DP-STabS: **Differentially Private Synthetic Tabular Stream**

Anonymous Author(s)

ABSTRACT

[GK: TODO]

KEYWORDS

differential privacy, stream, tabular

1 INTRODUCTION

2 PROBLEM SETUP

Notations:

- $[t]$: the set $\{1, 2, \ldots, t\}$ for any $t \in \mathbb{N}$
- $[t_1 : t_2]$: the set $\{t_1, t_1 + 1, \ldots, t_2\}$ for any $t_1, t_2 \in \mathbb{N}$ with $t_1 \leq t_2$
- $\mathbb{N}_+$: $\mathbb{N} \cup \{0\}$

Let $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2 \times \ldots \times \mathcal{X}_p$ denote a space of p dimensional points such that $\mathcal{X}_i$ is a discrete space of size $|\mathcal{X}_i|$ for any $i \in [p]$. For example, if each data point results from a survey of p boolean questions, then $\mathcal{X} = \{0, 1\}^p$. Let $f : \mathbb{N} \times \mathcal{X} \to \mathbb{N}_+$ be an input stream where $f(t, x)$ denotes the count of point x at time t. We provide an algorithm that generates a privacy-preserving synthetic data stream $g : \mathbb{N} \times \mathcal{X} \to \mathbb{N}_+$ that accurately represents the input stream f. We define the terms "streaming", "privacy-preserving" and "accurately" rigorously in the subsequent subsections.

For any $N \subseteq \mathbb{N}$, we will use the notation f_N to denote the restriction of the stream f to the time indices in the set N, that is $f_N : N \times \mathcal{X} \to \mathbb{N}_+$ such that $f_N(t, x) = f(t, x)$ for all $t \in N$ and $x \in \mathcal{X}$. Similarly, for any time $t \in \mathbb{N}$, $f_t : \mathcal{X} \to \mathbb{N}_+$ denotes a restriction of f to time t such that $f_t(x) = f(t, x)$ for all $x \in \mathcal{X}$. We refer to a mapping from $\mathcal{X}$ to $\mathbb{N}_+$ such as f_t as a *dataset* and denote the size of a dataset (the total count of all points) as $|f_t|$, that is, $|f_t| = \sum_{x \in \mathcal{X}} f_t(x)$. Let $\mathbb{N}_+^{\mathcal{X}}$ denote the set of all datasets over $\mathcal{X}$.

2.1 Streaming algorithm

Given the input stream f, an algorithm $\mathcal{A}$ is said to be streaming if for the output stream g, at any time $t \in \mathbb{N}$ it maps $f_{[t]}$ to g_t, that is, $g(t, x) := \mathcal{A}(f_{[t]})(t, x)$ for all $x \in \mathcal{X}$.

2.2 Differential Privacy

We use the notion of Differential Privacy (DP) for robust privacy guarantees by the algorithm. In a nutshell, an algorithm that satisfies DP is robust in the probability of observing an output when the input data is changed by a small amount. To this end, we need to define the concept of change in a data stream.

This work is licensed under the Creative Commons Attribution 4.0 International License. To view a copy of this license visit https://creativecommons.org/licenses/by/4.0/ or send a letter to Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.
Proceedings on Privacy Enhancing Technologies YYYY(X), 1–5
© YYYY Copyright held by the owner/author(s).
https://doi.org/XXXXXXX.XXXXXXX

Similar to [3], we keep track of the change of data stream f over time, via a differential stream defined as,

$$\nabla f(t, x) = f(t, x) - f(t - 1, x), \quad t \in \mathbb{N}, \tag{1}$$

assuming $f(0, x) = 0$. Furthermore, the total change of f over all times and locations is the quantity

$$\|f\| := \sum_{t \in \mathbb{N}} \sum_{x \in \mathcal{X}} |\nabla f(t, x)|. \tag{2}$$

Two streams, f and $\tilde{f}$ are neighbours if they satisfy

$$\|f - \tilde{f}\| = 1. \tag{3}$$

Definition 2.1 (Differential privacy). A randomized streaming algorithm $\mathcal{A}$ that takes as an input a data stream is ε-differentially private if for any two neighboring data streams f and $\tilde{f}$, and for any measurable set of outputs S,

$$\mathbb{P}\{\mathcal{A}(\tilde{f}) \in S\} \leq e^{\varepsilon} \cdot \mathbb{P}\{\mathcal{A}(f) \in S\} \tag{4}$$

2.3 Marginals and accuracy

In this work, we measure the accuracy of our output stream using marginal queries.

