Better Differentially Private Approximate Histograms and Heavy Hitters using the Misra-Gries Sketch

Christian Janos Lebeda
chle@itu.dk
Basic Algorithms Research Copenhagen
IT University of Copenhagen

Jakub Tětek
j.tetek@gmail.com
Basic Algorithms Research Copenhagen
University of Copenhagen

ABSTRACT

We consider the problem of computing differentially private approximate histograms and heavy hitters in a stream of elements. In the non-private setting, this is often done using the sketch of Misra and Gries [Science of Computer Programming, 1982]. Chan, Li, Shi, and Xu [PETS 2012] describe a differentially private version of the Misra-Gries sketch, but the amount of noise it adds can be large and scales linearly with the size of the sketch: the more accurate the sketch is, the more noise this approach has to add. We present a better mechanism for releasing Misra-Gries sketch under (ε, δ)-differential privacy. It adds noise with magnitude independent of the size of the sketch size, in fact, the maximum error coming from the noise is the same as the best known in the private non-streaming setting, up to a constant factor. Our mechanism is simple and likely to be practical. We also give a simple post-processing step of the Misra-Gries sketch that does not increase the worst-case error guarantee. It is sufficient to add noise to this new sketch with less than twice the magnitude of the non-streaming setting. This improves on the previous result for ε-differential privacy where the noise scales linearly to the size of the sketch.

1 INTRODUCTION

Computing the histogram of a dataset is one of the most fundamental tasks in data analysis. This problem has been investigated thoroughly in the differentially private setting [3, 4, 12, 15, 17, 19, 21]. These algorithms start by computing the histogram exactly and then adding noise to ensure privacy. However, in practice, the amount of data is often so large that computing the histogram exactly would be impractical. This is, for example, the case when computing the histogram of high-volume streams such as when monitoring computer networks, online users, financial markets, and similar. In that case, we need an efficient streaming algorithm. Since the streaming algorithm would only compute the histogram approximately, the above-mentioned approach that first computes the exact histogram is unfeasible. In practice, non-private approximate histograms are often computed using the Misra-Gries (MG) sketch [23]. The MG sketch of size k returns at most k items and their approximate frequencies $\hat{f}$ such that $\hat{f}(x) \in [f(x) - n/(k+1), f(x)]$ for all elements x where $f(x)$ is the true frequency and n is the length of the stream. This error is known to be optimal [8]. In this work, we develop a way of releasing a MG sketch in a differentially private way while adding only a small amount of noise. This allows us to efficiently and accurately compute approximate histograms in the streaming setting while not violating users' privacy. This can then be used to solve the heavy hitters problem in a differentially private way. Our result improves upon the work of Chan et al. [11] who also show a way of privately releasing the MG sketch, but who need a greater amount of noise; we discuss this below.

In general, the issue with making approximation algorithms differentially private is that although we may be approximating a function with low global sensitivity, the algorithm itself (or rather the function it implements) may have a much larger global sensitivity. We get around this issue by exploiting the structure of the difference between the MG sketches for neighboring inputs. This allows us to prove that the following simple mechanism ensures (ε, δ)-differential privacy: (1) We compute the Misra-Gries sketch, (2) we add to each counter independently noise distributed as $\mathrm{Laplace}(1/\varepsilon)$, (3) we add to all counters the same value, also distributed as $\mathrm{Laplace}(1/\varepsilon)$, (4) we remove all counters smaller than $1 + 2\ln(3/\delta)/\varepsilon$. Specifically, we show that this algorithm satisfies the following guarantees:

Theorem 11 (simplified). *The above algorithm is (ε, δ)-differentially private, uses $2k$ words of space, and returns a frequency oracle $\hat{f}$ with maximum error of $n/(k+1) + O(\log(1/\delta)/\varepsilon)$* with high probability for δ being sufficiently small.

A construction for a differentially private Misra-Gries sketch has been given before by Chan et al. [11]. However, the more accurate they want their sketch to be (and the bigger it is), their approach has to add *more* noise. The reason is that they directly rely on the global ℓ_1-sensitivity of the sketch. Specifically, if the sketch has size k (and thus error $n/(k+1)$ on a stream of n elements), its global sensitivity is k, and they thus have to add noise of magnitude k/ε. Their mechanism ends up with an error of $O(k\log(d)/\varepsilon)$ for ε-differential privacy with d being the universe size. This can be easily improved to $O(k\log(1/\delta)/\varepsilon)$ for (ε, δ)-differential privacy with a thresholding technique similar to what we do in step (4) of our algorithm above. This also means that they cannot get more accurate than error $\Theta\left(\sqrt{n\log(1/\delta)/\varepsilon}\right)$, no matter what value of k one chooses. We achieve that the biggest error, as compared to the values from the MG sketch, among all elements is $O(\log(1/\delta)/\varepsilon)$ assuming δ is sufficiently small (we show more detailed bounds including the mean squared errors in Theorem 11). This is the same as the best private solution that starts with an exact histogram [21]. In fact, for any mechanism that outputs at most k heavy hitters there exists input with error at least $n/(k+1)$ in the streaming setting [8] and input with error at least $O(\log(\min(d, 1/\delta))/\varepsilon)$ [4] under differential privacy. In Section 6 we discuss how to achieve ε-differential privacy with error $n/(k+1) + O(\log(d)/\varepsilon)$. Therefore the error of our mechanisms is asymptotically optimal for approximate and pure differential privacy, respectively. The techniques used in Section 6 could also be used to get approximate differential privacy, but the resulting sketch would not have strong competitiveness guarantees with respect to the non-private Misra-Gries sketch, unlike the sketch that we give.

Chan et al. [11] use their differentially private Misra-Gries sketch as a subroutine for continual observation and combine sketches with an untrusted aggregator. Those settings are not a focus of our paper but our work can replace their algorithm as the subroutine , leading to better results also for those settings. However, the noise magnitude increases linearly in the number of merges. As a side note, we show that in the case of a trusted aggregator, the approach of [11] can handle merge operations without increasing error. While that approach adds significantly more noise than ours if we do not merge, it can with this improvement perform better when the number of merges is very large (at least proportional to the sketch size).

