the other hand, the other two, called Uniform and Sample, are tailored to our sliding window methodology, and are comprised of sub mechanisms, each operating at a single timestamp.

The first competitor, termed as $\mathsf{FAST_w}$, instantiates each sub mechanism $\mathcal{M}_i$ with FAST, and allocates budget $\epsilon/2$. $\mathcal{M}_i$ views its sub sequence as a finite stream of (a priori known) length w, and applies an adaptive sampling technique to each column count separately. Given a pre-specified number of samples, it selects (on-the-fly as it receives the stream) only a subset of timestamps to publish, skipping the rest and approximating them with the corresponding lastly published statistics. The budget allocation depends on the number of samples and the length of the stream, which explains why FAST cannot work directly with infinite streams.

Bolot et al. [4] utilize LPA independently at every timestamp. In particular, they inject at every published statistic Laplace noise with scale proportional to w instead of the entire stream length. Our Uniform scheme employs a similar approach. Specifically, Uniform is a mechanism $\mathcal{M}$ composed of sub mechanisms $\mathcal{M}_1, \mathcal{M}_2, \ldots$, such that, at every timestamp i, $\mathcal{M}_i$ calculates $\mathbf{Q}(D_i)$, and injects to the result Laplace noise with scale $\lambda_i = w \cdot \Delta(\mathbf{Q})/\epsilon$. Recall that, in our setting, the sensitivity $\Delta(\mathbf{Q})$ of $\mathbf{Q}$ at every timestamp is equal to 1. Hence, due to Theorem 1, $\mathcal{M}_i$ satisfies ϵ_i-differential privacy, where $\epsilon_i = 1/\lambda_i = \epsilon/w$. Observe that all the ϵ_i's are equal, i.e., Uniform distributes privacy budget *uniformly* across the timestamps. It is easy to see that, for *any* window of size w, the sum of the ϵ_i's therein is equal to ϵ, which satisfies the condition of Theorem 3 and, hence, leads to w-event privacy.

Finally, since the adaptive sampling of FAST cannot be applied to infinite streams, we tailor a simpler sampling technique to our sliding window methodology. Specifically, we design Sample as a mechanism $\mathcal{M}$ composed of sub mechanisms $\mathcal{M}_1, \mathcal{M}_2, \ldots$. Mechanism $\mathcal{M}_i$ publishes $\mathbf{Q}(D_i)$ with Laplace noise of scale $\lambda_i = 1/\epsilon$ when $(i \mod w) = 1$, and with *infinite* scale otherwise. Perceiving each publication with infinite noise as *skipped* and approximating it with its immediately preceding release, Sample performs a *single* publication every w timestamps (i.e., it samples the time domain with a *fixed rate*). $\mathcal{M}_i$ is ϵ-differentially private when $(i \mod w) = 1$ and 0-differentially private otherwise. Therefore, Sample satisfies Formula (1) and achieves w-event privacy.

4. PROPOSED METHODS

Section 4.1 outlines the main idea behind our constructions. Sections 4.2 and 4.3 introduce our BD and BA mechanisms, respectively, and Section 4.4 analyzes their utility. Section 4.5 includes effective optimizations.

4.1 Main Idea

$\mathsf{FAST_w}$ inherits the budget allocation shortcomings of the general user-level privacy adaptation we discussed in Section 3. On the contrary, Uniform and Sample adhere to our sliding window methodology. The latter requires that the sum of the individual privacy budgets of the sub mechanisms operating in any window of size w is at most equal to the total budget ϵ. Uniform assigns the same budget to every timestamp (equal to ϵ/w). If w is large, this budget becomes very small, which makes the noise scale prohibitively high and destroys the statistics. On the other hand, Sample invests the entire budget ϵ on a single timestamp within the window, which makes that publication very accurate. However, it approximates the next $w-1$ statistics with the published one. If these statistics greatly differ from the preceding release, the resulting error may become excessive and even exceed that of Uniform.

Our new solutions follow the sliding window methodology, and are motivated by the shortcomings of Uniform and Sample. Specifically, contrary to Sample, we decide on which publications to skip based on the *dissimilarity* between statistics. The main idea is to check at every timestamp whether it is more beneficial to approximate the current statistics with the last release, than to publish them with the necessary noise. This check is facilitated by a private dissimilarity calculation sub mechanism. Moreover, in contrast to Uniform, we do not allocate the privacy budget uniformly, but rather invest it on publications that need it the most. At each timestamp, our statistics are either (i) published with low noise, or (ii) accurately approximated by another release.

Im more detail, each scheme consists of a sequence of sub mechanisms $\mathcal{M}_1, \mathcal{M}_2, \ldots$, where $\mathcal{M}_i$ operates at timestamp i. Figure 1 illustrates the architecture of $\mathcal{M}_i$. It takes as input all the previous private publications $\mathbf{o}_1, \ldots, \mathbf{o}_{i-1}$ that occurred at timestamps $1, \ldots, i-1$, respectively, as well as budgets $\epsilon_1, \ldots, \epsilon_{i-1}$, and outputs a new transcript $\mathbf{o}_i$.

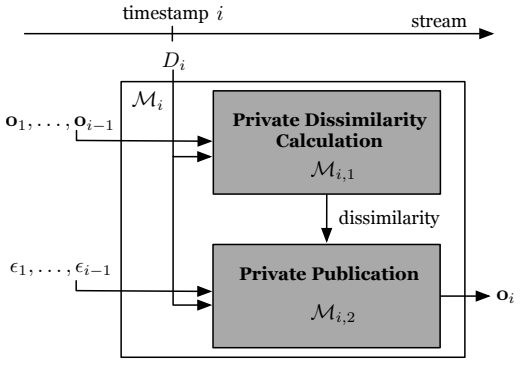


Figure 1: Internal mechanics of $\mathcal{M}_i$

Mechanism $\mathcal{M}_i$ is further decomposed into two sub mechanisms $\mathcal{M}_{i,1}$ and $\mathcal{M}_{i,2}$, which operate *sequentially* with *independent randomness*. $\mathcal{M}_{i,1}$ performs a dissimilarity computation algorithm between $\mathbf{c}_i$ of the incoming database D_i and the *last private release* belonging in $\mathbf{o}_1, \ldots, \mathbf{o}_{i-1}$. The dissimilarity measure is orthogonal to the implementation; any standard measure is applicable. The result of the calculation is forwarded to $\mathcal{M}_{i,2}$, which uses it to decide whether to publish $\mathbf{c}_i$ or not. $\mathcal{M}_{i,2}$ publishes $\mathbf{o}_i$, which is

either $\mathbf{c}_i$ with noise (decided based on $\epsilon_1, \ldots, \epsilon_{i-1}$), or *null*. In the latter case, $\mathbf{c}_i$ is approximated with the last non-null publication.

Both $\mathcal{M}_{i,1}$ and $\mathcal{M}_{i,2}$ must be private, i.e., we must invest privacy budget on them, which will determine the amount of noise injected in their outputs. Note that, although the dissimilarity result of $\mathcal{M}_{i,1}$ appears only within the internal operation of $\mathcal{M}_i$, we must consider it as being published. This is because the adversary knows that $\mathcal{M}_{i,1}$ computes the dissimilarity on the sensitive data D_i, whose value affects the decision taken by $\mathcal{M}_{i,2}$. Thus, $\mathcal{M}_{i,1}$ must add proper noise to the dissimilarity value. Let $\epsilon_{i,1}$ and $\epsilon_{i,2}$ be the budgets spent in $\mathcal{M}_{i,1}$ and $\mathcal{M}_{i,2}$, respectively. Then, due to Theorem 2, the privacy budget of $\mathcal{M}_i$ is $\epsilon_i = \epsilon_{i,1} + \epsilon_{i,2}$.

We next present two schemes, BD and BA, that follow the architecture described above. They share a common private dissimilarity calculation mechanism ($\mathcal{M}_{i,1}$), which always spends a *fixed* budget $\epsilon_{i,1}$ for each timestamp. However, they differ in the private publication mechanism ($\mathcal{M}_{i,2}$); each implements a different *dynamic* allocation method for budget $\epsilon_{i,2}$ over the stream. We refer to $\epsilon_{i,1}$ as the *dissimilarity budget*, and to $\epsilon_{i,2}$ as the *publication budget*.

