Private Aggregate Computations on Streams

Nina Budaeva[*]
Dept. of Physics,
Mathematics, and Astronomy
nbudaeva@caltech.edu

David Ding
Dept. of Physics,
Mathematics, and Astronomy
ding@caltech.edu

Angela Gong
Dept. of Computing and
Mathematical Sciences
angela@caltech.edu

Theresa Lee
Dept. of Computing and
Mathematical Sciences
theresa@caltech.edu

Mike Qian
Dept. of Computing and
Mathematical Sciences
mqian@caltech.edu

Kalpana Suraesh
Dept. of Computing and
Mathematical Sciences
ksuraesh@caltech.edu

ABSTRACT

Differential privacy is a relatively new area of computer science research that has only recently been explored. Currently much of the research on privacy mechanisms focuses on static databases (i.e. relational databases). Some papers have discussed stream databases, but relatively few discuss the possibility that privacy-preserving mechanisms other than the counting mechanism could be applied on databases. While a privacy-preserving counting mechanism is a very useful aggregate in analysis of sensitive data, the proposed algorithms severely limit the scope of analysis. We aim to widen this scope by proposing less restrictive mechanisms, as well as aggregates other than count, such as sum, average, and variance.

Keywords

Differential privacy, streaming algorithms, databases, continual mechanisms

1. INTRODUCTION

1.1 Motivation

Dwork [1] and Chan [3] both describe ε-differentially private mechanisms that add noise to each item in a binary stream. However, we may want to perform more sophisticated analyses on a stream. For example, it is often useful to obtain the sum, average, or variance of items observed over a stream. In addition, if the entries of a stream vary greatly in frequency or in magnitude, it may not be ideal to model a stream as a binary stream. Hence, it would be pragmatic to design differentially-private mechanisms that work for new aggregate operators as well as non-binary streams.

[*]This research was done entirely at the California Institute of Technology.

1.2 Contributions

The primary contributions of this paper include unbounded mechanisms to compute privacy-preserving sums, bounded and unbounded mechanisms to compute privacy-preserving averages, and a bounded mechanism to compute variance. Furthermore, these mechanisms all preserve ε-pan privacy.

Our mechanisms are significant because we have not found published algorithms that provably preserve ε-differential privacy on unbounded streams for aggregate operators such as sum, average, or variance. These are extremely important when dealing with potentially sensitive user data, for example. Additionally, these aggregate operators are commonly used on many databases, so these privacy-preserving mechanisms are very applicable in industry and the real world.

1.3 Related Work

1.3.1 *Differential Privacy Under Continual Observation*

[1] discusses pan privacy in the context of streaming and compares it to ε-differential privacy. The paper describes a pan-private counter and determines its privacy guarantees. In particular, the paper reviews the event-level counter algorithm and shows that it satisfies pan privacy as defined before and that a general event-level counter has a logarithmic lower bound in error. There is further discussion about transforming a generic single-output streaming algorithm into one that continually produces output, which was useful for our research.

1.3.2 *Private and Continual Release of Statistics*

[3] discusses pan privacy and consistency, and introduces the `p-sum` framework. The `p-sum`s are intermediate results from which an observer can estimate the count at every time step (further discussion can be found in Section 2.2.2). The paper also provides various time-bounded mechanisms for outputting differentially private counts, which generally become more complex as the error bounds get tighter. The most useful of these is the Binary Mechanism, which uses a noisy binary frequency tree to compute counts. Major

contributions of this paper include a method to convert any bounded ε-differentially private mechanism into an unbounded ε-differentially private mechanism, as well as a method to convert any ε-differentially private mechanism into an ε-pan private mechanism.

1.3.3 Private Decayed Predicate Sums on Streams

[2] discusses ε-differentially private bounded mechanisms for sums on streams. Specifically, they include mechanisms and algorithms for the normal sum (called windowed sum, here), polynomial sum, and exponentially decaying sum, and show tight lower bounds in the error. The decayed histogram problem is also addressed, as well as a nice theorem that helped prove the ε-differential privacy of our mechanisms.

2. TOOLS AND DEFINITIONS

2.1 Differential Privacy

2.1.1 Differential Privacy in the Traditional Setting

A mechanism's privacy is formally defined in terms of differential privacy, as first discussed in [1]. A mechanism is considered differentially private if the output when run on two nearly identical input streams is indistinguishable. An adversary who is trying to recover the data will be unable to determine whether or not an event took place simply by observing the output of the mechanism.

Definition 1. Two input streams σ and σ' are adjacent if they differ by at most one tuple t. A mechanism $\mathcal{M}$ preserves ε-differential privacy if for any adjacent streams σ and σ', and for any subset of the possible outputs of the mechanism $S \subseteq Range(\mathcal{M})$,

$$Pr[\mathcal{M}(\sigma) \in S] \leq \exp(\varepsilon) \cdot Pr[\mathcal{M}(\sigma') \in S]. \qquad (1)$$

The nature of the bound guarantees that an output with zero probability in a stream σ will also occur with zero probability in a neighboring stream σ'. ε is a public value, and is usually small (about 0.01 to 1.4).

2.1.2 Pan Privacy

We are primarily interested in pan privacy, which is an extension of differential privacy. A pan-private mechanism is one that can preserve differential privacy even if an adversary is able to access the intermediate states of the mechanism. Pan privacy guarantees are such that even if the government or some other organization accesses the database in between or during calculations, the information they gather is noisy and does not reveal any private information.

Definition 2. Let $\mathcal{M}$ be an algorithm, and I be the set of internal states of the algorithm (such as a counter, etc.), and $Range(\mathcal{M})$ the set of possible outputs by the mechanism. Then the mechanism is pan-private (against a single intrusion) if for all sets $I' \subseteq I$ and $S \subseteq Range(\mathcal{M})$ and all sets of neighboring streams σ and σ',

$$Pr[\mathcal{M}(\sigma) \in (I', S)] \leq \exp(\varepsilon) \cdot Pr[\mathcal{M}(\sigma') \in (I', S)]. \qquad (2)$$

2.2 Tools

2.2.1 Laplace Distribution

When ensuring differential or pan privacy, the Laplace distribution is generally used to draw random noise. Following convention, we use $\text{Lap}(b)$ to denote a Laplace distribution with mean 0 and variance $2b^2$.