Another approach that can be used is to use a randomized frequency oracle to recover heavy hitters. However, it seems hard to do this with the optimal error size. In its most basic form [18, Appendix D], this approach needs noise of magnitude $\Theta(\log(d)/\varepsilon)$, even if we have a sketch with sensitivity 1 (the approach increases the sensitivity to $\log(d)$, necessitating the higher noise magnitude), leading to maximum error at least $\Omega(\log(k)\log(d)/\varepsilon)$. [5] show a more involved approach which reduces the maximum error coming from the noise to $\Theta((\log(k) + \log(d))/\varepsilon)$, but at the cost of increasing the error coming from the sketch by a factor of $\log(d)$. This means that even if we had a sketch with error $\Theta(n/k)$ and sensitivity 1, neither of these two approaches would lead to optimal guarantees, unlike the algorithm we give in this paper.

Relation to [7]. Essentially the same result as Theorem 11 has been claimed in [7]. However, their approach ignores the discrepancy between the global sensitivity of a function we are approximating and that of the function the algorithm actually computes. Their mechanism adds noise scaled to the sensitivity of the exact histogram which is 1 when a user contributes a single element to the stream. But as shown by Chan et al. [11] the sensitivity of the Misra-Gries sketch scales linearly with the number of counters in the sketch. The algorithm from [7] thus does not achieve the claimed privacy parameters. Moreover, it seems unlikely this could be easily fixed – not without doing something along the lines of what we do in this paper.

2 TECHNICAL OVERVIEW

Misra-Gries sketch. Since our approach depends on the properties of the MG sketch, we describe it here. Readers familiar with the MG sketch may wish to skip this. We describe the standard version; in Section 5 we use a slight modification, but we do not need that here.

Suppose we receive a sequence of elements from some universe. At any time, we will be storing at most k of these elements. Each stored item has an associated counter, other elements have implicitly their counter equal to 0. When we process an element, we do one of the following three updates: (1) if the element is being stored, increment its counter by 1, (2) if it is not being stored and the number of stored items is $< k$, store the element and set its counter to 1, (3) otherwise decrement all k counters by 1 and remove those that reach 0. The exact guarantees on the output will not be important now, and we will discuss them in Section 5.

Our contributions. We now sketch how to release an MG sketch in a differentially private way.

Consider two neighboring data streams $S = (S_1, \cdots, S_n)$ and $S' = (S_1, \cdots, S_{i-1}, S_{i+1}, \cdots, S_n)$ for some $i \in [n]$. At step $i-1$, the state of the MG sketch on both inputs is exactly the same. MG_S then receives the item S_i while $MG_{S'}$ does not. This either increments one of the counters of MG_S (possibly by adding an element and raising its counter from 0 to 1) or decrements all its counters. In ℓ_1 distance, the vector of the counters thus changes by at most k. One can show by induction that this will stay this way: at any point in time, $\|MG_S - MG_{S'}\|_1 \leq k$. By a standard global sensitivity argument, one can achieve pure DP by adding noise of magnitude k/ε to each count. This is the approach used in [11]. Similarly, we could achieve (ε, δ)-DP by using the Gaussian mechanism [14] with noise magnitude proportional to the ℓ_2 sensitivity, which is $\sup_{S,S'} \|MG_S - MG_{S'}\|_2 \leq \sqrt{k}$. We want to instead achieve noise with magnitude $O(1/\varepsilon)$ at each count. To this end, we need to exploit the structure of $MG_S - MG_{S'}$.

What we just described requires that we add the noise to the counts of all items in the universe, also to those that are not stored in the sketch. This results in the maximum error of all frequencies depending on the universe's size, which we do not want. However, it is known that this can be easily solved under (ε, δ)-differential privacy by only adding noise to the stored items and then removing values smaller than an appropriately chosen threshold [21]. This may introduce additional error – for this reason, we end up with error $O(\log(1/\delta)/\varepsilon)$. As this is a somewhat standard technique, we ignore this in this section, we assume that the sketches MG_S and $MG_{S'}$ store the same set of elements; the thresholding allows us to remove this assumption, while allowing us to add noise only to the stored items, at the cost of only getting approximate DP.

We now focus on the structure of $MG_S - MG_{S'}$. After we add to MG_S the element S_i, it either holds (1) that $MG_S - MG_{S'}$ is a vector of all 0's and one 1 or (2) that $MG_S - MG_{S'} = 1^k$ [1]. We show by induction that this will remain the case as more updates are done to the sketches (note that the remainders of the streams are the same). We do not sketch the proof here, as it is quite technical.

How do we use the structure of $MG_S - MG_{S'}$ to our advantage? We add noise twice. First, we independently add to each counter noise distributed as $\mathrm{Laplace}(1/\varepsilon)$. Second, we add to all counters the same value, also distributed as $\mathrm{Laplace}(1/\varepsilon)$. That is, we release $MG_S + \mathrm{Laplace}(1/\varepsilon)^{\otimes k} + \mathrm{Laplace}(1/\varepsilon)1^k$ [2]. Intuitively speaking, the first noise hides the difference between S and S' in case (1) and the second noise hides the difference in case (2). We now sketch why this is so for worse constants: $2/\varepsilon$ in place of $1/\varepsilon$. When proving this formally, we use a more technical proof which leads to the better constant.

We now sketch why this is differentially private. Let m_S be the mean of the counters in MG_S for S being an input stream. We may represent MG_S as $(MG_S - m_S 1, m_S)$ (note that there is a bijection between this representation and the original sketch). We now argue that the ℓ_1-sensitivity of this representation is < 2 (treating the representation as a $k+1$-dimensional vector for the

[1]We use 1^k the denote the dimension k vector of all ones.

[2]For D being a distribution, we use $D^{\otimes k}$ to denote the k-dimensional distribution consisting of k independent copies of D.

sake of computing the ℓ_1 distances). Consider the first case. In that case, the averages $m_S, m_{S'}$ differ by $1/k$. As such, $MG_S - m_S 1^k$ and $MG_{S'} - m_{S'} 1^k$ differ by $1/k$ at $k-1$ coordinates and by $1-1/k$ at one coordinate. The overall ℓ_1 change of the representation is thus

$$(k-1) \cdot \frac{1}{k} + (1-1/k) + 1/k = 2 - 1/k < 2.$$

Consider now the second case when $MG_S - MG_{S'} = 1^k$. Thus, $MG_S - m_S = MG'_S - m_{S'}$. At the same time $|m_S - m_{S'}| = 1$. This means that the ℓ_1 distance between the representations is 1. Overall, the ℓ_1-sensitivity of this representation is < 2.

This means that adding noise from $\mathrm{Laplace}(2/\varepsilon)^{\otimes k+1}$ to this representation of MG_S will result in ε-differential privacy. The resulting value after adding the noise is $(MG_S - m_S 1 + \mathrm{Laplace}(2/\varepsilon)^{\otimes k}$, $m_S + \mathrm{Laplace}(2/\varepsilon))$. In the original vector representation of MG_S, this corresponds to $MG_S + \mathrm{Laplace}(2/\varepsilon)^{\otimes k} + \mathrm{Laplace}(2/\varepsilon)1^k$ and, by postprocessing, releasing this value is also differentially private. But this is exactly the value we wanted to show is differentially private!