4.2 Budget Distribution (BD)

The *Budget Distribution* (BD) scheme starts with the entire budget ϵ and (i) allocates some fixed dissimilarity budget per timestamp, (ii) distributes publication budget in an *exponentially decreasing* fashion to the timestamps where a publication is decided to occur, and (iii) recycles the budget spent in timestamps falling outside the active window. If the mechanism decides to publish at timestamp i, then the statistics at i are dissimilar from the last release. Thus, it is beneficial to invest a high budget (i.e., inject low noise) to the statistics at i, in order to (i) prevent distorting the dissimilarity of the current statistics to the previous ones, and (ii) retain the accuracy of the current statistics to better approximate future ones if necessary. BD hopes that few publications will take place in the same window. Hence, the first publications receive an exponentially higher portion of the budget than the subsequent ones as the window slides over time. When an old timestamp falls out of the active window, its publication budget is recycled, essentially "resetting" the available budget for future publications.

Figure 2 illustrates the pseudocode of $\mathcal{M}_i$ (operating at timestamp i) of BD. As explained in Section 4.1, $\mathcal{M}_i$ consists of the private dissimilarity calculation sub mechanism $\mathcal{M}_{i,1}$ (Lines 1-4), and the private publication sub mechanism $\mathcal{M}_{i,2}$ (Lines 5-8). $\mathcal{M}_{i,1}$ initially computes the query result on D_i (Line 1); this accounts for the vector $\mathbf{c}_i$ of the d column counts of D_i, where d is a public system parameter. Next, it finds the last non-null release $\mathbf{o}_l$ from $(\mathbf{o}_1, \ldots, \mathbf{o}_{i-1})$ in Line 2. Subsequently, it calculates the dissimilarity *dis* between $\mathbf{c}_i$ and $\mathbf{o}_l$, employing the *Mean Absolute Error* (MAE) as the dissimilarity measure (Line 3). We use MAE because this is how we calculate the error in our utility analysis and experiments. $\mathcal{M}_{i,1}$ invests some *fixed* dissimilarity budget $\epsilon_{i,1} = \epsilon/(2 \cdot w)$ to perturb the *dis* value. The reason behind this choice of $\epsilon_{i,1}$ will become clear soon. Next (Lines 3-4), it injects to *dis* Laplace noise with scale $\lambda_{i,1} = (2 \cdot w)/(\epsilon \cdot d)$ because, as we shall soon prove, this renders $\mathcal{M}_{i,1}$ as $\epsilon_{i,1}$-differentially private.

$\mathcal{M}_{i,2}$ calculates the remaining budget ϵ_{rm} for the active window $[i-w+1, i]$ (Line 5), since it must make sure that the total budget spent in the current window does not exceed ϵ, before determining $\epsilon_{i,2}$. Note that ϵ_{rm} is equal to ϵ minus (i) the dissimilarity budget spent in window $[i-w+1, i]$, and (ii) the publication budget spent in window $[i-w+1, i-1]$. The budget in (i) is equal to $\sum_{k=i-w+1}^{i} \epsilon_{k,1} = w \cdot (\epsilon/(2 \cdot w)) = \epsilon/2$, i.e., it is equal to half the maximum budget, which leaves (at most) another half available for $\mathcal{M}_{i,2}$ in the same window. This justifies our choice to fix $\epsilon_{i,1}$ to

Input: $D_i, (\mathbf{o}_1, \ldots, \mathbf{o}_{i-1}), (\epsilon_1, \ldots, \epsilon_{i-1})$
Output: $\mathbf{o}_i$

// Sub mechanism $\mathcal{M}_{i,1}$
1. Calculate $\mathbf{c}_i = \mathbf{Q}(D_i)$
2. Identify last non-null release $\mathbf{o}_l$ from $(\mathbf{o}_1, \ldots, \mathbf{o}_{i-1})$
3. Set $dis = \frac{1}{d} \sum_{j=1}^{d} |\mathbf{o}_l[j] - \mathbf{c}_i[j]|$ and $\lambda_{i,1} = (2 \cdot w)/(\epsilon \cdot d)$
4. Set $dis = dis + Lap(\lambda_{i,1})$

// Sub mechanism $\mathcal{M}_{i,2}$
5. Calculate remaining budget $\epsilon_{rm} = \epsilon/2 - \sum_{k=i-w+1}^{i-1} \epsilon_{k,2}$
6. Set $\lambda_{i,2} = 2/\epsilon_{rm}$
7. **If** $dis > \lambda_{i,2}$, **Return** $\mathbf{o}_i = \mathbf{c}_i + \langle Lap(\lambda_{i,2}) \rangle^d$
8. **Else Return** $\mathbf{o}_i = null$

Figure 2: Pseudocode of $\mathcal{M}_i$ in BD

$\epsilon/(2 \cdot w)$. On the other hand, the budget in (ii) is $\sum_{k=i-w+1}^{i-1} \epsilon_{k,2}$, where $\epsilon_{k,2}$ is not fixed (it depends on how publications occurred in the past). Observe that $\sum_{k=i-w+1}^{i-1} \epsilon_{k,2}$ encompasses only budgets spent in $[i-w+1, i-1]$, which implies that any budget used outside this window becomes available again (for technical reasons, for $k \leq 0$ we set $\epsilon_{k,1} = \epsilon/(2 \cdot w)$ and $\epsilon_{k,2} = 0$).

Next, in Lines 6-8, $\mathcal{M}_{i,2}$ publishes the counts $\mathbf{c}_i$ if it is more beneficial than approximating them with the last release $\mathbf{o}_l$. Specifically, it checks whether the error from the approximation with $\mathbf{o}_l$, which is equal to dis, is larger than the error that will be induced if $\mathbf{c}_i$ is published with Laplace noise of scale $\lambda_{i,2}$ (which is set to $2/\epsilon_{rm}$ in Line 6). The error from this noise, which is expressed as the MAE on pair $(\mathbf{c}_i, \mathbf{o}_i)$, is equal to $\lambda_{i,2}$. If the condition in Line 7 is true, then we publish $\mathbf{c}_i$ along with noise $\langle Lap(\lambda_{i,2}) \rangle^d$; in this case, we will show that the budget spent by $\mathcal{M}_{i,2}$ is equal to $\epsilon_{i,2} = 1/\lambda_{i,2} = \epsilon_{rm}/2$. Otherwise, $\mathcal{M}_{i,2}$ sets $\mathbf{o}_i$ to *null* (Line 8), which is equivalent to approximating $\mathbf{c}_i$ with $\mathbf{o}_l$ and, hence, spending no budget for $\mathcal{M}_{i,2}$, i.e., $\epsilon_{i,2} = 0$.

We make the following observations about BD. Every time we publish the current statistics $\mathbf{c}_i$ at some i, the budget allocated for $\epsilon_{i,2}$ is *half* the available budget for the active window. Now consider that m publications occur after i, without recycling any old budget. Then, in the m^{th} publication, the publication sub mechanism will use budget $\epsilon_{i,2}/2^m$. This suggests that BD allocates the $\epsilon_{i,2}$ budgets in an *exponentially decreasing* fashion. Moreover, recycling old budget "resets" the available budget and, hence, the latter does not decrease indefinitely, but rather fluctuates over time.

Figure 3 illustrates the operation of BD in 6 timestamps for $w = 3$. Assume that the mechanism decides to publish at timestamps $1, 3, 4$, releasing transcripts $\mathbf{o}_1, \mathbf{o}_3, \mathbf{o}_4$, respectively. At timestamps $2, 5, 6$, it outputs *null*, which implies that the statistics at 2 are approximated with $\mathbf{o}_1$, and those of $5, 6$ with $\mathbf{o}_4$. For every timestamp i, the $\mathcal{M}_{i,1}$ mechanism of BD enjoys a fixed budget $\epsilon_{i,1} = \epsilon/(2 \cdot w) = \epsilon/6$. Concerning $\mathcal{M}_{i,2}$, at timestamp 1, it assigns $\epsilon_{1,2} = (\epsilon/2 - 0)/2 = \epsilon/4$ according to Lines 5-6 in Figure 2. At timestamp 2, $\epsilon_{2,2} = 0$ since no publication occurs. Then, at timestamp 3, it allocates $\epsilon_{3,2} = (\epsilon/2 - (\epsilon/4 + 0))/2 = \epsilon/8$, which is *half* of $\epsilon_{1,2}$. This indicates the exponential decrease in the budgets across subsequent publications within the same window. At timestamp 4, BD assigns $\epsilon_{4,2} = (\epsilon_2 - (0 + \epsilon/8))/2 = 3\epsilon/16$. Observe that, although the mechanism still allocates half of the remaining budget at timestamp 4, $\epsilon_{4,2} > \epsilon_{3,2}$. This is because the budget invested at 1 ($\epsilon/4$) is *recycled*, i.e., added to the remaining budget. The mechanism continues similarly at timestamps 5 and 6. Finally, notice that, in every window of size $w = 3$, the sum of the budgets used by BD (which encompasses the budgets for *both* $\mathcal{M}_{i,1}$ and $\mathcal{M}_{i,2}$), is always smaller than or equal to ϵ, which is the requirement for satisfying 3-event privacy.