Property 1. Let $x, y \in \mathbb{R}$ where $|a - b| \leq \Delta$. If $\gamma \sim \text{Lap}(\Delta/\varepsilon)$ is a random variable drawn from a Laplace distribution, then for any subset $S \subseteq \mathbb{R}$, we have

$$Pr[x + \gamma \in S] \leq \exp(\varepsilon) \cdot Pr[y + \gamma \in S]. \qquad (3)$$

2.2.2 `p-sum` Framework

As first defined in [3], a `p-sum` is a partial sum over consecutive elements of a sequence. In particular, let $1 \leq i \leq j$. We use $\Sigma\,[i,j] := \sum_{k=i}^{j} \sigma\,(k)$ to denote a partial sum involving items i through j (inclusive) of stream σ.

When we add noise to a `p-sum`, we obtain a noisy `p-sum` denoted by $\widehat{\Sigma}$. As new items arrive in the stream, the mechanisms we consider release one or more noisy `p-sums`. An observer may compute an estimate for a function at each time step by operating over an appropriate selection of `p-sums` from the sequence released by the database. For instance, if the function is the count, and we observe a `p-sum` $\widehat{\Sigma}\,[1,k]$ and a second `p-sum` $\widehat{\Sigma}\,[k+1,t]$, an estimate for the count at time t may be computed by summing $\widehat{\Sigma}\,[1,k] + \widehat{\Sigma}\,[k+1,t]$.

As we develop our algorithms, we will often describe them using this common framework by showing how the output of each mechanism can be viewed as an operation defined over a suitable collection of noisy `p-sums`.

2.2.3 Corollaries to Measure Concentration

We cite the corollary on Measure Concentration from [3], which can be used to prove utility bounds on algorithms:

Corollary 2.9 (Chan). Suppose γ_i's are independent random variables, where each γ_i has identical Laplace distribution $\text{Lap}(b_i)$. Suppose $Y := \sum_i \gamma_i$, and $b_M := \max_i b_i$. Let $\nu \geq \sqrt{\sum_i b_i^2}$ and $0 < \gamma < \frac{2\sqrt{2}\nu^2}{b_M}$. Suppose $0 < \delta < 1$ and $\nu > \max\left\{\sqrt{\sum_i b_i^2}, b_M\sqrt{\ln\frac{2}{\delta}}\right\}$. Then $Pr\left(|Y| > \nu\sqrt{8\ln\frac{2}{\delta}}\right) \leq \delta$.

Here we will use a simplification of this Corollary since all the noise added for a particular algorithm is identically distributed.

Corollary 1. Suppose γ_i's are n independent random variables, where each $\gamma_i \sim \text{Lap}(b)$. Suppose $Y := \sum_i \gamma_i$, and $b_M := \max_i b_i = b$. To simplify the Corollary, let $\nu = b\sqrt{n\ln\frac{2}{\delta}}$ and $0 < \lambda < \frac{2\sqrt{2}\nu^2}{b_M}$. Suppose also that $0 < \delta < 1$. Then $Pr\left(|Y| > \nu\sqrt{8\ln\frac{2}{\delta}}\right) \leq \delta$. We can also use the slightly weaker result: with probability at least $1 - \delta$, the quantity $|Y|$ is at most $O\left(b\sqrt{n}\log\frac{1}{\delta}\right)$.

2.2.4 Privacy and Error Bound Observation

We also cite the following helpful observation from [3] as it can be used to (informally) reason about privacy and error bounds for mechanisms in terms of the number of `p-sums` involved in their outputs and the number of `p-sums` that differ between adjacent streams.

Observation 1. Suppose a mechanism $\mathcal{M}$ adds $\text{Lap}(1/\varepsilon)$ noise to every `p-sum` before it is released. In $\mathcal{M}$, each item in the stream appears in at most x `p-sums`, and each estimated count is the sum of at most y `p-sums`. Then $\mathcal{M}$ preserves $x\cdot\varepsilon$ differential privacy, and by Corollary 1, the error is $O(\sqrt{y}/\varepsilon)$ with high probability [3].

2.3 Counting Mechanisms

To help us with our understanding of counting mechanisms, we begin with some definitions adapted from [3].

Definition 3. (Stream) A stream $\sigma \in \{0,1\}^{\mathbb{N}}$ is a bit-string of countable length. $\mathbb{N}$ is the set of natural numbers and $\sigma(t) \in \{0,1\}$ is a bit at time $t \in \mathbb{N}$ so that $\sigma_T \in \{0,1\}^T$ is a length T prefix of the stream σ. A bit of the stream is often called an item.

Definition 4. (Continual Counting Query) The count c_σ for a stream $\sigma \in \{0,1\}^{\mathbb{N}}$ is a mapping $c_\sigma : \mathbb{N} \to \mathbb{Z}$ such that for each $t \in \mathbb{N}$, $c_\sigma(t) := \sum_{i=1}^{t} \sigma(i)$.

Definition 5. (Counting Mechanism) A counting mechanism $\mathcal{M}$ takes a stream $\sigma \in \{0,1\}^{\mathbb{N}}$ and produces a (possibly randomized) mapping $\mathcal{M}(\sigma) : \mathbb{N} \to \mathbb{R}$. For all $t \in \mathbb{N}$, $\mathcal{M}(\sigma)(t)$ is independent of all $\sigma(i)$ for $i > t$. $\mathcal{M}(\sigma)$ can also be viewed as a point in $\mathbb{R}^{\mathbb{N}}$.

Definition 6. (Time-bounded Mechanism) A mechanism $\mathcal{M}$ is unbounded if it accepts streams of indefinite length: $\forall\sigma$, $\mathcal{M}(\sigma) \in \mathbb{R}^{\mathbb{N}}$. Given $T \in \mathbb{N}$, $\mathcal{M}$ is T-bounded if it only accepts streams of length at most T and returns $\mathcal{M}(\sigma) \in \mathbb{R}^{\mathbb{T}}$. Such a mechanism needs to know the value T in advance and only looks at the length T prefix of any stream.