3 PRELIMINARIES

Setup of this paper. We use $\mathcal{U}$ to denote a universe of elements. We assume that $\mathcal{U}$ is a totally ordered set of size d. That is, $\mathcal{U} = [d]$ where $[d] = \{1, \ldots, d\}$. Given a stream $S \in \mathcal{U}^{\mathbb{N}}$ we want to estimate the frequency in S of each element of $\mathcal{U}$. Our algorithm outputs a set $T \subseteq \mathcal{U}$ of keys and a frequency estimate c_i for all $i \in T$. The value c_j is implicitly 0 for any $j \notin T$. Let $f(x)$ denote the true frequency of x in the stream S. Our goal is to minimize the largest error between c_x and $f(x)$ among all $x \in \mathcal{U}$.

Differential privacy. Differential privacy is a rigorous definition for describing the privacy loss of a randomized mechanism introduced by Dwork et al. [15]. Intuitively, differential privacy protects privacy by restricting how much the output distribution can change when replacing the input from one individual. This is captured by the definition of neighboring datasets. We use the add-remove neighborhood definition for differential privacy.

Definition 1 (Neighboring Streams). *Let S be a stream of length n. Two streams S and S' are neighboring denoted $S \sim S'$ if there exists an i such that $S = (S'_1, \ldots, S'_{i-1}, S'_{i+1}, \ldots, S'_{n+1})$ or $S' = (S_1, \ldots, S_{i-1}, S_{i+1}, \ldots, S_n)$.*

Definition 2 (Differential privacy [14]). *A randomized mechanism $\mathcal{M} : \mathcal{U}^{\mathbb{N}} \to \mathcal{R}$ satisfies (ε, δ)-differential privacy if and only if for all pairs of neighboring streams $S \sim S'$ and all measurable sets of outputs $Z \subseteq \mathcal{R}$ it holds that*

$$\Pr[\mathcal{M}(S) \in Z] \leq e^{\varepsilon} \Pr[\mathcal{M}(S') \in Z] + \delta \ .$$

Samples from a Laplace distribution are used in many differential private algorithms, most notably the Laplace mechanism [15]. We write $\mathrm{Laplace}(b)$ to denote a random variable with a Laplace distribution with scale b centered around 0. We sometimes abuse notation and write $\mathrm{Laplace}(b)$ to denote the value of a random variable drawn from the distribution.

Definition 3 (Laplace distribution). *The probability density and cumulative distribution functions of the Laplace distribution centered around 0 with scale parameter b are $f_b(x) = \frac{1}{2b} e^{-|x|/b}$, and $\Pr[\mathrm{Laplace}(b) \leq x] = \frac{1}{2} e^{x/b}$ if $x < 0$ and $1 - \frac{1}{2} e^{-x/b}$ for $x \geq 0$.*

4 RELATED WORK

Chan et al. [11] shows that the global ℓ_1-sensitivity of a Misra-Gries sketch is $\Delta_1 = k$. (They actually show that the sensitivity is $k+1$ but they use a different definition of neighboring datasets that assumes n is known. Applying their techniques under our definition yields sensitivity k.) They achieve privacy by adding Laplace noise with scale k/ε to all elements in the universe and keep the top-k noisy counts. This gives an expected maximum error of $O(k \log(d)/\varepsilon)$ with ε-DP for d being the universe size. They use the algorithm as a subroutine for continual observation and merge sketches with an untrusted aggregator. Those settings are not a focus of our paper but our work can replace their algorithm as the subroutine when approximate differential privacy is acceptable.

Böhler and Kerschbaum [7] worked on differential private heavy hitters with no trusted server by using secure multi-party computation. One of their algorithms adds noise to the counters of a Misra-Gries sketch. They avoid adding noise to all elements in the universe by removing noisy counts below a threshold which adds an error of $O(\log(1/\delta)/\varepsilon)$. This is a useful technique for hiding differences in keys between neighboring sketches that removes the dependency on d in the error. Unfortunately, as stated in the introduction their mechanism uses the wrong sensitivity. The sensitivity of the sketch is k.If the magnitude of noise and the threshold are increased accordingly the error of their approach is $O(k \log(k/\delta)/\varepsilon)$.

If we ignore the memory restriction in the streaming setting, the problem is the same as the top-k problem [10, 13, 22, 25]. The problem we solve can also be seen as a generalization of the sparse histogram problem. This has been investigated in [3, 4, 12, 21]. Notably, Balcer and Vadhan [4] provides a lower bound showing that for any (ε, δ)-differentially private mechanism that outputs at most k counters, there exists input such that the expected error for some elements is $\Omega(\min(\log(d/k)/\varepsilon, \log(1/\delta)/\varepsilon, n))$ (assuming $\varepsilon^2 > \delta$). The noise that we add in fact matches the second branch of the min over all elements.

A closely related problem is that of implementing frequency oracles in the streaming setting under differential privacy. This has been studied in e.g. [18, 24, 31]. These approaches do not directly return the heavy hitters. The simplest approach for finding the heavy hitters is to iterate over the universe which might be unfeasible. However, there are constructions for finding heavy hitters with frequency oracles more efficiently (see Bassily et al. [5]). However, as we discussed in the introduction, the approach of [5] leads to worse maximum error than what we get unless the sketch size is very large and the universe size is small.

The heavy hitters problem has also received a lot of attention in local differential privacy, starting with the paper introducing the RAPPOR mechanism [16] and continuing with [5, 9, 26, 28–30]. This problem is practically relevant, it is used for example by Apple to find commonly used emojis [2]. The problem has also been recently investigated when using cryptographic primitives [32].

[6, 27] have recently given general frameworks for designing differentially private approximation algorithms; however, if used naively, they are not very efficient for releasing multiple values

(not more efficient than using composition) and they are thus not suitable for the heavy hitters problem.

5 DIFFERENTIALLY PRIVATE MISRA-GRIES

In this section, we present our algorithm for releasing Misra-Gries summaries. We say that two input streams S_1, S_2 are neighboring if one can be obtained from the other by removing one element. This definition is convenient in that it allows us to use the algorithm even if the input length is not public knowledge.