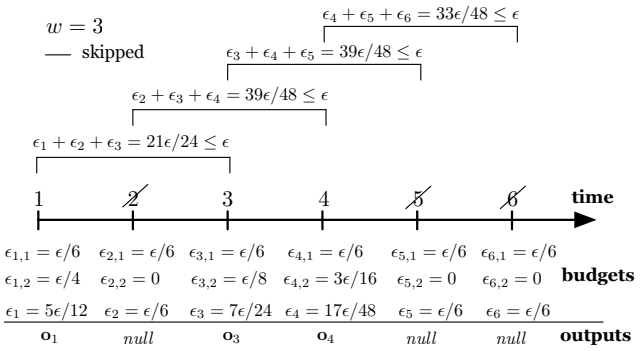


Figure 3: Example of BD

THEOREM 4. BD *satisfies w-event privacy.*

PROOF. We first prove that $\mathcal{M}_{i,1}$ satisfies $\epsilon_{i,1}$-differential privacy for $\epsilon_{i,1} = \epsilon/(2 \cdot w)$, and $\mathcal{M}_{i,2}$ is $\epsilon_{i,2}$-differentially private for $\epsilon_{i,2} = (\epsilon/2 - \sum_{k=i-w+1}^{i-1} \epsilon_{k,2})/2$ if it publishes, and $\epsilon_{i,2} = 0$ otherwise. $\mathcal{M}_{i,1}$ privately outputs the MAE value $\mathbf{Q}'(D_i) = \frac{1}{d}\sum_{j=1}^{d} |\mathbf{o}_l[j] - \mathbf{c}_i[j]|$. An insertion or deletion of a row from D_i results in altering the above result by at most $1/d$ (recall that, in our setting, every row has a single 1 with all the rest elements being 0). Hence, the sensitivity of $\mathbf{Q}'$ is $\Delta(\mathbf{Q}') = 1/d$. $\mathcal{M}_{i,1}$ injects to MAE Laplace noise with scale $\lambda_{i,1} = (2 \cdot w)/(\epsilon \cdot d)$. Due to Theorem 1, $\mathcal{M}_{i,1}$ is $\epsilon_{i,1}$-differential privacy for $\epsilon_{i,1} = \frac{1/d}{(2\cdot w)/(\epsilon \cdot d)} = \epsilon/(2 \cdot w)$. $\mathcal{M}_{i,2}$ privately publishes $\mathbf{Q}(D_i) = \mathbf{c}_i$ or $null$. In the former case, the sensitivity of $\mathbf{Q}$ is $\Delta(\mathbf{Q}) = 1$ in our setting, and the mechanism injects Laplace noise with scale $2/(\epsilon/2 - \sum_{k=i-w+1}^{i-1} \epsilon_{k,2})$. Hence, due to Theorem 1, $\mathcal{M}_{i,2}$ is $\epsilon_{i,2}$-differential privacy for $\epsilon_{i,2} = (\epsilon/2 - \sum_{k=i-w+1}^{i-1} \epsilon_{k,2})/2$. In the latter case, $\epsilon_{i,2}$ is trivially equal to zero, as no publication occurs.

According to Theorem 3, we must prove that, for every t and $i \in [t]$, it holds that $\sum_{k=i-w+1}^{i} \epsilon_k \leq \epsilon$. Due to Theorem 2, in BD it holds that $\epsilon_k = \epsilon_{k,1} + \epsilon_{k,2}$ and, hence, $\sum_{k=i-w+1}^{i} \epsilon_k = \sum_{k=i-w+1}^{i} \epsilon_{k,1} + \sum_{k=i-w+1}^{i} \epsilon_{k,2} = \epsilon/2 + \sum_{k=i-w+1}^{i} \epsilon_{k,2}$, since every $\epsilon_{k,1}$ is fixed to $\epsilon/(2 \cdot w)$. Therefore, it suffices to prove that $0 \leq \sum_{k=i-w+1}^{i} \epsilon_{k,2} \leq \epsilon/2$, where the comparison with zero is for correctness (differential privacy is not defined for $\epsilon < 0$). We can prove the above by induction, setting the first w timestamps as the base window/case, and sliding an arbitrary window in the stream by one timestamp in the inductive step (we omit the details due to space limitations). Intuitively, this holds because $\mathcal{M}_{i,2}$ always uses up to *half* of the available publication budget. □

4.3 Budget Absorption (BA)

The *Budget Absorption* (BA) mechanism follows a similar decision algorithm to BD for publishing at the timestamps based on dissimilarity, but implements a different budget allocation strategy. Specifically, it starts by *uniformly* distributing publication budget to all timestamps. If it decides not to publish at a timestamp, the corresponding budget becomes available for a future publication. On the other hand, if it decides to publish at a timestamp, it *absorbs* all the budget that became available from the previous skipped publications, and uses it in order to publish the current statistics with higher accuracy. An important subtlety arises in this case. Whenever the mechanism absorbs budget from previous timestamps, it must *nullify* the *same* budget from the immediately succeeding timestamps, *forcing* their outputs to become $null$. The nullified budgets cannot be absorbed in the future. This is vital for not exceeding the maximum budget ϵ as the active window slides over time.

Figure 4 presents the pseudocode of BA. The $\mathcal{M}_{i,1}$ sub mechanism is identical to that of BD (Lines 1-4 in Figure 2). The $\mathcal{M}_{i,2}$ sub mechanism allocates a *fixed* budget $\epsilon_{i,2} = \epsilon/(2 \cdot w)$ to each timestamp i. However, we distinguish two cases at i: (i) if previous publications were skipped, their budgets are added to $\epsilon_{i,2}$, and (ii) if a previous publication absorbed budget, $\epsilon_{i,2}$ may be nullified, i.e., set to zero, in which case the output is $\mathbf{o}_i = null$. The second case is captured first in Lines 5-6, whereas the first case is captured next in Lines 7-11. We explain these two cases in turn below.

Input: $D_i, (\mathbf{o}_1, \ldots, \mathbf{o}_{i-1}), (\epsilon_1, \ldots, \epsilon_{i-1})$
Output: $\mathbf{o}_i$

// Sub mechanism $\mathcal{M}_{i,1}$
// Same as Lines 1-4 in Figure 2

// Sub mechanism $\mathcal{M}_{i,2}$
5. $to\_nullify = \frac{\epsilon_{l,2}}{\epsilon/(2\cdot w)} - 1$
6. **If** $i - l \leq to\_nullify$, **Return** $\mathbf{o}_i = null$
7. **Else**
8. Set $to\_absorb = i - (l + to\_nullify)$
9. Set $\epsilon_{i,2} = \epsilon/(2 \cdot w) \cdot \min(to\_absorb, w)$ and $\lambda_{i,2} = 1/\epsilon_{i,2}$
10. **If** $dis > \lambda_{i,2}$, **Return** $\mathbf{o}_i = \mathbf{c}_i + \langle Lap(\lambda_{i,2}) \rangle^d$
11. **Else Return** $\mathbf{o}_i = null$