Definition 7. (Utility) A mechanism $\mathcal{M}$ is (λ, δ)-useful at time t if, for any stream σ, with probability at least $1-\delta$ we have $|c_\sigma(t) - \mathcal{M}(\sigma)(t)| \le \lambda$. Note that λ may itself be a function of δ and t. This definition of utility covers the usefulness of the mechanism for a single timestep.

2.3.1 General Event-Level Private Counting

Formally, time is viewed as a series of time periods named by natural numbers. Each time event has two possibilities: an event occurs or an event does not occur. This can be modeled by a binary stream described above in which a 1 corresponds to an event and a 0 corresponds to no event.

The *web counter problem* is to continually produce an accurate estimate of the number of events that have occurred thus far while obscuring for each time period t whether or not an event actually occurred.

Definition 8. A randomized streaming algorithm yields a (T, α, β) counter if in every execution, simultaneously for all $1 \le t \le T$, after processing a prefix of length t, the probability that the current output contains an estimate of the number of 1's in the prefix that differs from the true weight of the prefix by at most an additive amount α is at least $1-\beta$ over the coin flips for the algorithm.

Using an event-level counter for which α grows as roughly $\log(1/\beta)\log^{2.5} T$, the following algorithm is shown in [1] to be pan-private against a single intrusion:

```
Algorithm 1: Event-Level Counter
Initialization: Initialize $\xi = \frac{1+\log T}{\varepsilon}$ and sample
$count \sim \text{Lap}(\xi)$.
Segments: For $i \in \{1,\dots,\log T\}$, associate with each
string $s \in \{0,1\}^i$ the time segment $S$ of $2^{\log T - i}$ time
periods $\{s \circ 0^{\log T - i}, \dots, s \circ 1^{\log T - i}\}$. The first and last
elements are the times at which the segment begins and
ends, respectively.
Processing: In time period $t \in \{0,1,\dots,T-1\}$, let
$x_t \in \{0,1\}$ be the $t$-th input bit:
  1. $count \leftarrow count + x_t$.
  2. For every segment $S$ which begins at time $t$, sample
     noise $\eta_S \cong \text{Lap}(\xi)$.
  3. Let $S_1,\dots,S_{\log T}$ be the $\log T$ segments that contain
     $t$. Output $count + \sum_{i=1}^{\log T} \eta_{S_i}$.
  4. For every segment $S$ that ends in time $t$, erase $\eta_S$.
```

Theorem 1. The above algorithm run with parameters T, ε, $\beta > 0$ yields a $\big(T, 4\log(1/\beta)\cdot\log^{2.5} T/\varepsilon, \beta\big)$ counter with ε-differential privacy against a single intrusion.

Proof. Let S be a segment and η_S the noise value. This value is deleted during the atomic step in the last time period contained in S. Add $1+\log T$ independently chosen Laplace noise variables to the true answer. The variables have variance $2(\log T/\varepsilon)^2$ so the probability that the sum of noises is within $2\log(1/\beta) + \log T$ standard deviations is at least $1-\beta/T$. Therefore, in all rounds simultaneously, the probability that the sum of noises is less than $4\log(1/\beta)\cdot\log^{2.5} T/\varepsilon$ in magnitude is at least $1-\beta$.

The adversary's view between steps consists of the noisy count, segment noise values η_S in memory at the time of the intrusion, and the full sequence of all of the algorithm's outputs.

Let two adjacent databases x and x' differ in time t so that $x_t = 1$ and $x;_t = 0$. An intrusion occurs right after $t^* \ge t$. The true count at t^* is greater for x than for x'. Fixing an arbitrary execution E_x when the input stream is x, and thus fixing the randomness, fixes the generated noise values. The same is done for $E_{x'}$. Since the variable *count* is initialized with Laplace noise, increasing the noise by 1 in $E_{x'}$ does not change the *count* just after t^*, so it is identical in E_x and $E_{x'}$. Further, the noise in memory after t^* is independent of the input so it will be unchanged in $E_{x'}$.

In order to make the sum of all noise values in all times up to $t-1$ unchanged, but the sum from time t and on larger by 1 for x', make the sequence of outputs in $E_{x'}$ identical to those of E_x by changing a collection of $\log T$ segment noise values η_S that are in memory when the adversary intrudes.

Since the initialization noise for *count* was increased, the sum of segment noises for periods less than t should be decreased by 1. This can be done by finding a collection of disjoint segments with union $\{0, \ldots, t-1\}$. This collection is always no greater than size $\log T$ and can be constructed iteratively. Such segments will all end at time $t-1 < t \leq t^*$ so their noises are no longer in memory at the time of intrusion. Thus the complete view of the adversary for either input is identical and the probabilities of the noise values used for x and x' differ by at most a multiplicative factor $\exp(\varepsilon)$. This implies pan-privacy. □

A similar procedure can be done for the case $t^* < t$ by adding 1 to the sum of segment noises in every time period greater than t.

2.3.2 Simple Counting Mechanism I

According to [3], the Simple Counting Mechanism I directly extends the Laplace mechanism proposed by [1] by applying this mechanism to streams. Specifically, the mechanism answers a new query at each time step by returning a value that is close to the true count but is randomized with fresh independent noise drawn from a Laplace distribution.

Given a stream $\sigma \in \{0,1\}^{\mathbb{N}}$, a differential privacy parameter $\varepsilon > 0$, and an upper bound on time T, the mechanism does the following at each time step $t \in \mathbb{N}$:

1. Sample fresh Laplace noise $\gamma_t \sim \text{Lap}\left(\frac{1}{\varepsilon}\right)$.
2. Release $\mathcal{M}(\sigma)(t) = c(t) + \gamma_t$, where $c(t)$ is the true count at t.

We can use the `p-sum` framework to argue that the above mechanism is $O(T\varepsilon)$-differentially private. In particular, observe that flipping a single bit in the incoming stream affects $O(T)$ `p-sum`s released by Mechanism I. Applying Observation 1 gives the result directly.

2.3.3 Simple Counting Mechanism II

The Simple Counting Mechanism II is a variation on the Simple Counting Mechanism I. Instead of maintaining the true count, Mechanism II produces a "sanitized" stream by adding fresh independent Laplace noise to every item in the incoming stream. Like the previous mechanism, it can be implemented with $O(1)$ space. Although this mechanism is unbounded, we still require an upper bound on time T to analyze its utility.