We first present our variant of the non-private Misra-Gries sketch in Algorithm 1 and later show how we add noise to achieve (ε, δ)-differential privacy. The algorithm we use differs slightly from most implementations of MG in that we do not remove elements that have weight 0 until we need to re-use the counter. This will allow us to achieve privacy with slightly lower error.

At all times, k counters are stored as key-value pairs. We initialize the sketch with dummy keys that are not part of $\mathcal{U}$. This guarantees that we never output any elements that are not part of the stream, assuming we remove the dummy counters as post-processing.

The algorithm processes the elements of the stream one at a time. At each step one of three updates is performed: (1) If the next element of the stream is already stored the counter is incremented by 1. (2) If there is no counter for the element and all k counters have a value of at least 1 they are all decremented by 1. (3) Otherwise, one of the elements with a count of zero is replaced by the new element.

In case (3) we always remove the smallest element with a count of zero. This allows us to limit the number of keys that differ between sketches for neighboring streams as shown in Lemma 5. The choice of removing the minimum element is arbitrary but the order of removal must be independent of the stream so that it is consistent between neighboring datasets. The limit on differing keys allows us to obtain a slightly lower error for our private mechanism. However, it is still possible to apply our mechanism with standard implementations of MG. We discuss this in Section 5.1.

Algorithm 1: Misra-Gries (MG)

Input: Positive integer k and stream $S \in \mathcal{U}^{\mathbb{N}}$
$T \leftarrow \{d+1, \ldots, d+k\}$ // Start with k dummy counters
$c_i \leftarrow 0$ for all $i \in T$
foreach $x \in S$ **do**
    if $x \in T$ **then** // Branch 1
        $c_x \leftarrow c_x + 1$
    else if $c_i \geq 1$ *for all* $i \in T$ **then** // Branch 2
        $c_i \leftarrow c_i - 1$ for all $i \in T$
    else // Branch 3
        Let y be the smallest key satisfying $c_y = 0$
        $T \leftarrow (T \cup \{x\}) \setminus \{y\}$
        $c_x \leftarrow 1$
    end
end
return T, c

The same guarantees about correctness hold for our version of the MG sketch, as for the original version. This can be easily shown, as the original version only differs in that it immediately removes any key whose counter is zero. Since the counters for items not in the sketch are implicitly zero, one can see by induction that the estimated frequencies by our version are exactly the same as those in the original version. We still need this modified version, as the set of keys it stores is different from the original version, which we use below. The fact that the returned estimates are the same however allows us to use the following fact

Fact 4 (Bose et al. [8]). *Let $\hat{f}(x)$ be the frequency estimates given by an MG sketch of size k for n being the input size. Then for all $x \in \mathcal{U}$, it holds $\hat{f}(x) \in [f(x) - n/(k+1), f(x)]$, where $f(x)$ is the true frequency of x.*

Note that this is optimal for any mechanism that returns a set of at most k elements. This is easy to see for an input stream that contains $k+1$ distinct elements each with frequency $n/(k+1)$ since at least one element must be removed.

We now analyze the value of $MG_S - MG_{S'}$ for S, S' being neighboring inputs. We will then use this in order to prove privacy. As mentioned in Section 4, Chan et al. [11] showed that the ℓ_1-sensitivy for Misra-Gries sketches is k. They show that this holds after processing the element that differs for neighboring streams and use induction to show that it holds for the remaining stream. Our analysis follows a similar structure. We expand on their result by showing that the sets of stored elements for neighboring inputs differ in at most two keys when using our variant of Misra-Gries. We then show how all this can be used to get differential privacy with only a small amount of noise.

Lemma 5. *Let $T, c \leftarrow \mathrm{MG}(k, S)$ and $T', c' \leftarrow \mathrm{MG}(k, S')$ be the outputs of Algorithm 1 on a pair of neigboring streams S, S' such that S' is obtained by removing an element from S. Then $|T \cap T'| \geq k - 2$ and all counters not in the intersection have a value of at most 1. Moreover, it holds that either (1) $c_i = c'_i - 1$ for all $i \in T'$ and $c_j = 0$ for all $j \notin T'$ or (2) there exists an $i \in T$ such that $c_i = c'_i + 1$ and $c_j = c'_j$ for all $j \neq i$.*

Proof. Let $S \sim S'$ be pair of neighboring streams where S' is obtained by removing one element from S. We show inductively that the Lemma holds for any such S and S'. Let $w = T - T'$ and $w' = T' - T$ denote the set of keys that are only in one sketch. Let c_0 and c'_0 denote the smallest element with a zero count in the respective sketch when such an element exists. Then at any point during execution after processing the element removed from S the sketches are in one of the following states:

(S1) $T = T'$ and $c_i = c'_i - 1$ for all $i \in T$.
(S2) There exist $x_1, x_2 \in \mathcal{U}$ such that $w = \{x_1\}$ and $w' = \{x_2\}$, $c_{x_1} = 0$, $c'_{x_2} = 1$ and $c_i = c'_i - 1$ for all $i \in T \cap T'$.
(S3) $T = T'$ and there exists $x_1 \in \mathcal{U}$ such that $c_{x_1} = c'_{x_1} + 1$ and $c_i = c'_i$ for all $i \in T \setminus \{x_1\}$.
(S4) There exist $x_1, x_2 \in \mathcal{U}$ such that $w = \{x_1\}$ and $w' = \{x_2\}$, $c_{x_1} = 1$, $c'_{x_2} = 0$ and $c_i = c'_i$ for all $i \in T \cap T'$.
(S5) There exist $x_1, x_2, x_3 \in \mathcal{U}$ such that $c_{x_1} = c'_{x_1} + 1$, $w = \{x_2\}$, $w = \{x_3\}$, $c_{x_2} = 0$, $c'_{x_3} = 0$ and $c_i = c'_i$ for all $i \in T \cap T' \setminus \{x_1\}$.
(S6) There exist $x_1, x_2, x_3, x_4 \in \mathcal{U}$ such that $w = \{x_1, x_2\}$ and $w' = \{x_3, x_4\}$, $c_{x_1} = 1$, $c_{x_2} = c'_{x_3} = c'_{x_4} = 0$, $c_i = c'_i$ for all $i \in T \cap T'$ and $x_4 = c'_0$.

Let $x = S_i$ be the element in stream S which is not in stream S'. Since the streams are identical in the first $i-1$ steps the sketches are clearly the same before step i. If there is a counter for x in the sketch we execute Branch 1 and the result corresponds to state S3. If there is no counter for x and no zero counters we execute Branch 2 and the result corresponds to state S1. Otherwise, the 3rd branch of the algorithm is executed and c_0 is replaced by x which corresponds to state S4. Therefore we must be in one of the states S1, S3, or S4 for $T, c \leftarrow \mathrm{MG}(k, (S_1, \ldots, x_i))$ and $T', c' \leftarrow \mathrm{MG}(k, (S'_1, \ldots, x'_{i-1}))$.