Definition 2.2 (k-way marginal query). A k-way marginal query $q : \mathcal{X} \to \{0, 1\}$ is a mapping defined by a tuple $q_c = (c_1, c_2, \ldots, c_k)$ of k column indices and their corresponding values $q_v = (v_1, v_2, \ldots, v_k)$ such that $v_i \in \mathcal{X}_{c_i}$ for all $i \in [k]$ and the mapping is defined as,

$$q(x) = \prod_{i=1}^{k} \left(\mathbb{1}_{\{x_{c_i} = v_i\}} \right), \tag{5}$$

for any $x \in \mathcal{X}$.

With a slight abuse of notation, we extend the definition of marginal query from points to datasets as follows.

Definition 2.3 (k-way marginal query for a dataset). A k-way marginal query q can be extended to any dataset $h : \mathcal{X} \to \mathbb{N}_+$ as,

$$q(h) = \sum_{x \in \mathcal{X}} h(x) q(x) \tag{6}$$

Definition 2.4 (Accuracy of a dataset). A synthetic dataset $g : \mathcal{X} \to \mathbb{N}_+$ is said to be (α, β) accurate in representing a given dataset $f : \mathcal{X} \to \mathbb{N}_+$ with respect to a set of marginal queries Q if,

$$\mathbb{P}\left\{ \max_{q \in Q} |q(g) - q(f)| \geq \alpha \right\} \leq \beta. \tag{7}$$

We use this notion of the accuracy of a dataset as a base when defining the accuracy of a stream. In Definition 2.5 and 2.6 we present two separate notions of the accuracy of a stream. Note that α and β in the definitions may be a function of time t.

[GK: TODO: verify below definitions]

Definition 2.5 (Differential accuracy of stream). A synthetic stream g is said to have a differential accuracy of (α, β) in representing a given data stream f, with respect to marginal queries Q, if at each time $t \in \mathbb{N}$, ∇g_t is (α, β) accurate in representing ∇f_t.

Definition 2.6 (Snapshot accuracy of a stream). A synthetic stream g is said to have a snapshot accuracy of (α, β) in representing a given data stream f, with respect to a set of marginal queries Q, if at each time $t \in \mathbb{N}$, g_t is (α, β) accurate in representing f_t.

Lemma 2.7. *(Differential accuracy implies snapshot accuracy) If a synthetic stream g has a differential accuracy of $(\alpha_t(\beta), \beta)$ at any time $t \in \mathbb{N}$, then it has a snapshot accuracy of $\left(\sum_{i=1}^{t} \alpha_i\left(\frac{\beta}{t}\right), \beta\right)$ at time t.*

Proof. Taking a union bound at any time $t \in \mathbb{N}$, we have that with probability at least $1 - \beta t$,

$$\max_{q\in Q}\left|q\left(\nabla g_i - \nabla f_i\right)\right| \leq \alpha_i(\beta) \text{ for all } i \in [t]$$

$$\implies \max_{q\in Q}\left|q(g_t) - q(f_t)\right| = \max_{q\in Q}\left|q\left(\sum_{i=1}^{t}\nabla g_i\right) - q\left(\sum_{i=1}^{t}\nabla f_i\right)\right|$$

$$= \max_{q\in Q}\left|\sum_{i=1}^{t}\left(q\left(\nabla g_i - \nabla f_i\right)\right)\right| \leq \max_{q\in Q}\sum_{i=1}^{t}\left|\left(q\left(\nabla g_i - \nabla f_i\right)\right)\right|$$

$$\leq \sum_{i=1}^{t}\max_{q\in Q}\left|\left(q\left(\nabla g_i - \nabla f_i\right)\right)\right| \leq \sum_{i=1}^{t}\alpha_i(\beta)$$

Hence, the algorithm has a snapshot accuracy of $\left(\sum_{i=1}^{t}\alpha_i(\beta), \beta t\right)$ or reparametrizing to give $\left(\sum_{i=1}^{t}\alpha_i\left(\frac{\beta}{t}\right), \beta\right)$. □

3 DIFFERENTIAL PRIVACY PRILIMINARIES

In this section, we present some differentially private algorithms that we will use as sub-routines and will serve as building blocks for our algorithm DP-STabS. We also distinguish whether the algorithm is typically used for offline (one-time) or stream data input.