Given a stream $\sigma \in \{0,1\}^{\mathbb{N}}$, a differential privacy parameter $\varepsilon > 0$, and an upper bound on time T, the mechanism does the following at each time step $t \in \mathbb{N}$:

1. Sample fresh Laplace noise $\gamma_t \sim \text{Lap}\left(\frac{1}{\varepsilon}\right)$.
2. Define $\alpha_t := \sigma(t) + \gamma_t$, where $\sigma(t)$ is the bit at time t.
3. Release $\mathcal{M}(\sigma)(t) = \sum_{i \leq t} \alpha_i$.

In Mechanism II, flipping a single bit of the incoming stream affects only 1 `p-sum`, so by Observation 1, it preserves ε-differential privacy.

2.4 Intrusions

2.4.1 Definition

An intrusion occurs when an adversary is able to take snapshots of a mechanism's internal states. We assume that an intrusion can only occur when the mechanism has finished its update at a certain time step (i.e. intrusions can only occur between atomic steps) [3].

2.4.2 Pan Privacy against Single Unannounced Intrusion

Given a mechanism $\mathcal{M}$, where $Range(\mathcal{M})$ is the range of possible outputs of the mechanism on different streams, a stream σ, the time t of the unannounced intrusion, and the knowledge i_t gained after an intrusion at time t, the output of the mechanism $\mathcal{M}(\sigma)$ on a stream σ at time t is some element $(i_t, s) \in Range(\mathcal{M}) \times \mathbb{R}^{\mathbb{N}}$.

The mechanism $\mathcal{M}$ is ε-pan private against a single intrusion if for any adjacent streams σ and σ', and any subset $S \subseteq Range(\mathcal{M}) \times \mathbb{R}^{\mathbb{N}}$,

$$Pr[\mathcal{M}(\sigma) \in S] \leq exp(\varepsilon) \cdot Pr[\mathcal{M}(\sigma') \in S]. \tag{4}$$

2.4.3 Pan Privacy against Multiple Announced Intrusions

Given a mechanism $\mathcal{M}$, where $Range(\mathcal{M})$ is the range of possible outputs of the mechanism $\mathcal{M}$, a stream σ, a number of intrusions k, the knowledge i gained during the k intrusions, and a subset $K \subset \mathbb{N}$ of size k representing the time steps in which intrusions occurred, the output of the mechanism $\mathcal{M}(\sigma, K)$ is an element $(i, s) \in Range(\mathcal{M})^{|K|} \times \mathbb{R}^{\mathbb{N}}$.

The mechanism $\mathcal{M}$ is ε-pan private against multiple intrusions if for any adjacent streams σ and σ', and any subset $S \subseteq Range(\mathcal{M})^{|K|} \times \mathbb{R}^{\mathbb{N}}$,

$$Pr[\mathcal{M}(\sigma, K) \in S] \leq exp(\varepsilon) \cdot Pr[\mathcal{M}(\sigma', K) \in S]. \tag{5}$$

2.4.4 Transformations

Any ε-differentially private mechanism in the `p-sum` framework can be transformed into an ε-pan private mechanism to protect against intrusions, with only a constant-factor loss in the mechanism's utility.

In general, a mechanism in the `p-sum` framework initializes some counter $\beta = 0$ at the start of the computation, and at each time t, adds the true value of the stream $\sigma(t)$. At specific intervals, noise $\text{Lap}(\alpha/\varepsilon)$ is added. While this is ε-differentially private, it does not protect against intrusions, because if an attacker were to intrude at a time when no noise has been added to β yet, the true value of the computation would be exposed.

Theorem 2. Any ε-differentially private mechanism can be converted into an ε-pan private mechanism against a single intrusion with the same privacy guarantee and same asymptotic error guarantee.

Corollary 2. If we add $n \cdot \mathrm{Lap}(\alpha/\varepsilon)$ more noise to the p-sums, then there will be a constant factor $\sqrt{n}$ in the error bound.

Proof. Consider our β from before. If it were initialized to $\beta = \mathrm{Lap}(\alpha/\varepsilon)$, then it becomes ε-pan private because the attacker would no longer know the true value of the computation, and only one p-sum is affected. By Corollary 2, there will only be a constant factor of $\sqrt{2}$ increase in the error. □

3. TIME-BOUNDED COUNTING MECHANISMS

3.1 Count for Non-Binary Streams

[3] defines a stream $\sigma \in \{0,1\}^{\mathbb{N}}$ to be a sequence of bits, where a bit at time t denotes whether an event of interest occurred at time t. This is not necessarily representative of a real stream, where events such as purchasing items may happen in "bursts," with the distribution of events changing over time. We therefore propose an alternate definition of a stream, in which we consider time intervals t of constant length, and $\sigma(t)$ is a set of 1's. Each 1 corresponds to an event during t (in particular, a null set means that no events occurred during t).

For convenience, we may consider the equivalent definition of a stream to be $\sigma \in \{0,1,\ldots,N\}$, where the $\sigma(t)$ is the number of events that occurred during t, and N is an upper bound that we place on the number of events that can occur per time interval. This makes notation consistent with the notation in [3].

Theorem 3. For $T \in \mathbb{N}$, the Binary Mechanism for Non-Binary Streams preserves T-bounded ε-differential privacy.

Proof. The proof is analogous to the proof in [3] that the Binary Counting Mechanism on binary streams preserves T-bounded ε-differential privacy. The result follows immediately from Property 1 since each noisy p-sum maintains $\frac{\varepsilon}{N \log T}$-differential privacy. □

3.1.1 Consistency of Bursty Streams

[3] defined a mechanism $\mathcal{M}$ to be consistent if for any stream σ, for all $t \in \{1,2,\ldots,T\}$, $\mathcal{M}(\sigma)(t) - \mathcal{M}(\sigma)(t-1) \in \{0,1\}$, with the convention $\mathcal{M}(0) = 0$. We can easily create an analogous definition of a consistent mechanism for our non-binary streams by simply requiring that $\mathcal{M}(\sigma)(t) - \mathcal{M}(\sigma)(t-1) \in \{0,1,\ldots,N\}$ instead of being restricted to 0 and 1. Intuitively, this means that each value in the transformed stream is an integer, and that the count does not increase by more than the specified maximum bound N, so that the stream can correspond to a real stream.