Figure 4: Pseudocode of $\mathcal{M}_i$ in BA

Since we know the last publication $\mathbf{o}_l$ from $\mathcal{M}_{i,1}$, we can calculate the amount of budget $\epsilon_{l,2}$ it received. If we divide $\epsilon_{l,2}$ by $\epsilon/(2 \cdot w)$ and subtract 1 (Line 5), we get the number of skipped publications whose budgets $\epsilon_{l,2}$ absorbed. This value, called $to\_nullify$ in our pseudocode, indicates the number of timestamps after l whose budgets we must nullify. Therefore, if $i - l \leq to\_nullify$, then the budget at i is nullified and the mechanism outputs $\mathbf{o}_i = null$ (Line 6). Otherwise, $\mathcal{M}_{i,2}$ calculates $\epsilon_{i,2}$ after potentially absorbing the budget of skipped publications, and decides on whether to publish the noisy statistics or approximate them with $\mathbf{o}_l$ (Lines 8-11). In particular, it first computes the number of skipped publications between the last publication timestamp l and i, and stores it in variable to_absorb (Line 8). Note that we must not consider the nullified budgets as skipped budgets; the nullified budgets cannot be absorbed. Next (Line 9), the mechanism sets $\epsilon_{i,2}$, which is between $\epsilon/(2 \cdot w)$ and $\epsilon/2$, since the number of skipped budgets to be absorbed cannot exceed $w - 1$ (otherwise a window will utilize more than ϵ budget and violate w-event privacy). BA proceeds similarly to BD, that is, it sets $\lambda_{i,2} = 1/\epsilon_{i,2}$ (Line 9), compares the dissimilarity dis to the error from injecting Laplace noise with scale $\lambda_{i,2}$, and outputs the noisy counts if the dissimilarity is larger (Line 10), or $null$ otherwise (Line 11).

Figure 5 demonstrates the BA mechanism in 6 timestamps with $w = 3$. At all timestamps, BA allocates $\epsilon/(2 \cdot w) = \epsilon/6$ budget for both $\mathcal{M}_{i,1}$ and $\mathcal{M}_{i,2}$. At the first timestamp, it publishes the counts using the current budget $\epsilon_{1,2} = \epsilon/6$. Suppose that, at timestamp 2, BA decides not to publish the current counts, i.e., it outputs $null$ and the statistics are approximated by $\mathbf{o}_1$. Budget $\epsilon_{2,1}$ is then set to 0, and $\epsilon/6$ becomes available for absorption at a future timestamp. At timestamp 3, the mechanism decides to publish the current counts. It absorbs the $\epsilon/6$ budget that became available at timestamp 2, and adds it to its already allocated budget, yielding $\epsilon_{3,2} = \epsilon/3$. Due to this absorption, the budget at timestamp 4 must be nullified (i.e., $\epsilon_{4,2} = 0$) and, hence, the mechanism outputs $null$. At timestamp 5, the mechanism decides to publish, but there is no budget to absorb (recall that the nullified budget at timestamp 4 cannot be absorbed). Thus, $\mathcal{M}_{5,2}$ uses only its own budget $\epsilon/6$ to publish $\mathbf{o}_5$. Finally, at timestamp 6, the mechanism decides not to publish, and proceeds similarly. For any window of size 3, the sum

of the budgets therein is at most ϵ. Note that, if BA does not nullify the budget at timestamp 4, the sum of the budgets in window $[3, 5]$ is $7\epsilon/6 > \epsilon$, which violates w-event privacy.

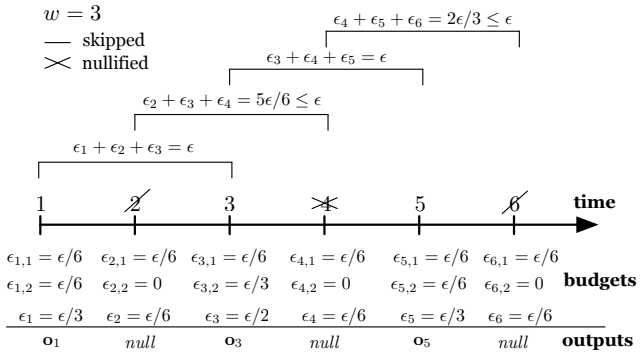


Figure 5: Example of BA

THEOREM 5. BA *satisfies w-event privacy.*

PROOF. Similar to BD, $\mathcal{M}_{i,1}$ satisfies $\epsilon_{i,1}$-differential privacy for $\epsilon_{i,1} = \epsilon/(2 \cdot w)$, and $\mathcal{M}_{i,2}$ is $\epsilon_{i,2}$-differentially private, where $\epsilon_{i,2}$ depends on previous publications, since it may be nullified (Lines 5-6 in Figure 4), or absorb additional budget (Lines 8-9), or be absorbed (Line 11). Moreover, in any window of size w, the sum of budgets spent by $\mathcal{M}_{i,1}$ is equal to $\epsilon/2$. Therefore, it suffices to prove that $0 \leq \sum_{k=i-w+1}^{i} \epsilon_{k,2} \leq \epsilon/2$.

Let i be a timestamp which absorbed budget from α preceding timestamps. According to BA, it holds that (i) $\epsilon_{i,2} = (\alpha + 1) \cdot \epsilon/(2 \cdot w)$, (ii) $\epsilon_{k,2} = 0$ for $(i - \alpha \leq k \leq i - 1) \wedge (i + 1 \leq k \leq i + \alpha)$, and (iii) $0 \leq \alpha \leq w - 1$. Then, any window of size w that contains i also covers $n \geq \alpha$ timestamps with $\epsilon_{k,2} = 0$ which were either absorbed or nullified *exclusively* by i. Therefore, the sum of the budgets of i along with the n zero-budget timestamps is at most $(\alpha + 1) \cdot \epsilon/(2 \cdot w)$, i.e., at most equal to the case where each of these $n + 1$ timestamps receives *uniform* budget $\epsilon_{k,2} = \frac{(\alpha+1)\cdot\epsilon/(2\cdot w)}{n+1} \leq \epsilon/(2 \cdot w)$. The above holds *independently* also for any timestamp i' that absorbed budget from α' previous timestamps and lies in the same window as i. Therefore, $\sum_{k=i-w+1}^{i} \epsilon_{k,2} \leq \sum_{k=i-w+1}^{i} \epsilon/(2 \cdot w) = \epsilon/2$. Finally, since $\alpha \geq 0$, $\epsilon_{k,2} \geq 0$ for every k and, hence, $\sum_{k=i-w+1}^{i} \epsilon_{k,2} \geq 0$. □

To sum up, the difference between BA and BD is the following. BD optimistically assumes that few publications will take place in each window and, hence, at each publication it eagerly allocates a large portion of the available budget. On the other hand, BA initially assumes that all publications are likely to occur in the window and, thus, uniformly allocates the budget among them. However, it absorbs the entire budget from the skipped publications, and nullifies the budgets from the immediately succeeding timestamps, because it optimistically assumes that successive statistics may not differ substantially. Therefore, it assigns a large budget to the current publication, hoping that the latter can accurately approximate at least the next few publications.

4.4 Utility Analysis

Recall that *both* BD and BA are *data-dependent*. The error at any timestamp depends on (i) the budget used in past releases (if a publication occurs), and (ii) how well the statistics at this timestamp are approximated by the previous release (if a publication does not occur). Such data-dependent mechanisms should be evaluated through exhaustive experiments on real datasets, in order to better capture their effectiveness, a task we undertake in Section 5. In this section we also include a rigorous utility analysis, which pronounces though the data-dependent aspects. We calculate the error of publication $\mathbf{o}_i$ as the MAE on pair $(\mathbf{o}_i, \mathbf{c}_i)$ if $\mathbf{o}_i \neq null$, and as the MAE on $(\mathbf{o}_l, \mathbf{c}_i)$ otherwise, where $\mathbf{o}_l$ is the first non-null release preceding i. In the following, we focus in turn on Uniform, Sample, BD and BA.

Analysis of Uniform and Sample. The expected error of Uniform is equal to the error stemming from the Laplace noise addition. Since its scale is always $\lambda = w/\epsilon$, the error at each timestamp is w/ϵ. Similarly, Sample results in an error of $1/\epsilon$ at any timestamp i, for $(i \mod w) = 1$, due to the injected Laplace noise with scale $\lambda = 1/\epsilon$. However, the error at any other timestamp is equal to the error from the approximation with the previous release, which we cannot quantify in Sample.

Analysis of BD. To facilitate presentation, we first analyze the error of $\mathcal{M}_{i,2}$ considering that $\mathcal{M}_{i,1}$ is not private and returns the dissimilarity value without error. Subsequently, we eliminate this assumption and assess how the error in $\mathcal{M}_{i,1}$ affects the overall error of BD. We assume that every publication approximates the same number of skipped publications.