3.1 Laplace Mechanism

The Laplace Mechanism is the most fundamental algorithm in differential privacy for generating answers to queries evaluated over datasets.

Definition 3.1 (Laplace Mechanism). Let $q : \mathbb{N}_+^{\mathcal{X}} \to \mathbb{R}^d$ be a function we are interested in and define its sensitivity Δ_q as,

$$\Delta_q := \max_{\substack{f,\tilde{f}\in\mathbb{N}_+^{\mathcal{X}},\\ \left\|f-\tilde{f}\right\|=1}} \left\|q(f) - q(\tilde{f})\right\|_{\ell_1}. \tag{8}$$

Then, a randomized algorithm $\mathcal{A}$ defined as

$$\mathcal{A}(f) := q(f) + \left(\text{Lap}_1\left(\frac{\Delta_q}{\varepsilon}\right), \text{Lap}_2\left(\frac{\Delta_q}{\varepsilon}\right), \ldots, \text{Lap}_d\left(\frac{\Delta_q}{\varepsilon}\right)\right), \tag{9}$$

satisfies ε-differential privacy. Here, $\text{Lap}_{(\cdot)}$ are independent Laplace random variables centered at 0 and with variance scale $\frac{\Delta_q}{\varepsilon}$. [GK: TODO: is there a better way to write Lap random variables?]

3.2 Differentially Private Selection: Exponential Mechanism

Let $\mathcal{R}$ be a finite set. Let $u : N_+^{\mathcal{X}} \times \mathcal{R} \to \mathbb{R}$ be a function such that $u(f, r)$ denotes the utility of an element $r \in \mathcal{R}$ for the dataset $f \in N_+^{\mathcal{X}}$. Given a dataset f, we want to find the element in $\mathcal{R}$ with maximum utility while preserving differential privacy. [GK: needs example or more details?]

Definition 3.2 (Exponential Mechanism). Let Δ_u denote the sensitivity of the utility function defined as,

$$\Delta_u := \max_{\substack{f,\tilde{f}\in\mathbb{N}_+^{\mathcal{X}},\\ \left\|f-\tilde{f}\right\|=1}} \max_{r\in\mathcal{R}}\left|u(f, r) - u(\tilde{f}, r)\right|. \tag{10}$$

Then, an algorithm $\mathcal{A} : \mathbb{N}_+^{\mathcal{X}} \to \mathcal{R}$ such that, for all $r \in \mathcal{R}$,

$$\mathbb{P}\left\{\mathcal{A}(f) = r\right\} \propto \exp\left(-\frac{\varepsilon}{2\Delta_u}u(f, r)\right)$$

satisfies ε-differential privacy.

3.3 Tabular Synthetic Data Generation: Select, Measure, Learn, and Iterate

Consider the task of generating synthetic tabular data when the dataset is available at once (offline). Let $f \in \mathbb{N}_+^{\mathcal{X}}$ be a dataset over $\mathcal{X}$. Assume we are interested in generating a synthetic dataset $g \in \mathbb{N}_+^{\mathcal{X}}$ that is accurate for marginal queries Q. A straightforward approach to generating the synthetic dataset would be: (1) generate a differentially private measurement for all queries using the Laplace Mechanism as

$$m = \left(q(f) + \text{Lap}\left(\frac{\Delta q}{\varepsilon \cdot |Q|}\right)\right)_{q\in Q};$$

(2) find a dataset that minimizes the average error over the query set by solving the following optimization problem,

$$\arg\min_{g\in\mathcal{N}_+^{\mathcal{X}}} \frac{1}{|Q|}\sum_{q\in Q}\left|m_q - q(g)\right|. \tag{11}$$

There are however two key problems with this approach: (1) the size of the query set is typically polynomial in the dimension p which leads to a very small budget for answering an individual query, that is a large amount of noise is added in Laplace Mechanism, and (2) the optimization problem in Equation 11 is a high-dimensional discrete optimization problem which is NP-Hard and cannot be solved in time polynomial in dimension p. Many existing algorithms thus circumvent the above two problems by: (1) measuring only a subset of the queries in Q which have the largest error, and (2) approximating the optimization problem in Equation 1. We refer the reader to [4] for further discussions on the various approximations of Equation 11. For now, we assume that there exists an algorithm $\mathcal{A}_{Dataset}$ which takes as input a set of queries and their corresponding measurements and returns a dataset in $\mathcal{N}_+^{\mathcal{X}}$. [GK: TODO: add details for MW and PGM in Appendix and refer here.]