[3] also provide a method to make any mechanism $\mathcal{M}$ into a consistent mechanism $\mathcal{M}'$. We provide an analogous method to make any mechanism consistent, given our new definition:

1. If $\mathcal{M}$ does not report integer values, round each value to an integer, introducing an error of at most $\frac{1}{2}$.
2. Set $\widehat{\mathcal{M}}(\sigma)(0) := 0$.
3. For each $t \in \mathbb{N}$, if $\mathcal{M}(\sigma)(t) > \widehat{\mathcal{M}}(\sigma)(t-1)$, $\widehat{\mathcal{M}}(\sigma)(t) :=$
$$\widehat{\mathcal{M}}(\sigma)(t-1) + \min(N, \mathcal{M}(\sigma)(t) - \widehat{\mathcal{M}}(\sigma)(t-1))$$
Otherwise, $\widehat{\mathcal{M}}(\sigma)(t) := \widehat{\mathcal{M}}(\sigma)(t-1)$.

4. UNBOUNDED AND BOUNDED AGGREGATES

4.1 Unbounded Sum

This algorithm is an extension of the Hybrid Mechanism, which was used in [3] to calculate unbounded count. The Hybrid Mechanism converts any generic bounded counting mechanism $\mathcal{M}$ into an unbounded mechanism by combining it with the output of the Logarithmic Counting Mechanism ($\mathcal{L}$).

[2] proposed a bounded counting mechanism for sums that is ε-differentially private, which we will use as the bounded mechanism $\mathcal{M}$ mentioned above. Following their lead, we can use a definition of local sensitivity for a function f as the smallest real number that satisfies $\forall x_1,\ldots,x_T, \forall j \in [T], \forall x'_j \in [a,b]$:

$$\left\| f(x_1,\ldots,x_j,\ldots,x_T) - f(x_1,\ldots,x'_j,\ldots,x_T) \right\|_1 \leq S_f \quad (6)$$

Given this local sensitivity S_f, Theorem 1 of [2] proves that an algorithm which outputs a noisy result $\hat{f}(x_1,\ldots,x_T) = f(x_1,\ldots,x_T) + \mathrm{Lap}(S_f/\varepsilon)$ satisfies ε-differential privacy.

Given this theorem, the tracking of sums is identical to tracking of count. So, the windowed sum mechanism that we use is nearly identical to the Binary Mechanism presented by [3], i.e., it adds noise $\sim \mathrm{Lap}(\frac{\log W+1}{\varepsilon})$, for windows of size W.

The complete algorithm for unbounded sum is therefore to run the Hybrid Mechanism $\mathcal{H}$ detailed on page 15 of [3], and instantiate it with the bounded mechanism WindowSum detailed in [2], which we can call $\mathcal{M}$. The privacy parameter for both $\mathcal{L}$ and $\mathcal{M}$ will be $\varepsilon/2$. We will call this specific mechanism the Unbounded Sum Mechanism.

Theorem 4. The Unbounded Sum Mechanism, which can be defined as the Hybrid Mechanism instantiated with the WindowSum mechanism as $\mathcal{M}$, is ε-differentially private and $\left(O\left(\frac{1}{\varepsilon}\right)\sqrt{\log t}\log\frac{1}{\delta} + \gamma, \delta\right)$-useful at time t. This occurs with a window of size W and with $\gamma = O(\frac{1}{\varepsilon}\log^{1.5} W \log^{0.5}\frac{1}{\delta})$ when $\log W \geq \log\frac{1}{\delta}$ and $\gamma = O(\frac{1}{\varepsilon}\log W \log\frac{1}{\delta})$ when $\log W < \log\frac{1}{\delta}$.

Proof. We use the following theorem from [3]:

Theorem 4.7 (Chan). Assume that given any $\varepsilon > 0$ and $0 < \delta < 1$, Logarithmic Mechanism $\mathcal{L}$ is ε-differentially private and is $(g(\varepsilon,T,\tau,\delta),\delta)$-useful at time $\tau \in [T]$, where g is monotonically increasing with T and τ. Then, the Hybrid Mechanism that uses the Logarithmic Mechanism $\mathcal{L}$ instantiated with $\varepsilon/2$ and a bounded mechanism $\mathcal{M}$ instantiated with $\varepsilon/2$ is unbounded, preserves ε-differential privacy, and is $\big(f(\frac{\varepsilon}{2}, t, \frac{\delta}{2}) + g(\frac{\varepsilon}{2}, t, t, \frac{\delta}{2}), \delta\big)$-useful at time t.

From [3], we know that the Logarithmic Mechanism is ε-differentially private and $\big(O(\frac{1}{\varepsilon})\sqrt{\log t}\log\frac{1}{\delta}, \delta\big)$ useful when instantiated with the privacy parameter ε. We also know that the `WindowSum` mechanism is ε-differentially private and (γ, δ) useful when instantiated with the privacy parameter ε, and window size W where $\gamma = O(\frac{1}{\varepsilon}\log^{1.5} W \log^{0.5}\frac{1}{\delta})$ when $\log W \geq \log\frac{1}{\delta}$ and $\gamma = O(\frac{1}{\varepsilon}\log W \log\frac{1}{\delta})$ when $\log W < \log\frac{1}{\delta}$, from [2]. Thus, using the theorem above from [3], the Unbounded Sum Mechanism is ε-differentially private and $(O(\frac{1}{\varepsilon})\sqrt{\log t}\log\frac{1}{\delta} + \gamma, \delta)$ useful at time t, for the specified γ. □

4.2 Bounded Average

We can extend the idea of a summing mechanism along with a counting mechanism to form a mechanism that can compute an average (since an average is just a sum over a count). This mechanism is derived from the Binary Mechanism in [3]. At a time t, the mechanism will group the data points that arrive to create differently-sized `p-sums`. The key for each group is the timestamp of the data point t, which is converted into its binary representation. Essentially, the Binary Average Mechanism will run either the Binary (Counting) Mechanism for Non-Binary Streams or a bounded sum mechanism, along with a standard binary counting mechanism simultaneously.