LEMMA 1. *The average error per timestamp for $\mathcal{M}_{i,2}$ in* BD *is at most $4\frac{2^m-1}{m\epsilon}$, if m publications occur in a window and given that $\mathcal{M}_{i,1}$ is not private.*

PROOF. At any timestamp i, if $\mathcal{M}_{i,2}$ publishes, the error is $2/\epsilon_{rm}$ since the Laplace scale is $\lambda_{i,2} = 2/\epsilon_{rm}$; otherwise, the approximation provides a better error than $2/\epsilon_{rm}$. In both cases, we can upper bound the error at timestamp i by $2/\epsilon_{rm}$, given that $\mathcal{M}_{i,1}$ returns the actual dissimilarity value without noise. Consider the worst case, in which $m \leq w$ publications occur in a window of size w, and *no* budget is recycled from old timestamps that lie outside the window. Recall that BD distributes the budgets to the m publications in an exponentially decreasing fashion, namely the budget sequence is $\epsilon/4, \epsilon/8, \ldots, \epsilon/2^{m+1}$. Since each publication approximates the same number of skipped releases, every publication approximates $w/m - 1$ skipped releases on average. In other words, the error induced by each publication is shared among w/m timestamps. Hence, the average error per timestamp in the window is bounded by $\frac{1}{w} \cdot \left(\frac{w}{m} \cdot \frac{4}{\epsilon} + \ldots + \frac{w}{m} \cdot \frac{2^{m+1}}{\epsilon}\right) = 4\frac{2^m-1}{m\epsilon}$. □

Sub mechanism $\mathcal{M}_{i,1}$ introduces error to the dissimilarity value. Thus, it contributes an additive factor to the error of $\mathcal{M}_{i,2}$. This is captured in the theorem below, which states the total error of BD.

THEOREM 6. *The average error per timestamp in* BD *is at most $4\frac{2^m-1}{m\epsilon} + \frac{2\cdot w}{\epsilon\cdot d}$, if m publications occur in a window.*

PROOF. $\mathcal{M}_{i,1}$ induces error when its noisy dissimilarity output guides $\mathcal{M}_{i,2}$ into making a *false* publication decision, i.e., when $\mathcal{M}_{i,2}$ (i) falsely skips a publication, or (ii) falsely performs a publication. When a *correct* publication occurs, the error of BD is the error induced by $\mathcal{M}_{i,2}$, which is captured by Lemma 1. When a publication is *correctly* skipped, the error is the *actual* dissimilarity between the current statistics $\mathbf{c}_i$ and the last publication $\mathbf{o}_l$, which is bounded by the error of $\mathcal{M}_{i,2}$ (again captured by Lemma 1). However, when a publication is *falsely* skipped, the actual dissimilarity is *underestimated* due to the noise of scale $\lambda_{i,1}$ added by $\mathcal{M}_{i,1}$. Conversely, when a *false* publication occurs, the actual dissimilarity is *overestimated* due to the noise of scale $\lambda_{i,1}$ added by $\mathcal{M}_{i,1}$. The expected under/overestimation of dissimilarity is equal to $\frac{2\cdot w}{\epsilon\cdot d}$ due to the noise of $\mathcal{M}_{i,1}$. Hence, due to Lemma 1, the average error of BD is $4\frac{2^m-1}{m\epsilon} + \frac{2\cdot w}{\epsilon\cdot d}$. □

BD is benefited when the number of publications m per window is small, otherwise its error increases exponentially with m. Moreover, the error from $\mathcal{M}_{i,1}$ (second term of the total error) rises with w, but diminishes as d increases. This is justified by the fact that the dissimilarity measure is MAE, which averages the attribute differences over d and, hence, reduces the sensitivity.

Analysis of BA. Contrary to BD, BA distinguishes between skipped and nullified publications. The former are decided based on dissimilarity, whereas the latter are enforced due to budget absorption. We can assess the error of the skipped publications similarly to BD. However, we cannot quantify the error of the nullified publications, as we possess no information about the underlying statistics; nullification is enforced prior to their arrival. Therefore, we will express the error as a function of the average error for a nullified publication, which we denote by err_{nlf}. Moreover, in contrast to BD, the error of BA depends on the average number of absorbed budgets (i.e., skipped releases) per publication, denoted by α, rather than the number of publications per window.

THEOREM 7. *The average error per timestamp in* BA *is at most* $\frac{1}{2\alpha+1} \cdot (\frac{2 \cdot w}{\epsilon} \cdot H_{\alpha+1} + \alpha \cdot err_{nlf}) + \frac{2 \cdot w}{\epsilon \cdot d}$*, where* α *is the number of absorbed budgets per publication, and* H_x *is the* x^{th} *harmonic number.*

PROOF. Similar to BD, we first analyze the error of $\mathcal{M}_{i,2}$, and then add the error contributed by $\mathcal{M}_{i,1}$, which is the same as in BD (since $\mathcal{M}_{i,1}$ is identical in BD and BA). Every publication is associated with α skipped publications preceding it (whose budgets it absorbs), and α nullified publications succeeding it. The publication receives budget $\frac{(\alpha+1) \cdot \epsilon}{2 \cdot w}$ and, hence, its error is $\frac{2 \cdot w}{(\alpha+1) \cdot \epsilon}$. The first of the α skipped publications cannot have a larger error than $\frac{2 \cdot w}{\epsilon}$ because, otherwise, it would not have been skipped. The second skipped publication cannot have a larger error than $\frac{2 \cdot w}{2 \cdot \epsilon}$, since it performs the dissimilarity check considering that it absorbs the budget from the first skipped publication. In a similar fashion, we can obtain that the α^{th} skipped publication induces at most error $\frac{2 \cdot w}{a \cdot \epsilon}$. Furthermore, each of the α nullified publications introduces error err_{nlf}. Averaging over the above $2\alpha + 1$ errors, we derive the average error per timestamp of $\mathcal{M}_{i,2}$ in BA, which is $\frac{1}{2\alpha+1} \cdot (\frac{2 \cdot w}{\epsilon} + \ldots + \frac{2 \cdot w}{\alpha \cdot \epsilon} + \frac{2 \cdot w}{(\alpha+1) \cdot \epsilon} + \alpha \cdot err_{nlf}) = \frac{1}{2\alpha+1} \cdot (\frac{2 \cdot w}{\epsilon} \cdot H_{\alpha+1} + \alpha \cdot err_{nlf})$. Adding the error incorporated by $\mathcal{M}_{i,1}$ (see Theorem 6), we get the average error per timestamp in BA, namely $\frac{1}{2\alpha+1} \cdot \left(\frac{2 \cdot w}{\epsilon} \cdot H_{\alpha+1} + \alpha \cdot err_{nlf}\right) + \frac{2 \cdot w}{\epsilon \cdot d}$. □

The error in BA decreases as the number α of skipped publications increases. This is because (i) the skipped publications are approximated with a smaller error than that they would have induced were they published with noise, and (ii) budgets from the skipped publications are utilized by others to increase their accuracy. The average error err_{nlf} of a nullified publication contributes to the total error, but it solely depends on the underlying data. Finally, $\mathcal{M}_{i,1}$ injects some error as well which, similar to BD, decreases as d becomes larger than w.

4.5 Optimizations and Extensions

In this section we provide optimizations for BD and BA, and explain how to augment our schemes with pan-privacy. Due to space limitations, we only sketch the modifications, delegating the detailed descriptions and proofs to the long version of our paper.

Column partitioning. When our schemes calculate the dissimilarity (MAE) between counts $\mathbf{c}_i$ and last release $\mathbf{o}_l$, they consider *all* the elements of $\mathbf{c}_i$. Suppose that very few counts from $\mathbf{c}_i$ change over time, whereas others remain roughly constant. In this case, only a few counts are responsible for increasing MAE and, hence, for potentially causing a publication to occur at timestamp i. When this release occurs, *all* the counts receive the noise, i.e., even the ones that could have been approximated with the previous release more accurately. Ideally, we should allow column counts to be published with different rates, based on their fluctuation over time.

Towards this goal, we *partition* the columns into groups, based on the magnitude of change of their counts over time. We form the groups periodically by observing the *private* releases. Then, we execute an *independent instantiation* of BD or BA on every group per timestamp. Note that we do not compromise the privacy of the overall scheme, as we do not access any raw data. The result is that the instantiations of the groups with high magnitude of change will publish more frequently than those of groups with low magnitude.