Algorithm 1 is a meta-algorithm that is based on the above idea and falls in a "select, measure, optimize, and iterate" paradigm. The algorithm has a fixed number of iterations k. In each iteration i, the following happens: (1) while upholding differential privacy,

we select a query q_{l_i} that performs the worst on the current synthetic dataset h_{i-1}; (2) an approximation of the value of this query is generated as m_{l_i} using the Laplace Mechanism; and finally (3) the dataset is updated from h_{i-1} to h_i by using the sub-routine $\mathcal{A}_{Dataset}$.

Algorithm 1 Meta algorithm: generating differentially private synthetic tabular dataset

1: **Input:** Given dataset f, an ordered set of queries Q, privacy budget ε, a differentially private selection mechanism $\mathcal{A}_{Select}$, a subroutine $\mathcal{A}_{Dataset}$ to find a dataset given noisy query measurements, and the number of iterations k.
2: **Output:** A dataset $g \in \mathbb{N}_+^{\mathcal{X}}$.
3: Create a dataset $h_0 \in \mathbb{N}_+^{\mathcal{X}}$ with $h_0(x) = 1$ for all $x \in \mathcal{X}$.
4: Set $M \leftarrow \emptyset$ as a set of selected queries and their measurements.
5: **for** $i = 1, 2, \ldots, k$ **do**
6: Set $e_i \leftarrow \left(\left|q(h_{i-1}) - q(f)\right|\right)_{q \in Q}$ as the error in queries.
7: **Select:** $l_i \leftarrow \mathcal{A}_{Select}(e_i, 2/\varepsilon)$, an index of query.
8: **Measure:** $m_{l_i} \leftarrow q(f) + \text{Lap}\left(2\Delta_{q_i}/\varepsilon\right)$, value of query.
9: Set $M \leftarrow M \cup \{(q_{l_i}, m_{l_i})\}$; add selected query and its value.
10: **Optimize:** Dataset $h_i \leftarrow \mathcal{A}_{Dataset}(M)$.
11: **end for**

3.4 Counters and Selective Counting

In the previous subsections, we discussed some DP algorithms over datasets. In this subsection, we discuss DP algorithms for streams, and in particular we discuss the counter algorithms. In our final algorithm, we will measure a query $q \in Q$ at multiple instances over time. Thus we need an algorithm that can do this measurement over an input stream accurately and privately. We use the notion of counters introduced in [2] and [1]. The below definition of a counter is borrowed from [3].

Definition 3.3. An (α, β)-accurate counter C is a randomized streaming algorithm that estimates the sum of an input stream of values $f : \mathbb{N} \to \mathbb{R}$ and maps it to an output stream of values $g : \mathbb{N} \to \mathbb{R}$ such that for each time $t \in \mathbb{N}$,

$$\mathbb{P}\left\{\left|g(t) - \sum_{t' \le t} f(t')\right| \le \alpha(t, \delta)\right\} \ge 1 - \beta,$$

where the probability is over the randomness of C and β is a small constant.

We will be using the Simple II and Log Binary Tree counters from [1], hereafter referred to as Simple and Binary Tree Counters respectively (see Appendix **??** for further details). Moreover, based on the Two Level Counter in [1], we also introduce a new counter in this work termed Unbounded Block counter in Section **??**.

3.4.1 Selective Counting. Algorithm 4 (Online Selective Counting) in [3] provides a differentially private and streaming meta-algorithm that only updates a selected subset of counters at any time. Our algorithm DP-STabS is an instance of this algorithm and inherits its privacy guarantees.

4 PROPOSED METHODS

In this section, we first propose a baseline method in Algorithm 2 and then improve upon it to give our final algorithm in Algorithm 3.

4.1 Baseline

Let $\mathcal{A}$ be any differentially private algorithm for generating a synthetic dataset given an input true dataset. In Algorithm 2 we provide a general framework that converts $\mathcal{A}$ into a differentially private streaming algorithm for producing synthetic stream. Given an input stream f, at any time t, Algorithm 2 simply runs an independent instance of algorithm $\mathcal{A}$ on the differential dataset at time t, that is ∇f_t and produces the differential synthetic dataset ∇g_t.