We can then prove inductively that the Lemma holds since the streams are identical for the elements $(S_{i+1}, \ldots, S_n)$. We have to consider the possibility of each of the branches being executed for both sketches. The simplest case is when the element has a counter in both sketches and Branch 1 is executed on both inputs. This might happen in all states and we stay in the same state after processing the element. But many other cases lead to new states.

Below we systematically consider all outcomes of processing an element $x \in \mathcal{U}$ when the sketches start in each of the above states. When processing each element, one of the three branches is executed for each sketch. This gives us up to 9 combinations to check, although some are impossible for certain states. Furthermore, when Branch 3 is executed we often have to consider which element is replaced as it leads to different states. We refer to T, c and T', c' as sketches 1 and 2, respectively.

State S1: If $x \in T$ then $x \in T'$ and Branch 1 is executed for both sketches. Similarly, if Branch 2 is executed for sketch 1 it must also be executed for sketch 2 as all counters are strictly larger. Therefore we stay in state S1 in both cases. It is impossible to execute Branch 3 for sketch 2 since all counters are non-zero by definition. As such the final case to consider is when $x \notin T$ and there is a counter with value 0 in sketch 1. In this case, we execute Branch 3 for sketch 1 and Branch 2 for sketch 2. This result in state S4.

State S2: If $x \in T$ we execute Branch 1 on sketch 1 and there are two possible outcomes. If $x \neq x_1$ we also execute Branch 1 on sketch 2 and remain in state S2. If $x = x_1$ we execute Branch 2 on sketch 2 in which case there are no changes to T or T' but now $c_x = 1$ and $c_i = c'_i$ for all $i \in T \cap T'$. As such, we transitioned to state S4.

Since $c_{x_1} = 0$ by definition Branch 2 is never executed on sketch 1 and Branch 3 is never executed on sketch 2 as all counters are non-zero. If $x = x_2$ Branch 3 is executed on sketch 1 and Branch 1 is executed for sketch 2. If $c_0 = x_1$ the sketches have the same keys after processing x and transition to state S1, otherwise if $c_0 \neq x_1$ the sketches still differ for one key and remain in state S2.

Finally, if Branch 3 is executed on sketch 1 and Branch 2 is executed on sketch 2 we again have two possibilities. In both cases, the sketches store the same count on all elements from $T \cap T'$ after processing x. If $c_0 = x_1$ it is removed from T and replaced by x with $c_x = 1$ which corresponds to state S4. If $c_0 \neq x_1$ we must have that $c_0 \in T \cap T'$. The two sketches differ on exactly two keys after processing x. One of the two keys stored in sketch 2 that are not in sketch 1 must be the minimum zero key since the elements c_0 and x_2 now have counts of zero in sketch 2 and c_0 was the minimum zero key in $T \cap T'$. Therefore we transition to state S6.

State S3: The simplest case is $x \in T$ since then $x \in T'$ and Branch 1 is executed for both sketches. If Branch 2 is executed for sketch 1 and $c'_{x_1} \neq 0$ Branch 2 is also executed for sketch 2. For both cases, we remain in state S3. Instead, if $c'_{x_1} = 0$ Branch 3 is executed for sketch 2. Since all counters are decremented for sketch 1 and x_1 is replaced in sketch 2 we transition to state S2. Lastly, if Branch 3 is executed for sketch 1 it is also executed for sketch 2 and there are two cases. If the same element is removed we remain in state S3. Otherwise, if x_1 is replaced in sketch 2 we transition to state S4.

State S4: Since sketch 2 contains a counter with a value of zero in the remaining states Branch 2 is never executed. If Branch 1 is executed for both sketches we stay in the same state as always but if $x = x_1$ Branch 1 is executed for sketch 1 and Branch 3 is executed for sketch 2. If $c'_0 = x_2$ then $T = T'$ after processing x and we transition to state S3. If $c'_0 \neq x_2$ another element is removed from sketch 2 which must also have a count of zero in sketch 1 and we go to state S5.

If Branch 2 is executed on sketch 1 we know that c'_{x_2} must be the only zero counter in sketch 2. Therefore it does not matter if Branch 1 or 3 is executed on sketch 2. For both cases, we set $c_x = 1$ and the sketches differ in one key which corresponds to state S2.

Finally, if Branch 3 is executed on sketch 1 we again have two cases that lead to the same state. If $x = x_2$ or $c'_0 = x_2$ the current counter c'_{x_2} is replaced but the counter that was replaced in sketch 1 remains in sketch 2. Otherwise, we have $c_0 = c'_0$ and we update the same counter in sketches 1 and 2. Therefore we remain in state S4 in both cases.

State S5: Since by definition both sketches contain counters with a value of zero, the second branch is never executed while in this state. If $x \in T \cap T'$ we remain in the same state as always. We have to consider the cases where $x = x_2$, $x = x_3$, and $x \notin T \cup T'$. The resulting state depends on the elements that are replaced in the sketch. For $x = x_2$ we transition to state S3 if $c'_0 = x_3$ and remain in state S5 otherwise. The same argument shows that we transition to state S3 or S5 based on c_0 if $x = x_3$. When $x \notin T \cup T'$ we execute Branch 3 on both sketches. We transition to state S3 only if $c_0 = x_2$ and $c'_0 = x_3$ since otherwise both sketches still have a zero counter that is not stored in the other sketch and we stay in state S5.

State S6: Similar to state S5, the second branch is never executed from this state. Here we have to consider the five cases where $x \in T \cap T'$, $x = x_1$, $x = x_2$, $x \in w'$, and $x \notin T \cup T'$. We know that x_4 is replaced whenever $x \notin T'$. If $x \in T \cap T'$ we execute Branch 1 on both sketches and remain in state S6. If $x = x_1$ we transition to state S5 and for $x = x_2$ we transition to state S4. When $x \in w'$ there are two possibilities. We always have $c_x = c'_x$ after updating. If $c_0 = x_2$ the sketches will share $k-1$ keys and transition to state S4. If $c_0 \neq x_2$ then another element that has a count of zero in both sketches is replaced in sketch 1. We know that either this element or the remaining zero-valued element of w' must be the smallest zero-valued element in sketch 2. Therefore we remain in state S6.