Shifting budget from $\mathcal{M}_{i,1}$ to $\mathcal{M}_{i,2}$. Both BD and BA assign a fixed budget $\epsilon_{i,1} = \epsilon/(2 \cdot w)$ to $\mathcal{M}_{i,1}$, which results in injecting noise with scale $\lambda_{i,1} = (2 \cdot w)/(\epsilon \cdot d)$ due to the sensitivity $1/d$ of MAE. Over a window of size w, the sum of the budgets for the dissimilarity calculation sub mechanism is $\epsilon/2$, which leaves only half of the maximum budget to $\mathcal{M}_{i,2}$. We investigate whether we can shift some budget from $\mathcal{M}_{i,1}$ to $\mathcal{M}_{i,2}$, in order to improve the accuracy of the publications, without significantly affecting the accuracy of the noisy dissimilarity value returned by $\mathcal{M}_{i,1}$. Specifically, we observe that, when $d \gg w$, $\lambda_{i,1}$ becomes too small. Motivated by the fact that the typical scale value of $1/\epsilon$ yields high accuracy in other scenarios (e.g., histogram queries [12]), we set $\lambda_{i,1} = 1/\epsilon$, which translates to spending budget $\epsilon_{i,1} = \epsilon/d < \epsilon/(2 \cdot w)$. This leaves $\epsilon - w \cdot (\epsilon/d) > \epsilon/2$ budget for publication in any window.

Pan-Privacy. A notion relevant to ϵ-differential privacy on streams is *pan-privacy* [15]. A mechanism is pan-private if it can preserve ϵ-differential privacy even when an adversary observes snapshots of the mechanism's *internal states*. These states account for the memory contents during the execution of the mechanism. There are two different intrusion types in the literature; *single unannounced intrusion* and *multiple announced intrusions*. The former assumes that the adversary breaches the system only *once* during its lifetime, without the curator being able to detect the intrusion. The latter considers that the adversary infiltrates the system multiple times, and the curator detects each breach just after it occurs.

We modify BD and BA so that they satisfy pan-privacy. A crucial point towards this goal is to never store raw data in main memory. Therefore, at every timestamp i, we reset dis and initialize it with noise $Lap(\lambda_{i,1})$. We do the same for all the elements of $\mathbf{c}_i$ using noise $Lap(\lambda_{i,2})$. Then, dis and $\mathbf{c}_i$ are calculated as D_i arrives from the communication channel, without storing any raw item in a variable without pre-assigned noise. Values $\lambda_{i,1}$ and $\lambda_{i,2}$ are calculated as follows. For $\mathcal{M}_{i,1}$, $\lambda_{i,1}$ remains unaffected because, as mentioned in Section 4.2, we perceive dis as being published at every timestamp. Hence, any intrusion does not disclose any additional information on dis. On the other hand, we must modify $\lambda_{i,2}$ for BD and BA. The reason is that, although an intrusion at a timestamp where a publication must occur does not help the adversary (since we release what the adversary accessed in main memory anyway), an intrusion at a timestamp where $\mathcal{M}_{i,2}$ outputs $\mathbf{o}_i = null$ gives extra information. This case becomes equivalent to publishing the noisy $\mathbf{c}_i$, since the adversary accessed it in memory. Consequently, the budget that would have been saved by the approximation is lost and, hence, future budgets (and corresponding noise scales) must be adjusted to retain w-event privacy.

We first focus on the case of a single unannounced intrusion. In BD, instead of setting $\epsilon_{i,2} = \epsilon_{rm}/2$, we set it to $\epsilon_{rm}/3$, i.e., to a

smaller value. For the windows where no intrusion takes place, privacy is not affected; following a similar analysis to Theorem 4, we derive that the sum of budgets spent for publication therein are at most $\epsilon/3 \leq \epsilon/2$ as required for w-event privacy. Consider now any window where the (single) intrusion takes place for a timestamp i where $\mathcal{M}_{i,2}$ outputs $null$. In this case, $\epsilon_{i,2}$ must be considered *twice*; once for noisy $\mathbf{c}_i$ in the intruded main memory, and once as part of ϵ_{rm} at the next timestamp a publication occurs. The value of $\epsilon_{i,2}$ is $\epsilon_{rm}/3 \leq (\epsilon/2)/3 = \epsilon/6$. Therefore, the budget spent by the publication sub mechanisms in a window where a single unannounced intrusion occurs is at most $\epsilon/6+\epsilon/3 = \epsilon/2$, and Theorem 4 holds. The utility of BD degrades by a factor of 1.5, because every publication uses 1.5 times less budget ($\epsilon_{rm}/3$ versus $\epsilon_{rm}/2$).

On the other hand, in BA, we set the initial budget $\epsilon_{i,2}$ at timestamp i as $\epsilon/(4 \cdot w)$ instead of $\epsilon/(2 \cdot w)$. We can prove in the same fashion as Theorem 5 that the sum of budgets per any window where the intrusion did not occur is at most $\epsilon/4$. For any window where the intrusion took place, we must add the maximum budget that can be absorbed by a timestamp, which is $(w-1)\cdot(\epsilon/(4\cdot w)) < \epsilon/4$. Hence, the total publication budget in the window becomes at most $\epsilon/2$, and w-event privacy is retained. The utility degrades by a factor of 2, since we halve $\epsilon_{i,2}$ as compared to the original scheme. A final remark concerns an intrusion during a nullified publication. This is handled by discarding D_i from the communication channel when i corresponds to a nullified publication. In this case, BA constructs no private information on D_i and, thus, spends no budget.

We next turn to the setting of multiple announced intrusions. Similar to the case of a single intrusion, BD and BA keep only noisy versions of dis an $\mathbf{c}_i$ in main memory, and $\mathcal{M}_{i,1}$ remains intact. However, here BD and BA do not alter $\lambda_{i,2}$. The reason is that here the curator always knows when an intrusion takes place, and adjusts the behavior of $\mathcal{M}_{i,2}$ on-the-fly. Specifically, when it detects an intrusion, it simply *forces* the publication to occur (i.e., behaves as if the publication had to occur). The schemes never use a timestamp's budget more than once, and the privacy proofs remain identical to those in the original schemes. These intrusions considerably affect the utility of the schemes. The full analysis is rather complex, and we defer it to the long version of this work.

5. EXPERIMENTAL EVALUATION

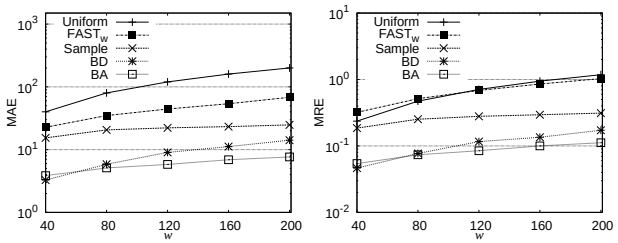
We compared BD and BA with benchmarks FAST$_w$, Uniform and Sample over real datasets. We implemented FAST$_w$ in Java, using the Fan et al. [17] implementation of FAST[1], and configured it according to [17]. We implemented all other methods also in Java, and fine-tuned BD and BA for every experiment according to the optimizations of Section 4.5. We conducted the experiments on a machine with Intel Core i5 CPU 2.53GHz and 4GB RAM, running Windows 7. We ran each experiment 100 times, and reported the average error, expressed as the Mean Absolute Error (MAE) and the Mean Relative Error (MRE). We fixed $\epsilon = 1$.

We experimented with two real datasets. For the first, we connected our system to an actual online traffic monitoring service in Rome [2], and we periodically retrieved the data *real-time* as they were generated by the service. We executed our schemes for 10 consecutive days, where every data collection period was 10 minutes (i.e., our time domain consisted of 1442 timestamps). At each timestamp i, we directly retrieved vector $\mathbf{c}_i$, where element $\mathbf{c}_i[j]$ is the number of commuters on road j at i. The total number of roads (and, hence, the size of $\mathbf{c}_i$) is $d = 25,936$. Note that, in such a scenario, our schemes protect any single trajectory of any user consisting of up to w road segments.