Algorithm 2 Baseline: from differentially private dataset to stream

1: **Input:** An input data stream f, a differentially private algorithm $\mathcal{A}$ to generate synthetic datasets, the privacy budget ε
2: **Output:** A synthetic stream g.
3: Initialize $g(0, x) = 0$ for all $x \in \mathcal{X}$.
4: **for** $t = 1, 2, \ldots$ **do**
5: Set $h_t \leftarrow \mathcal{A}(\nabla f_t, \varepsilon)$
6: Set $g(t, x) = g(t-1, x) + h_t(x)$ for all $x \in \mathcal{X}$
7: Release g_t.
8: **end for**

Proof of privacy of Basline Algorithm 2. Let $\mathcal{A}^+$ denote the Algorithm 2. For neighboring data streams f and $\tilde{f}$ satisfying Equation 3, there exists almost one $\tau \in \mathbb{N}$ and $x \in \mathcal{X}$ such that $\nabla f(\tau, x) \neq \nabla \tilde{f}(\tau, x)$. Since $\mathcal{A}$ is ε-DP, it follows that for any output h_τ,

$$\mathbb{P}\left\{\mathcal{A}(\nabla f_\tau, \varepsilon) = h_\tau\right\} \le e^\varepsilon \cdot \mathbb{P}\left\{\mathcal{A}(\nabla \tilde{f}_\tau, \varepsilon) = h_\tau\right\}.$$

Since at any time $t \neq \tau$, the input streams f and ∇f are identical, it follows that,

$$\mathbb{P}\left\{\mathcal{A}^+(f, \mathcal{A}, \varepsilon) = g\right\} \le e^\varepsilon \cdot \mathbb{P}\left\{\mathcal{A}^+(\tilde{f}, \mathcal{A}, \varepsilon) = g\right\}.$$

Hence, Baseline Algorithm 2 is ε-DP. □

4.2 Main Algorithm DP-STabS

4.2.1 Outline. In a nutshell, our algorithm also follows the "select, measure, optimize, and iterate" paradigm described in Algorithm 1. At any time $t \in \mathbb{N}$, the goal is to ensure that f_t and g_t are close to each other as evaluated using the queries Q. We start with a dataset $h_{t,0} = g_{t-1}$ and update it over k iterations from $h_{t,0}, h_{t,1}, \ldots,$ to $h_{t,k}$. At any iteration $l \in [k]$, we select a query index $\eta_{t,l} \in [|Q|]$ for which our dataset $h_{t,l-1}$ has approximately the highest error when compared to f_t. We will discuss how exactly this selection is done soon, but for now, let us accept it as a black-box. At the end of the k iterations, g_t is set to some aggregate of the datasets $h_{t,1}, h_{t,2}, \ldots,$ and $h_{t,k}$. We present our proposed method as a meta-algorithm DP-STabS in Algorithm 3.

4.2.2 Measure. Let $m : \mathbb{N} \times [|Q|] \to \mathbb{R}$ be a map such that $m(t, i)$ denotes our differentially private approximation of $q_i(f_t)$, that is the value of query $q_i \in Q$ at time t. Since a single query may be selected at multiple time instances, we use a counter algorithm to measure the value of the query efficiently over time. We associate each query

in Q with an instance of some counter Algorithm, say $\mathcal{A}_{Counter}$. Consider a query $q_i \in Q$ and let C_i be its corresponding counter. We use the notation $C_i(t)$ to conveniently refer to the value of the counter C_i at time t. Let $N_i(t) \subseteq [t]$ be the time instances until time t when the query q_i was selected to be measured using the true data. Also, let $\bar{N}_i(t) := [t] \setminus N_i(t)$ be the time instances until time t at which query q_i was not selected. Then the output $C_i(t)$ of the counter algorithm is based solely on the stream $\nabla f_{N_i(t)}$.

However, to generate the dataset g_t we need an approximate measurement of the value $q_i(f_t)$. In other words, we are missing the measurement of the query on times $\bar{N}_i(t)$ when the index i was not selected. At any such time $\tau \in \bar{N}_i(t)$, since q_i was not selected, we assume that the query value $q_i(g_\tau)$ on the synthetic dataset g_τ is close to the true value $q_i(f_\tau)$. We create a map $r : \mathbb{N} \times [|Q|] \to \mathbb{R}$ such that $r(t, i)$ denotes our differentially private approximation of the value of query q_i over times in $\bar{N}_i(t)$. Assuming $r(0, i) = 0$, we define $r(t, i)$ for any $t \in \mathbb{N}$ as,

$$r(t,i) = \begin{cases} q_i(g_t) - C_i(t), & t \in N_i(t), \\ r(t-1,i), & otherwise. \end{cases}$$