The final case to consider is when $x \notin T \cup T'$. In this case Branch, 3 is executed for sketch 2 and x_4 is replaced with x in T'. If $c_0 = x_2$ we transition to state S4. Otherwise, either x_3 or the element that was replaced from sketch 1 must be the minimum element with a count of zero in sketch 2. As such, we remain in state S6. □

Next, we consider how to add noise to release the Misra-Gries sketch under differential privacy. Recall that Chan et al. [11] achieves privacy by adding noise to each counter which scales with k. We avoid this by utilizing the structure of sketches for neighboring streams shown in Lemma 5. We sample noise from $\mathrm{Laplace}(1/\varepsilon)$ independently for each counter, but we also sample one more random variable from the same distribution which is added to all counters. Small values are then discarded using a threshold to hide differences in the sets of stored keys between neighboring inputs. This is similar to the technique used by e.g. [21]. The algorithm takes the output from MG as input. We sometimes write $\mathrm{PMG}(k, S)$ as a shorthand for $\mathrm{PMG}(\mathrm{MG}(k, S))$.

Algorithm 2: Private Misra-Gries (PMG)

Parameters: $\varepsilon, \delta > 0$
Input: Output from Algorithm 1: $T, c \leftarrow \mathrm{MG}(k, S)$

1 $\tilde{T} \leftarrow \emptyset$ Sample $\eta \sim \mathrm{Laplace}(1/\varepsilon)$
2 **foreach** $x \in T$ **do**
3     $c_x \leftarrow c_x + \eta + \mathrm{Laplace}(1/\varepsilon)$
4   **if** $c_x \geq 1 + 2\ln(3/\delta)/\varepsilon$ **then**
5         $\tilde{T} \leftarrow \tilde{T} \cup \{x\}$
6         $\tilde{c}_x \leftarrow c_x$
7   **end**
8 **end**
9 **return** $\tilde{T}, \tilde{c}$

We prove the privacy guarantees in three steps. First, we show that changing either a single counter or all counters by 1 does not change the output distribution significantly (Corollary 7). This assumes that, for neighboring inputs, the set of stored elements is exactly the same. By Lemma 5, we have that the difference between the sets of stored keys is small and the corresponding counters are ≤ 1. Relying on the thresholding, we bound the probability of outputting one of these keys (Lemma 8). Finally, we combine these two lemmas to show that the privacy guarantees hold for all cases (we do this in Lemma 9).

Lemma 6. *Let us have $x, x' \in \mathbb{R}^k$ such that one of the following three cases holds*

(1) $\exists i \in [k]$ such that $|x_i - x'_i| = 1$ and $x_j = x'_j$ for all $j \neq i$.
(2) $x_i = x'_i - 1$ for all $i \in [k]$.
(3) $x_i = x'_i + 1$ for all $i \in [k]$.

Then we have for any measurable set Z that

$$\begin{aligned}&\Pr[x + \mathrm{Laplace}^{\otimes k}(1/\varepsilon) + \mathrm{Laplace}(1/\varepsilon)1^k \in Z] \\ &\qquad \leq e^{\varepsilon} \Pr[x' + \mathrm{Laplace}(1/\varepsilon)^{\otimes k} + \mathrm{Laplace}(1/\varepsilon)1^k \in Z]\end{aligned}$$

Proof. We first focus on the simpler case (1). It holds by the law of total expectation that

$$\begin{aligned}\Pr[x + \mathrm{Laplace}(1/\varepsilon)^{\otimes k} + \mathrm{Laplace}(1/\varepsilon)1^k \in Z] &= \\ E_{N \sim \mathrm{Laplace}(1/\varepsilon)}\left[\Pr[\mathrm{Laplace}(1/\varepsilon)^{\otimes k} \in Z - x - N1^k | y]\right] &\leq \\ e^{\varepsilon} E_{N \sim \mathrm{Laplace}(1/\varepsilon)}\left[\Pr[\mathrm{Laplace}(1/\varepsilon)^{\otimes k} \in Z - x' - N1^k) | y]\right] &= \\ e^{\varepsilon} \Pr[x' + \mathrm{Laplace}(1/\varepsilon)^{\otimes k} + \mathrm{Laplace}(1/\varepsilon)1^k \in Z]&\end{aligned}$$

where to prove the inequality, we used that for any measurable set A, it holds $\Pr[\mathrm{Laplace}(1/\varepsilon)^{\otimes k} \in A] \leq e^{\varepsilon} \Pr[\mathrm{Laplace}(1/\varepsilon)^{\otimes k} \in A - \phi]$ for any $\phi \in \mathbb{R}^k$ with $\|\phi\|_1 \leq 1$ (see [15]). Specifically, we have set $A = Z - x - N1^k$ and $\phi = x - x'$ such that $\|\phi\|_1 = 1$.

We now focus on the cases (2), (3). We will prove below that for x, x' satisfying one of the conditions (2), (3) and for any measurable A, Z and $N_1 \sim \mathrm{Laplace}(1/\varepsilon)^{\otimes k}$, it holds

$$\begin{aligned}&\Pr[x + N_1 + \mathrm{Laplace}(1/\varepsilon)1^k \in Z | N_1 \in A] \\ &\qquad \leq e^{\varepsilon} \Pr[x' + N_1 + \mathrm{Laplace}(1/\varepsilon)1^k \in Z | N_1 \in A]\end{aligned}$$

This allows us to argue like above:

$$\begin{aligned}\Pr[x + \mathrm{Laplace}(1/\varepsilon)^{\otimes k} + \mathrm{Laplace}(1/\varepsilon)1^k \in Z] &= \\ E_{N_1 \sim \mathrm{Laplace}(1/\varepsilon)^{\otimes k}}\left[\Pr[x + N_1 + \mathrm{Laplace}(1/\varepsilon)1^k \in Z | N_1]\right] &\leq \\ e^{\varepsilon} E_{N_1 \sim \mathrm{Laplace}(1/\varepsilon)^{\otimes k}}\left[\Pr[x' + N_1 + \mathrm{Laplace}(1/\varepsilon)1^k \in Z | N_1]\right] &= \\ e^{\varepsilon} \Pr[x' + \mathrm{Laplace}(1/\varepsilon)^{\otimes k} + \mathrm{Laplace}(1/\varepsilon)1^k \in Z]&\end{aligned}$$

which would conclude the proof. Let $g : \mathbb{R} \to \mathbb{R}^k$ be the function $g(a) = a1^k$ and define $g^{-1}(B) = \{a \in \mathbb{R} | g(a) \in B\}$ and note that g is measurable. We focus on the case (2); the same argument works for (3) as we discuss below. It holds