For the second, we created a stream from a well-known archived dataset, namely World Cup[2]. The latter contains 1,352,804,107 Web server logs from the FIFA 1998 Soccer World Cup website, gathered in 88 consecutive days. Each log entry consists of a client ID, the ID of the requested URL, the size of the response, etc. This dataset does not define a specific period length for each timestamp and, hence, we regard the number of timestamps per day as a variable parameter under investigation. At every timestamp i, we collected from the stream a database D_i, where each row is a unique log, every attribute is a URL, and cell (u, j) is 1 if log u accessed URL j, and 0 otherwise. Moreover, at each timestamp i, we calculated vector $\mathbf{c}_i$ such that $\mathbf{c}_i[j]$ is the number of logs accessing URL j at i. The number of unique URLs (and, hence, the size of $\mathbf{c}_i$) is $d = 89,997$. Our schemes protect any single sequence of URL accesses of a user over at most w timestamps (given that each user browses at most a single URL per timestamp).

Varying parameter w. Figure 6 illustrates (in logarithmic scale) the MAE and MRE of our schemes as a function of w for the Rome dataset. We vary w between 40 and 200. BA is the best method. It outperforms Uniform and FAST$_w$ by up to one order of magnitude in both MAE and MRE. It is up to five (four) times better than Sample in MAE (MRE). Moreover, its MAE (MRE) is 46% (35%) smaller than in BD. Finally, its MRE varies in range 5-11%.

The budget in Uniform is fixed for all timestamps, and linear in w. Hence, its performance deteriorates significantly for large w. FAST$_w$ behaves similarly to Uniform, because it uses a fixed number of samples, assigning a uniform budget over all samples. As w is raised, the samples increase and, thus, a smaller budget is available for the publications. FAST$_w$ outperforms Uniform in MAE, but features a similar performance in MRE. This is because FAST$_w$ approximates well the large counts (which dominate the MAE), but does not equally benefit the small counts, whose collective MRE becomes dominant in the total MRE. Sample performs better than FAST$_w$ and Uniform, and is relatively unaffected by w, mainly because (i) it utilizes more budget per publication, and (ii) the statistics do not vary a lot across timestamps and, thus, each sample approximates relatively well the counts in subsequent timestamps. BA and BD are superior due to their data-dependent sampling and approximation (as opposed to Sample), as well as sophisticated budget allocation (as opposed to all benchmarks). For small w, they have a similar performance. However, as w increases, the performance gap between BA and BD increases as well. This is due to the combination of (i) the exponential decay of budget in BD as more publications occur in a larger window, and (ii) the fact that a larger window prevents BD from recycling budget. On the contrary, BA allocates the budget over the multiple publications in the larger window more effectively, via the initial uniform budget distribution and the subsequent dynamic budget absorption.



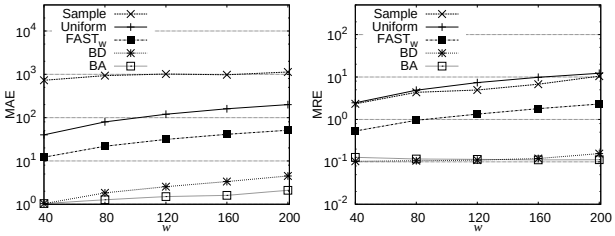
(a) MAE (b) MRE

Figure 6: Error vs. w (Rome dataset)

[1] http://www.mathcs.emory.edu/~lfan3/FAST/tool/

[2] http://ita.ee.lbl.gov/html/contrib/WorldCup.html

Figure 7 illustrates the MAE and MRE of our mechanisms versus w for the World Cup dataset. We fix the number of publications per day to 15 (which translates to an experiment spanning 1320 timestamps), and vary w between 40 and 200. BA outperforms Uniform by approximately two orders of magnitude in MAE and MRE, Sample by up to three (two) orders of magnitude in MAE (resp. MRE), FAST_w by more than one order of magnitude in both MAE and MRE, and BD by up to 51% in MAE and 30% in MRE. Moreover, the MRE in BA is always below 11.8%. Sample is always worse than Uniform in terms of MAE. The reason is that the statistics have great fluctuations in successive timestamps in this dataset, inducing a huge approximation error to Sample, which is higher than that of Uniform even for large values of w. The difference between Sample and Uniform in MRE is marginal. This is because the largest approximation error in Sample occurs in the most popular URLs. This greatly affects the total error in absolute terms, but does not impact it in relative terms to the same extent. Finally, contrary to the case of the Rome dataset, FAST_w outperforms Uniform even in MRE. This is because the sampling of FAST_w approximates more effectively the highly variable counts of the World Cup dataset across time.



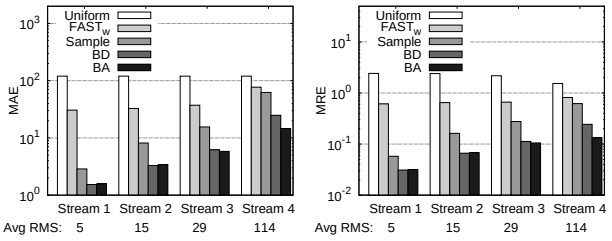
(a) MAE (b) MRE

Figure 7: Error vs. w (World Cup dataset)

The effect of the magnitude of change. The next set of experiments evaluates the effectiveness of our schemes, when the incoming updates change the statistics over time with various magnitudes. We derived datasets of variable magnitude from our two real datasets. In the Rome dataset, we grouped the database columns based on the average *Root Mean Square* (RMS) of their counts over the entire stream (the RMS is a standard statistical tool for measuring the magnitude of change of a varying quantity). In the World Cup dataset, we varied the number of timestamps per day, as this adjusts the magnitude of change; the larger the number of timestamps, the fewer updates per timestamp and, hence, the smoother the change in the statistics in successive timestamps.

Figure 8 presents our results for the Rome dataset. We partitioned the stream into four sub streams. Each sub stream focused on a disjoint subset of roads. The reported counts of the roads in the same sub stream featured a similar RMS. We ran every mechanism on each sub stream independently, and reported its MAE and MRE, setting $w = 120$. For each sub stream in the x-axis, we also include the average RMS of the counts it generates (smaller RMS values indicate smaller magnitudes of change). BA and BD are always superior to the benchmarks, once again due to their more effective budget allocation. BA is the most robust scheme to RMS, for both MAE (Figure 8(a)) and MRE (Figure 8(b)). For small RMS values, BD and BA have comparable performance, featuring excellent utility (about 3% MRE when RMS=5 in Stream 1). The reason is that the small magnitude of change enables the accurate approximation of the current statistics with an older release. They also handle larger magnitude of change better than the benchmarks and continue to be effective even for RMS=29 in Stream 3 (about 11% MRE). For the largest RMS value (114 in Stream 4), BD does not perform well compared to BA (24% vs. 13% MRE) because the large magnitude of change results in numerous publications per window, which greatly impact BD. On the contrary, BA manages to attain excellent performance even for a large RMS, due to its ability to allocate the privacy budget more effectively than BD when many publications occur per window.

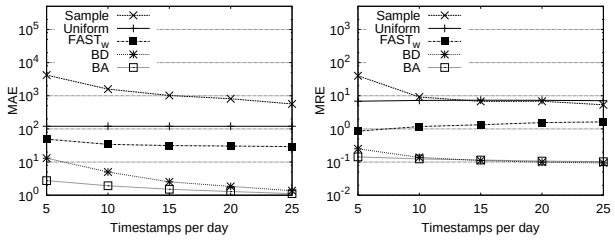
On the other hand, Uniform always performs badly, as it publishes the current statistics with large noise independently of the RMS. FAST_w is also relatively unaffected by small RMS values, and only for extreme RMS values does its error increase noticeably. Sample outperforms FAST_w in small RMS values. This is because very few samples are needed per window to effectively approximate the skipped statistics, and FAST_w unnecessarily allocates budget to a fixed number of samples, which is always larger than that of Sample. However, Sample rapidly deteriorates as RMS increases, since a steep count change over time leads to an excessive approximation error.



(a) MAE (b) MRE

Figure 8: Error vs. RMS (Rome dataset)

Figure 9 plots the MAE and MRE of all schemes for the World Cup dataset, where we vary the timestamps per day and set $w = 120$. Note that every value in the x-axis corresponds to a different dataset. A smaller number of timestamps per day translates to more abrupt changes in consecutive statistics. The performance trends of all mechanisms are similar to those in Figure 8, i.e., the error increases for larger magnitudes of change. Our schemes outperform the baselines form one to up to three (two) orders of magnitude in MAE (resp. MRE). The errors of BA and BD become marginally close as the number of timestamps increases, whereas their MRE ranges in 10-14% and 10-25%, respectively.