Finally, our differentially private approximation $m(t, i)$ of the query q_i at time t becomes

$$m(t,i) = C_i(t) + r(t,i).$$

4.2.3 Optimize. At any time t and iteration l, Algorithm 3 uses the Algorithm $\mathcal{A}_{Dataset}$ as a subroutine to generate the synthetic dataset $h_{t,l}$ using the query indices selected so far at time t, that is $\{\eta_{t,1}, \ldots, \eta_{t,l}\}$, and their corresponding differentially private values $\{m(t, \eta_{t,1}), \ldots, m(t, \eta_{t,l})\}$. $\mathcal{A}_{Dataset}$ can be any algorithm and is not required to satisfy differential privacy.

4.2.4 Select. We are finally ready to talk about query selection. During iteration l of time t, we want to select the query with maximum error over the synthetic dataset $h_{t,l-1}$ as compared to the true dataset f_t. However, accessing $q(f_t)$ results in high sensitivity. Indeed a simple change at some time $\tau \in \mathbb{N}$ can affect the selection at all times $t > \tau$.

To control the sensitivity, we follow the same trick as [3] and approximate f_t as $g_{t-1} + \nabla f_t$ for selection. For any query $q_i \in Q$, $l \in [k]$, and $t \in \mathbb{N}$, we define

$$e_{t,l} := \left(|q_i(\nabla f_t + g_{t-1}) - q_i(h_{t,l-1})| \right)_{i \in [|Q|]}.$$

Finally, we use Exponential Mecnahism as defined in Definition 3.2 for selecting a query index $\eta_{t,l}$ given the vector of query utilities $e_{t,l}$. Note that Algorithm 3 does not find the error for all queries but instead only for queries that have not been chosen so far at iteration l of time t (whose indices are in the set $J_{t,l}$).

Lemma 4.1 (Privacy of Algorithm 3). *If the Algorithms $\mathcal{A}_{Counter}$ and $\mathcal{A}_{Dataset}$ satisfy differential privacy, then Algorithm 3 is ε-differentially private.*

Proof. Since Algorithm 3 is an instance of the "Online Selective Counting" algorithm due to [3], it is therefore differentially private. [GK: TODO: Will add more to this.] □

Algorithm 3 Main Algorithm: DP-STabS

Input: An input data stream f, an ordered set of marginal queries Q, number of marginals to select at any time k, the privacy budget ε, a counter algorithm $\mathcal{A}_{Counter}$, and a subroutine $\mathcal{A}_{Dataset}$ to find a dataset given noisy query measurements.
Output: A synthetic stream g.
Initialize $C_1, C_2, \ldots, C_{|Q|}$ as independent instances of the counter algorithm $\mathcal{A}_{Counter}$, one for each query in the set Q, with privacy budget $\varepsilon/2k$.
Initialize $g(0, x) \leftarrow 1$ for all $x \in \mathcal{X}$.
Initialize $m(0, i) \leftarrow 0$ for all $i \in [|Q|]$; query measurements of selected queries
Initialize $r(0, i) \leftarrow 0$ for all $i \in [|Q|]$; remainder of query value for times when the query is not selected.
for $t = 1, 2, \ldots$ **do**
  Set $I_{t,0} \leftarrow \emptyset$.
  for $l = 1, 2, \ldots, k$ **do**
    Set $J_{t,l} \leftarrow [|Q|] \setminus I_{t,l-1}$; as query indices not selected.
    Set $e_{t,l} \leftarrow \left(|q_i(\nabla f_t + g_{t-1}) - q_i(h_{t,l-1})| \right)_{i \in J_{t,l}}$.
    $\eta_{t,l} \leftarrow ExponentialMechanism\left(e_{t,l}, \varepsilon/2k\right)$.
    Using shorthand j for $\eta_{t,l}$. [GK: Can I write like this?]
    Set $I_{t,l} \leftarrow I_{t,l-1} \cup \{j\}$.
    Update counter C_j by counting on ∇f_t.
    Set $r(t, j) \leftarrow r(t-1, j)$.
    Set $m(t, j) \leftarrow C_j + r(t, j)$.
    Set $h_{t,l} \leftarrow \mathcal{A}_{Dataset}\left(\{(q_i, m(t, i))\}_{i \in I_{t,l}} \right)$.
  end for
  Set $g_t \leftarrow \text{avg}_{l \in [k]}\, h_{t,l}$.
  Set $C_i(t) \leftarrow C_i(t-1)$ for all $i \in [|Q|] \setminus I_{t,k}$.
  Set $r(t, i) \leftarrow q_i(g_t) - C_i(t)$ for all $i \in [|Q|] \setminus I_{t,k}$.
end for