$$\begin{aligned}\Pr[x + N_1 + \mathrm{Laplace}(1/\varepsilon)1^k \in Z | N_1 \in A] &= \\ \Pr[\mathrm{Laplace}(1/\varepsilon)1^k \in Z - x - N_1 | N_1 \in A] &= \\ \Pr[\mathrm{Laplace}(1/\varepsilon) \in g^{-1}(Z - x - N_1) | N_1 \in A] &= \\ \Pr[\mathrm{Laplace}(1/\varepsilon) \in g^{-1}(Z - x' - 1^k - N_1) | N_1 \in A] &= \\ \Pr[\mathrm{Laplace}(1/\varepsilon) \in g^{-1}(Z - x' - N_1) - 1 | N_1 \in A] &\leq \\ e^{\varepsilon} \Pr[\mathrm{Laplace}(1/\varepsilon) \in g^{-1}(Z - x' - N_1) | N_1 \in A] &= \\ e^{\varepsilon} \Pr[\mathrm{Laplace}(1/\varepsilon)1^k \in Z - x' - N_1 | N_1 \in A] &= \\ e^{\varepsilon} \Pr[x' + N_1 + \mathrm{Laplace}(1/\varepsilon)1^k \in Z | N_1 \in A]&.\end{aligned}$$

To prove the inequality, we again used the standard result that for any measurable A, $\Pr[\mathrm{Laplace}(1/\varepsilon) \in A] \leq e^{\varepsilon} \Pr[\mathrm{Laplace}(1/\varepsilon) \in A - 1]$ holds. The same holds for $A + 1$; this allows us to use the exact same argument in case (3), in which the proof is exactly the same except that -1 on lines 4,5 of the manipulations is replaced by $+1$. □

Corollary 7. *Let T, c and T', c' be two sketches such that $T = T'$ and one of following holds:*

(1) $\exists i \in T$ such that $|c_i - c'_i| = 1$ and $c_j = c'_j$ for all $j \neq i$.
(2) $c_i = c'_i - 1$ for all $i \in T$.
(3) $c_i = c'_i + 1$ for all $i \in T$.

Then for any measurable set of outputs Z, we have:

$$\Pr[\mathrm{PMG}(T, c) \in Z] \leq e^{\varepsilon} \Pr[\mathrm{PMG}(T', c') \in Z]$$

Proof. Consider first a modified algorithm PMG' that does not perform the thresholding: that is, if we remove the condition on line 4. It can be easily seen that PMG' only takes the vector c and releases $c + \mathrm{Laplace}(1/\varepsilon)^{\otimes k} + \mathrm{Laplace}(1/\varepsilon)1^k$. We have just shown in Lemma 6 that this means that

$$\Pr[\mathrm{PMG}'(T, c) \in Z'] \leq e^{\varepsilon} \Pr[\mathrm{PMG}'(T', c') \in Z']$$

for any measurable Z'.

Let $\tau(x) = x$ for $x \geq 1 + 2\ln(3/\delta)/\varepsilon$ and 0 otherwise. Since $\mathrm{PMG}(T, c) = \tau(PMG'(T, c))$, it then holds

$$\Pr[\mathrm{PMG}(T, c) \in Z] = \Pr[PMG'(T, c) \in \tau^{-1}(Z)] \leq$$
$$e^{\varepsilon} \Pr[\mathrm{PMG}'(T', c') \in \tau^{-1}(Z)] = e^{\varepsilon} \Pr[\mathrm{PMG}(T', c') \in Z]$$

as we wanted to show. □

Lemma 8. *Let T, c and T', c' be two sketches of size k and let $\hat{T} = T \cap T'$. If we have that $|\hat{T}| \geq k - 2$, $c_i = c'_i$ for all $i \in \hat{T}$, and for all $x \notin \hat{T}$, it holds $c_x, c'_x \leq 1$. Then for any measurable set Z, it holds*

$$\Pr[\mathrm{PMG}(T, c) \in Z] \leq \Pr[\mathrm{PMG}(T', c') \in Z] + \delta$$

Proof. Let $\mathrm{PMG}'(T, c)$ denote a mechanism that runs $\mathrm{PMG}(T, c)$ and performs postprocessing by discarding any elements not in $\hat{T}$. It is easy to see that (a) $\Pr[\mathrm{PMG}'(T, c) \in Z] = \Pr[\mathrm{PMG}'(T', c') \in Z]$ since the input sketches are identical for all elements in $\hat{T}$. Moreover, for any output $\tilde{T}, \tilde{c} \leftarrow \mathrm{PMG}(T, c)$ for which $\tilde{T} \subseteq \hat{T}$, the postprocessing does not affect the output. This gives us the following inequalities: (b) $\Pr[\mathrm{PMG}(T, c) \in Z] \leq \Pr[\mathrm{PMG}'(T, c) \in Z] + \Pr[\tilde{T} \not\subseteq \hat{T}]$ and (c) $\Pr[\mathrm{PMG}'(T', c') \in Z] \leq \Pr[\mathrm{PMG}(T, c) \in Z] + \Pr[\tilde{T}' \not\subseteq \hat{T}]$. Combining $(a) - (c)$, we get the inequality $\Pr[\mathrm{PMG}(T, c) \in Z] \leq \Pr[\mathrm{PMG}(T', c') \in Z] + \Pr[\tilde{T} \not\subseteq \hat{T}] + \Pr[\tilde{T} \not\subseteq \hat{T}']$.

As such, the Lemma holds if $\Pr[\tilde{T} \not\subseteq \hat{T}] + \Pr[\tilde{T}' \not\subseteq \hat{T}] \leq \delta$. That is, it suffices to prove that with probability at most δ any noisy count for elements not in $\hat{T}$ is at least $1 + 2\ln(3/\delta)/\varepsilon$. The noisy count for such a key can only exceed the threshold if one of the two noise samples added to the key is at least $\ln(3/\delta)/\varepsilon$. From Definition 3 we have $\Pr[\mathrm{Laplace}(1/\varepsilon) \geq \ln(3/\delta)/\varepsilon] = \delta/6$. There are at most 4 keys not in $\hat{T}$ which are in $T \cup T'$ and therefore at most 6 noise samples affect the probability of outputting such a key (the 4 individual Laplace noises and then the 2 global Laplace noises, one for each sketch). By a union bound the probability that any of these samples exceeds $\ln(3/\delta)/\varepsilon$ is at most δ. □

We are now ready to prove the privacy guarantee of Algorithm 2.