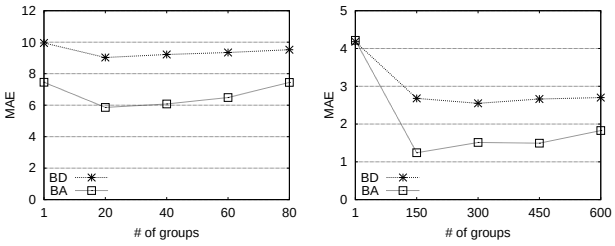
(a) MAE (b) MRE

Figure 9: Error vs. timestamps per day (World Cup dataset)

Effect of column partitioning. Figures 10(a) and 10(b) assess the effect of the column partitioning optimization for BD and BA (see Section 4.5) on the Rome and World Cup datasets, respectively, when varying the number of groups, and setting $w = 120$. We decomposed the database columns into disjoint groups, ran our schemes independently on every group, and reported the average MAE over all groups. As the mechanisms received updates from the stream, they periodically modified the groups, such that each

group contained columns whose counts feature a similar magnitude of change (measured as the RMS).

In Figure 10(a), BD and BA exhibit the best performance when the number of groups is 20. BD improves its MAE by up to 10%, and BA by up to 21%, as compared the case of the single group. Observe that the error increases when the number of groups becomes large after its initial improvement in both schemes. When the number of groups increases, the number of attributes in each group decreases, resulting in higher sensitivity for computing dis in $\mathcal{M}_{i,1}$. Thus, more budget is assigned to the dissimilarity calculation sub mechanism, leaving less budget for publishing. Hence, a large number of groups results in higher error. In Figure 10(b), the error curves feature similar trends as in Figure 10(a), i.e., the number of groups initially improves the performance of both methods, but then adversely impacts it for the same reasons explained for the Rome dataset. BD and BA exhibit the best performance when the number of groups is 150. BD improves its MAE by up to 36%, and BA by up to 71%, as compared the case of the single group.



(a) Rome (b) World Cup

Figure 10: Error vs. number of groups

Summary. Our experimental evaluation demonstrated the superiority of BD and BA over the benchmark methods, and their practicality in two real datasets. Specifically, our novel schemes outperformed the baselines approaches by orders of magnitude in the majority of our settings, in both absolute and relative error. Moreover, BA outperformed BD in all experiments, due to its more effective budget allocation when several publications must occur per window. The error in BA was up to about 50% smaller than that of BD in all scenarios. Furthermore, the relative error of BA ranged in 3-14% in all settings and datasets. This renders BA practical for data mining tasks, and confirms the viability of the w-event privacy concept in real-life applications. Finally, we demonstrated the performance boost achieved by our column partitioning optimization.

6. CONCLUSION

We introduced the novel notion of *w-event ϵ-differential privacy* in privacy-preserving statistics publishing on infinite streams. This concept protects a sequence of w events occurring in successive time instants, and finds practical application in numerous scenarios where sensitive information can be inferred from user activity spanning over a time window. We formulated a sliding window methodology for satisfying this property, and designed three benchmark methods. We then introduced two novel mechanisms that are based on sophisticated sampling and dynamic privacy budget allocation techniques, and outlined several optimizations. We conducted thorough experimentation with real datasets, which demonstrated the superiority of our mechanisms against the benchmarks, and the practicality of our novel privacy notion in real applications.

Acknowledgements

Georgios Kellaris and Dimitris Papadias were supported by grant 618011 from Hong Kong RGC.

7. REFERENCES

[1] HCUP net. http://hcupnet.ahrq.gov/.
[2] informaservizi.it. http://apprendistato.informaservizi.it/.
[3] People Talking About This. www.insidefacebook.com/2012/01/10/people-talking-about-this-defined/.
[4] J. Bolot, N. Fawaz, S. Muthukrishnan, A. Nikolov, and N. Taft. Private decayed predicate sums on streams. In *ICDT*, 2013.
[5] J. Cao, Q. Xiao, G. Ghinita, N. Li, E. Bertino, and K.-L. Tan. Efficient and accurate strategies for differentially private sliding window queries. In *EDBT*, 2013.
[6] T. Chan, E. Shi, and D. Song. Private and continual release of statistics. *TISSEC*, 14(3):26, 2011.
[7] T.-H. H. Chan, M. Li, E. Shi, and W. Xu. Differentially private continual monitoring of heavy hitters from distributed streams. In *PETS*, 2012.
[8] G. Cormode, C. M. Procopiuc, D. Srivastava, E. Shen, and T. Yu. Differentially private spatial decompositions. In *ICDE*, 2012.
[9] G. Cormode, C. M. Procopiuc, D. Srivastava, and T. T. L. Tran. Differentially private publication of sparse data. In *ICDT*, 2012.
[10] C. Dwork. Differential privacy. In *ICALP*, 2006.
[11] C. Dwork. Differential privacy in new settings. In *SODA*, 2010.
[12] C. Dwork. A firm foundation for private data analysis. *CACM*, 54(1):86–95, 2011.
[13] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor. Our data, ourselves: Privacy via distributed noise generation. In *EUROCRYPT*, 2006.
[14] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, 2006.
[15] C. Dwork, M. Naor, T. Pitassi, G. Rothblum, and S. Yekhanin. Pan-private streaming algorithms. In *ICS*, 2010.
[16] C. Dwork, M. Naor, T. Pitassi, and G. N. Rothblum. Differential privacy under continual observation. In *STOC*, 2010.
[17] L. Fan and L. Xiong. Real-time aggregate monitoring with differential privacy. In *CIKM*, 2012.
[18] M. Hardt and K. Talwar. On the geometry of differential privacy. In *STOC*, 2010.
[19] M. Hay, V. Rastogi, G. Miklau, and D. Suciu. Boosting the accuracy of differentially private histograms through consistency. *PVLDB*, 3(1-2):1021–1032, 2010.
[20] K. Jiang, D. Shao, S. Bressan, T. Kister, and K.-L. Tan. Publishing trajectories with differential privacy guarantees. In *SSDBM*, 2013.
[21] C. Li and G. Miklau. Optimal error of query sets under the differentially-private matrix mechanism. In *ICDT*, 2013.
[22] N. Li, W. Qardaji, D. Su, and J. Cao. PrivBasis: Frequent itemset mining with differential privacy. *PVLDB*, 5(11):1340–1351, 2012.
[23] Y. D. Li, Z. Zhang, M. Winslett, and Y. Yang. Compressive mechanism: Utilizing sparse representation in differential privacy. In *WPES*, 2011.
[24] A. Machanavajjhala, D. Kifer, J. Abowd, J. Gehrke, and L. Vilhuber. Privacy: Theory meets practice on the map. In *ICDE*, 2008.
[25] F. McSherry. Privacy Integrated Queries: An Extensible Platform for Privacy-preserving Data Analysis. In *SIGMOD*, 2009.
[26] F. McSherry and K. Talwar. Mechanism design via differential privacy. In *FOCS*, 2007.
[27] D. Mir, S. Muthukrishnan, A. Nikolov, and R. N. Wright. Pan-private algorithms via statistics on sketches. In *PODS*, 2011.
[28] K. Nissim, S. Raskhodnikova, and A. Smith. Smooth sensitivity and sampling in private data analysis. In *STOC*, 2007.
[29] V. Rastogi and S. Nath. Differentially private aggregation of distributed time-series with transformation and encryption. In *SIGMOD*, 2010.
[30] A. Roth and T. Roughgarden. Interactive privacy via the median mechanism. In *STOC*, 2010.
[31] X. Xiao, G. Bender, M. Hay, and J. Gehrke. iReduct: Differential privacy with reduced relative errors. In *SIGMOD*, 2011.
[32] X. Xiao, G. Wang, and J. Gehrke. Differential privacy via wavelet transforms. *TKDE*, 23(8):1200–1214, 2011.
[33] C. Zeng, J. F. Naughton, and J.-Y. Cai. On differentially private frequent itemset mining. *PVLDB*, 6(1):25–36, 2012.