The major differences between the Binary Average Mechanism and the Binary Mechanism illustrated in [3] are:

- Instead of a binary stream of `1`'s and `0`'s, the stream σ contains real values $x_i \in \mathbb{R}$.
- The Binary Average Mechanism must keep a running `p-sum` for the current sum as well as the current count.

Before we prove the algorithm's differential privacy, we first cite a theorem of [2].

Theorem (DMNS06). Let $\mathcal{A}_1$ be an algorithm that satisfies ε_1-differential privacy and $\mathcal{A}_2$ one that satisfies ε_2-differential privacy. Then an algorithm $\mathcal{A}$ that on input x outputs $\mathcal{A}(\mathcal{A}_1(x), \mathcal{A}_2(x))$ satisfies $(\varepsilon_1 + \varepsilon_2)$-differential privacy.

Then we can show that our mechanism is ε-differentially private.

Theorem 5. For an upper bound on time T, the Binary Average Mechanism $\mathcal{A}$ preserves T-bounded ε-differential privacy.

Proof. From [3], we found that the Binary (Counting) Mechanism $\mathcal{C}$ satisfies ε-differential privacy if we set ε' to $\varepsilon/\log T$. For the Binary Average Mechanism, we set

$$\varepsilon' = \varepsilon/(2\log T) \tag{7}$$

which makes $\mathcal{C}$ satisfy $\varepsilon/2$-differential privacy. Similarly, the Binary Sum Mechanism $\mathcal{S}$ satisfies ε-differential privacy as shown in Section 3.1 when $\varepsilon' = \varepsilon/\log T$. Again, for the Binary Average Mechanism, we set

$$\varepsilon' = \varepsilon/(2\log T)$$

so we have $\varepsilon/2$-differential privacy. By Theorem (DMNS06), if we have an algorithm $\mathcal{A}$ that divides these two (i.e., is a function of $\mathcal{C}$ and $\mathcal{S}$), then the differential privacy guarantee provided by $\mathcal{A}$ is $\frac{\varepsilon}{2} + \frac{\varepsilon}{2} = \varepsilon$. We conclude that the Binary Average Mechanism is ε-differentially private. □

Conjecture 1. (Utility) For each $t \in [T]$, the T-bounded Binary Average Mechanism is $(O(\frac{1}{\varepsilon})(2\log T)\sqrt{\log t}\log\frac{1}{\delta}, \delta)$-useful at time $t \in [T]$.

Proof. We imagine that the proof of this is analogous to that found in [3] to prove the utility of the Binary Mechanism, but we were unable to fully adapt the details of that proof for our purposes. □

4.3 Bounded Variance

We can use the Binary Average Mechanism to form a mechanism that can compute variance using the rolling variance formula. The rolling variance formula uses the existing data to compute an updated variance of the data when new data points are added to the existing data points [6].

Theorem 6. For an upper bound on time T, the Binary Variance Mechanism $\mathcal{V}$ preserves T-bounded ε-differential privacy.

Proof. As proven before, we found that the Binary Average Mechanism $\mathcal{A}$ satisfies ε-differential privacy. In the Binary Variance Mechanism, we are using $\mathcal{A}$ to get noisy `p-sums` for the estimated average and the estimated power sum average. Thus, the Binary Average (and therefore the power sum average) Mechanism $\mathcal{A}$ satisfies ε-differential privacy, if $\varepsilon' = \varepsilon/\log T$. When we use the rolling variance formula, we are taking a sum of at most $O(\log T)$ noisy `p-sums` and then dividing this by another noisy p-sum (also a sum of at most $O(\log T)$ noisy `p-sums`). By Theorem (DMNS06), if we have an algorithm $\mathcal{V}$ that divides these two (i.e., is a function of $\mathcal{C}$ and $\mathcal{A}$), then $\mathcal{V}$ satisfies $(\frac{\varepsilon}{2} + \frac{\varepsilon}{2} = \varepsilon)$-differential privacy. We conclude that the Binary Variance Mechanism is ε-differentially private. □

Conjecture 2. (Utility) For each $t \in [T]$, the T-bounded Binary Variance Mechanism is $(O(\frac{1}{\varepsilon})(2\log T)\sqrt{\log t}\log\frac{1}{\delta}, \delta)$-useful at time $t \in [T]$.

Proof. We imagine that the proof of this is analogous to that found in [3] to prove the utility of the Binary Mechanism, but we were unable to fully adapt the details of that proof for our purposes. □

5. APPLICATIONS

5.1 Use in Distributed Computation

One application of pan-private streaming algorithms is in distributed computing, where computational tasks are executed in parallel across multiple machines, possibly in different physical locations. For example, Netflix may want to calculate recommendations for its users based on the previous movies they have watched. This obviously involves computations on sensitive data.

However, if Netflix does not have enough resources, then they would like to outsource these computations to large computing clusters, or even the everyday person, without leaking any private information. With a pan-private mechanism, this is very easy to do, since pan-privacy guarantees the privacy of data even when one or more intrusion occurs. This also means that pan-privacy guarantees the privacy of data when an intentional "intrusion" occurs (i.e. when Netflix willingly gives the private data away). Therefore, in order to run any of the ε-pan private mechanisms mentioned in this paper, the following must be done:

1. Add noise $\sim \text{Lap}(1/\varepsilon)$ to each data point to make it private.
2. Send the data to all machines involved in the distributed computation.
3. Instruct each machine to run any ε-pan private mechanism.

5.2 Stream Databases

The mechanisms mentioned earlier can easily be translated and implemented for practical purposes. Certain companies can take advantage of the privacy-protecting measures of pan-private mechanisms to learn certain statistics regarding the demographics of their customers without compromising the privacy of users. In this situation, these companies can apply these privacy-preserving mechanisms to the aggregate operators before computing them on their database.