4.3 A new (unbounded) Block counter

We extend the Two-Level counter mechanism (also referred to as Block counter) due to [1] to unbounded streams. We present it formally in Algorithm 4. The idea is similar to how the bounded Binary Mechanism is extended to the unbounded Hybrid Mechanism in [1]. As shown in [1], an optimal block size of the Bounded Block Counter for a stream of size T is $\sqrt{T}$. The key idea is to partition the time dimension of the stream $f : \mathbb{N} \to \mathbb{R}$ into intervals of size $4, 9, 16, \ldots$ (that is perfect squares), and within each of the corresponding intervals, we use a bounded block counter of block size $2, 3, 4, \ldots$ respectively.

Theorem 4.2 (Privacy of unbounded block counter). *The unbounded block counter, as presented in Algorithm 4, satisfies ε-differential privacy.*

Proof. Note that Algorithm 4 is exactly the block counter algorithm, except the size of the block changes over time. However, the change in the block size is independent of the input data stream. Hence, similar to the Block counter, Algorithm 4 is ε-differentially private. [GK: Needs more discussion?]. □

Algorithm 4 Unbounded Block Counter

Input: An input data stream $f : \mathbb{N} \to \mathbb{R}$, the privacy budget ε.
Output: A synthetic stream $g : \mathbb{N} \to \mathbb{R}$.
Initialize partition size $T \leftarrow 4$.
Initialize block size $B \leftarrow 2$.
Last block value $\alpha_{lastBlock} \leftarrow 0$.
True value within block $\alpha_{trueInBlock} \leftarrow 0$.
Synthetic value within block $\alpha_{synthInBlock} \leftarrow 0$.
Time when the last partition changed $t_{atPartition} \leftarrow 0$.
Set $g(0) = 0$.
for $t = 1, 2, \ldots$ **do**
    Set $\delta \leftarrow t - t_{atPartition}$.
    Update $\alpha_{trueInBlock} \leftarrow \alpha_{trueInBlock} + f(t)$.
    if $\delta = kB$ for some $k \in Z$ **then**
        Update $\alpha_{lastBlock} \leftarrow \alpha_{lastBlock} + \alpha_{tueInBlock} + \text{Lap}\left(\frac{2}{\varepsilon}\right)$.
        Update $\alpha_{trueInBlock} \leftarrow 0$ and $\alpha_{synthInBlock} \leftarrow 0$.
        Set $g(t) \leftarrow \alpha_{lastBlock}$.
        if $\delta = T$ **then**
            Update $t_{atPartition} \leftarrow t$.
            Update $B \leftarrow B + 1$ and $T \leftarrow B^2$.
        end if
    else
        Update $\alpha_{synthInBlock} \leftarrow \alpha_{synthInBlock} + f(t) + \text{Lap}\left(\frac{2}{\varepsilon}\right)$.
        Set $g(t) \leftarrow \alpha_{lastBlock} + \alpha_{synthInBlock}$.
    end if
    Release $g(t)$.
end for

REFERENCES

[1] T-H. Hubert Chan, Elaine Shi, and Dawn Xiaodong Song. 2010. Private and Continual Release of Statistics. *ACM Trans. Inf. Syst. Secur.* 14 (2010), 26:1–26:24.

[2] Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N. Rothblum. 2010. Differential privacy under continual observation. In *Symposium on the Theory of Computing*.

[3] Girish Kumar, Thomas Strohmer, and Roman Vershynin. 2024. An Algorithm for Streaming Differentially Private Data. arXiv:2401.14577 [cs.DB]

[4] Terrance Liu, Giuseppe Vietri, and Steven Z. Wu. 2021. Iterative Methods for Private Synthetic Data: Unifying Framework and New Methods. In *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (Eds.), Vol. 34. Curran Associates, Inc., 690–702. https://proceedings.neurips.cc/paper_files/paper/2021/file/0678c572b0d5597d2d4a6b5bd135754c-Paper.pdf