5.2.1 Electricity Usage

One clear example of the advantages of privacy-preserving aggregates is its usage in electricity demand data. An electricity company may want to install smart meters in homes in order to monitor live electricity demand in order to better serve the neighborhood. However, a household's live electricity usage is very sensitive and prone to intrusion by a malicious attacker via side-channel attacks. The intruder could, for example, find the troughs of the electricity usage to figure out when the family is not home in order to rob them. In order to prevent such intrusions, the electricity company could instead install a privacy-preserving meter that would get a noisy average of the household's electricity usage (possibly using the Binary Average Mechanism). If the electricity company were to sum up the average usage of all households, the noise should cancel out (since the mechanisms all add Laplace noise centered near zero). The electricity company would then be able to simultaneously have both an accurate sum usage of the entire neighborhood, while still maintaining private individual household averages.

5.2.2 Live Poll Results

As people vote in a poll (particularly one in which there is more than one choice or the choices come in a range), their vote can affect the overall result. If their vote is very far from the central, "accepted" choice, then it will show up significantly when finding the average or variance of the voting results. It would be useful to make such averages or variances differentially private as to protect the privacy of voters. The Binary Average Mechanism and Binary Variance Mechanism may then be useful in determining live poll results without risking a loss of accuracy.

5.2.3 Medical Records

Health information is a prime example of highly sensitive data that individuals prefer to keep private, and laws such as HIPAA strictly regulate the dissemination of such information. It should be possible to maintain this data in such a way that privacy-preserving aggregates can be run to survey the health of a population, while still allowing doctors and patients access to the information on an as-needed basis in order to provide proper, informed care. As an example, regional or national health agencies often benefit from knowledge about the spatiotemporal distribution, incidence, and prevalence of certain conditions, and many communicable diseases even have specific reporting requirements when diagnosed in the population. If this aggregate data is stored in a database where it is not directly being used to treat patients, then it may be wise to make the information pan-private against possible intrusions. At the same time, patients generally do not want their health insurance companies to be privy to their entire medical histories as this knowledge may affect the premiums they pay, so it would be useful to have some form of privacy applied to their data in transmission that still allows the health care provider to bill for services rendered.

6. CONCLUSION

6.1 Future Work

Although we have now significantly increased the variety of privacy-preserving streaming mechanisms available, there is certainly more work that could be done. Should we continue our work, some things to consider include:

- Finding better bounds on the error and utility of the mechanisms that we have developed. This may involve further modifying our mechanisms or creating entirely new ones.
- Create privacy-preserving mechanisms for other operators, such as maximum and minimum. There may be other aggregates that are useful that have not been considered yet.
- Practical implementation of the mechanisms. Currently we have only theoretically proven the value and utility of these mechanisms. It may be useful to actually code these to see how well useful they are in a practical setting.

6.2 Summary

In this paper, we have considered how certain aggregates such as sum, average, and variance could be outputted in a privacy-preserving manner. We proposed mechanisms to calculate each of these with a bounded stream, and some with an unbounded stream as well. We also extended the basic binary counting algorithm commonly studied to include counting for bursty and non-binary streams. We proved the ε-differential privacy for these algorithms, and provided a way to transform these into an ε-pan private mechanism. Furthermore, we provided various resources to make some of our mechanisms consistent. Finally, we considered some applications of our new mechanisms.

6.3 Acknowledgments

We would like to thank our facilitator, Akshay Pai, for guiding us through the project and for providing helpful feedback on our progress. We would also like to thank Prof. Katrina Ligett for teaching us about data privacy and for recommending papers for our research.

7. REFERENCES

[1] C. Dwork, T. Pitassi, M. Naor, G. Rothblum. *Differential privacy under continual observation.* In Proceedings of the 42nd ACM Symposium on Theory of Computing (STOC '10), pages 715-724, 2010.

[2] J. Bolot, N. Fawaz, S. Muthukrishnan, A. Nikolov, N. Taft. *Private decayed predicate sums on streams.* In Proceedings of the 16th International Conference on Database Theory (ICDT '13), pages 284-295, 2013.

[3] T-H. H. Chan, E. Shi, D. Song. *Private and continual release of statistics.* ACM Transactions on Information and System Security, 14(3):1-24, 2011.

[4] C. Dwork, M. Naor, T. Pitassi, G. Rothblum, S. Yekhanin. *Pan-private streaming algorithms.* In Proceedings of the 1st Symposium on Innovations in Computer Science (ICS 2010), 2010.

[5] D. Mir, S. Muthukrishnan, A. Nikolov, R. Wright. *Pan-private algorithms via statistics on sketches.* In Proceedings of the 30th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS '11). ACM, 2011.

[6] J. McCusker. *Running Standard Deviations.* Subliminal Messages: A science blog for all scientists, even amateurs. 31 Jul, 2008. 15 May, 2013. <http://subluminal.wordpress.com/2008/07/31/running-standard-deviations/#more-15>.

Algorithm 2: Binary Mechanism $\mathcal{B}$ for Non-Binary Streams

Assumptions: The input stream $\sigma \in \{0, 1, \ldots, N\}^{\mathbb{N}}$ is a sequence of non-negative integers with upper bound $N \in \mathbb{Z}_+$, where $\sigma(t) = 0$ when there is no data at a particular time t.

Definitions:

- $Bin(t)$ - The binary representation of t.
- $Bin_i(t)$ - The ith digit in $Bin(t)$, where $Bin_0(t)$ is the least significant digit, such that $t = \sum_j Bin_j(t) \cdot 2^j$.
- α_i - A `p-sum` $= \Sigma\left[t - 2^i + 1, t\right]$ involving 2^i items, with $i \in \mathbb{Z}$. Contains the count of the data.
- $\hat{\alpha_i}$ - The noisy version of α_i.

Inputs:

- $N \in \mathbb{N}$ - An upper bound on each item in the stream.
- $T \in \mathbb{N}$ - An upper bound on the time.
- ε - The privacy parameter.
- $\sigma \in \{0, 1, \ldots, N\}^{\mathbb{N}}$ - The input stream, where $\sigma(t)$ is the value of the data point at time t.

Output: At each time t step, output estimate $\mathcal{B}(t)$.

```
/* Initialize the p-sums and noisy p-sums to 0 */
forall the $i$ do
    $\alpha_i \leftarrow 0$
    $\hat{\alpha_i} \leftarrow 0$
end

$\varepsilon' \leftarrow \varepsilon / N \log T$

/* Go through each time step up to the upper bound T */
for $t \leftarrow 1$ to $T$ do
    Express $t$ in binary form: $t' \leftarrow Bin(t) = \sum_j Bin_j(t) \cdot 2^j$.
    Let $i := \min j : Bin_j(t) \neq 0$.

    /* Store the count as a p-sum */
    $\alpha_i \leftarrow \sum_{j<i} \alpha_j + \sigma(t)$

    /* The previous values of the p-sums are overwritten to ensure privacy and reduce storage */
    for $j \leftarrow 0$ to $i - 1$ do
        $\alpha_j \leftarrow 0$
        $\hat{\alpha_j} \leftarrow 0$
    end

    /* Add Laplace noise scaled by the upper bound N to the p-sum to create the noisy p-sum */
    $\hat{\alpha_i} \leftarrow c_i + \mathrm{Lap}(1/\varepsilon')$

    Output the estimate for the count at time $t$.

        $$\mathcal{B}(t) \leftarrow \sum_{j:Bin_j(t)=1} \hat{\alpha_j}$$
end
```

Algorithm 3: Binary Average Mechanism $\mathcal{A}$

Assumptions: The input stream $\sigma(t) = 0$ when there is no data at a particular time t.

Definitions:

- c_i - A `p-sum` $\Sigma\left[t - 2^i + 1, t\right]$ where $i \in \mathbb{Z}$. Contains the count of the data.
- $\hat{c_i}$ - The noisy `p-sum` $\hat{\Sigma}\left[t - 2^i + 1, t\right]$ corresponding to the count.
- s_i - A `p-sum` $\Sigma\left[t - 2^i, t\right]$ where $i \in \mathbb{Z}$. Contains the sum of the data points.
- $\hat{s_i}$ - The noisy `p-sum` $\hat{\Sigma}\left[t - 2^i + 1, t\right]$ corresponding to the sum.
- $Bin(t)$ - The binary representation of t.
- $Bin_i(t)$ - The ith digit in the binary representation of t, where $Bin_0(t)$ is the least significant digit.

Inputs:

- T - An upper bound on the time.
- ε - The privacy parameter.
- $\sigma \in \mathbb{R}^T$ - The input stream, where $\sigma(t)$ is the value of the data point at time t.

Output: At each time t, an estimate for the average of the data $\mathcal{A}$.

```
/* Initialize the p-sums and noisy p-sums to 0 */
forall the $i$ do
    $\alpha_i \leftarrow 0$
    $\hat{\alpha}_i \leftarrow 0$
end

for $t \leftarrow 1$ to $T$ do
    $t' \leftarrow Bin(t)$
    $i :=$ smallest $j$ such that $Bin_j(t) \neq 0$.

    /* Store the count and sum into their respective p-sums */
    $c_i \leftarrow \left(\sum_{j<i} c_j\right) + \text{sgn}\,|\sigma(t)|$
    $s_i \leftarrow \left(\sum_{j<i} s_j\right) + \sigma(t)$

    /* The previous values of the p-sums are overwritten to ensure privacy and reduce storage */
    for $j \leftarrow 0$ to $i - 1$ do
        $c_j \leftarrow 0$
        $\hat{c_j} \leftarrow 0$
        $s_j \leftarrow 0$
        $\hat{s_j} \leftarrow 0$
    end

    /* Add Laplace noise to the p-sums to create the noisy p-sums */
    $\hat{c_i} \leftarrow c_i + \text{Lap}(1/\varepsilon')$
    $\hat{s_i} \leftarrow s_i + \text{Lap}(1/\varepsilon')$

    Output the estimate for the average at time $t$.
        $\mathcal{A}(t) \leftarrow \frac{\sum_{j|Bin_j(t)=1} \hat{s_j}}{\sum_{j|Bin_j(t)=1} \hat{c_j}}$
end
```

Algorithm 4: Binary Variance Mechanism $\mathcal{V}$

Assumptions: The input stream $\sigma(t) = 0$ when there is no data at a particular time t.

Definitions:

- c_i - A `p-sum` $= \Sigma\left[t - 2^i + 1, t\right]$ where $i \in \mathbb{Z}$. Contains the count of the data.
- $\hat{c_i}$ - The noisy `p-sum` $= \hat{\Sigma}\left[t - 2^i + 1, t\right]$ corresponding to the count.
- $\hat{\mu_t}$ - The Binary Average estimate of the data at time t.
- $\hat{\mathcal{P}}$ - The Binary Average estimate of the power sum average at time t.

Inputs:

- T - An upper bound on the time.
- ε - The privacy parameter.
- $\sigma \in \mathbb{R}^T$ - The input stream, where $\sigma(t)$ is the value of the data point at time t.
- $\sigma^2 \in \mathbb{R}^T$ - The power sum input stream, where $\sigma^2(t)$ is the value of the $\sigma(t) \cdot \sigma(t)$.

Output: At each time t, an estimate for the variance of the data $\mathcal{V}$.

```
/* Initialize the p-sums and noisy p-sums to 0 */
forall the $i$ do
    $\alpha_i \leftarrow 0$
    $\hat{\alpha}_i \leftarrow 0$
end

$\varepsilon' \leftarrow \varepsilon / \log T$

/* Go through each time step up to the upper bound T */
for $t \leftarrow 1$ to $T$ do
    /* Store the count into its p-sum */
    $c_i \leftarrow \left(\sum_{j<i} c_j\right) + \operatorname{sgn}|\sigma(t)|$

    /* The previous values of the p-sums are overwritten to ensure privacy and reduce storage */
    for $j \leftarrow 0$ to $i - 1$ do
        $c_j \leftarrow 0$
        $\hat{c_j} \leftarrow 0$
    end

    /* Add Laplace noise to the p-sum to create the noisy p-sum */
    $\hat{c_i} \leftarrow c_i + \mathrm{Lap}(1/\varepsilon')$

    /* Update the estimate of the power sum average using the Binary Average Mechanism */
    $\hat{\mathcal{P}}_t \leftarrow B(\sigma^2)$

    Output the estimate for the variance at time $t$.
    $$\mathcal{V}(t) \leftarrow \frac{\hat{\mathcal{P}}_t(\hat{c_j}) - \hat{c_j}\hat{\mu_t}^2}{\hat{c_j} - 1}$$
end
```