Lemma 9. *Algorithm 2 is (ε, δ)-differentially private for any k.*

Proof. The Lemma holds if and only if for any pair neighboring of neighboring streams $S \sim S'$ and any measurable set Z we have:

$$\Pr[\mathrm{PMG}(T, c) \in Z] \leq e^{\varepsilon} \Pr[\mathrm{PMG}(T', c') \in Z] + \delta,$$

where $T, c \leftarrow \mathrm{MG}(k, S)$ and $T', c' \leftarrow \mathrm{MG}(k, S')$ denotes the non-private sketches for each stream.

We prove the guarantee above using an intermediate sketch that "lies between" T, c and T', c'. The sketch has support T' and we denote the counters as $\hat{c}$. By Lemma 5, we know that $|T \cap T'| \geq k - 2$. We will now come up with some conditions on $\hat{c}$ such that if these conditions hold, the lemma follows. We will then prove the existence of such $\hat{c}$ below. Assume that $\hat{c}_i = c_i$ for all $i \in T$ and $\hat{c}_j \leq 1$ for all $j \in T' \setminus T$. Lemma 8 then tells us that

$$\Pr[\mathrm{PMG}(T, c) \in Z] \leq \Pr[\mathrm{PMG}(T', \hat{c}) \in Z] + \delta.$$

Assume also that one of the required cases for Corollary 7 holds between c' and $\hat{c}$. We have

$$\Pr[\mathrm{PMG}(T', \hat{c}) \in Z] \leq e^{\varepsilon} \Pr[\mathrm{PMG}(T', c') \in Z].$$

Therefore, if such a sketch $T', \hat{c}$ exists for all S and S' the lemma holds since

$$\Pr[\mathrm{PMG}(T, c) \in Z] \leq \Pr[\mathrm{PMG}(T', \hat{c}) \in Z] + \delta$$
$$\leq e^{\varepsilon} \Pr[\mathrm{PMG}(T', c') \in Z] + \delta \ .$$

It remains to prove the existence of $\hat{c}$ such that $\hat{c}_i = c_i$ for all $i \in T$ and $\hat{c}_j \leq 1$ for all $j \in T' \setminus T$ and such that one of the conditions $(1) - (3)$ of Corollary 7 holds between $\hat{c}$ and c'. We first consider neighboring streams where S' is obtained by removing an element from S. From Lemma 5 we have two cases to consider. If $c_i = c'_i - 1$ for all $i \in T'$ we simply set $\hat{c} = c$. Recall that we implicitly have $c_i = 0$ for $i \notin T$. Therefore the sketch satisfies the two conditions above since $\hat{c}_i = c_i$ for all $i \in \mathcal{U}$ and condition (2) of Corollary 7 holds. In the other case where $c_i = c'_i + 1$ for exactly one $i \in T$ there are two possibilities. If $i \in T'$ we again set $\hat{c} = c$. When $i \notin T'$ there must exist at least one element $j \in T'$ and $j \notin T$ with $c'_j = 0$. We set $\hat{c}_j = 1$ and $\hat{c}_i = c_i$ for all $i \neq j$. In both cases $\hat{c}_i = c_i$ for all $i \in T$ and $\hat{c}_j$ is at most one for $j \notin T$. There is exactly one element with a higher count in $\hat{c}$ than c' which means that condition (1) of Corollary 7 holds.

If S is obtained by removing an element from S' the cases from Lemma 5 are flipped. If $c_i - 1 = c'_i$ for all $i \in T$ and $c'_j = 0$ for $j \notin T$ we set $\hat{c}_i = c_i$ if $i \in T$ and $\hat{c}_i = 1$ otherwise. It clearly holds that $\hat{c}_i = c_i$ for all $i \in T$ and $\hat{c}_j \leq 1$ for all $j \notin T$. Since $\hat{c}_i = c'_i + 1$ for all $i \in T'$ condition (3) of Corollary 7 holds. Finally, if $c_i + 1 = c'_i$ for exactly one $i \in T'$ we simply set $\hat{c} = c$. $\hat{c}_i = c_i$ clearly holds for all $i \in T$ and condition (1) of Corollary 7 holds between $\hat{c}$ and c'. □

Next, we analyze the error compared to the non-private sketch. We state the error in terms of the largest error among all elements of the sketch. Recall that we implicitly say that the count is zero for any element not in the sketch.

Lemma 10. *Let $\tilde{T}, \tilde{c} \leftarrow \mathrm{PMG}(T, c)$ denote the output of Algorithm 2 for any sketch T, c with $|T| = k$. Then with probability at least $1 - \beta$ we have*

$$\tilde{c}_x \in \left[c_x - \frac{2\ln\left(\frac{k+1}{\beta}\right)}{\varepsilon} - 1 - \frac{2\ln\left(3/\delta\right)}{\varepsilon}, c_x + \frac{2\ln\left(\frac{k+1}{\beta}\right)}{\varepsilon}\right]$$

for all $x \in T$ and $\tilde{c}_x = 0$ for all $x \notin T$.

Proof. The two sources of error are the noise samples and the thresholding step. We begin with a simple bound on the absolute value of the Laplace distribution.

$$\Pr\left[|\mathrm{Laplace}(1/\varepsilon)| \geq \frac{\ln((k+1)/\beta)}{\varepsilon}\right] =$$
$$2 \cdot \Pr\left[\mathrm{Laplace}(1/\varepsilon) \leq -\frac{\ln((k+1)/\beta)}{\varepsilon}\right] = \beta/(k+1) \ .$$

Since $k + 1$ samples are drawn we know by a union bound that the absolute value of all samples is bounded by $\ln((k+1)/\beta)/\varepsilon$ with probability at least $1 - \beta$. As such the absolute error from the Laplace samples is at most $2\ln((k+1)/\beta)/\varepsilon$ for all $x \in T$ since two samples are added to each count. Removing noisy counts below the threshold potentially adds an additional error of at most

$1 + 2\ln(3/\delta)/\varepsilon$. It is easy to see that $\tilde{c}_x = 0$ for all $x \notin T$ since the algorithm never outputs any such elements. □

Theorem 11. *$\mathrm{PMG}(k, S)$ satisfies (ε, δ)-differential privacy. Let $f(x)$ denote the frequency of $x \in \mathcal{U}$ in S and let $\hat{f}(x)$ denote the estimated frequency of x from the output of $\mathrm{PMG}(k, S)$. For any element x with $f(x) = 0$ we have $\hat{f}(x) = 0$ and with probability at least $1 - \beta$ we have for all $x \in \mathcal{U}$ that*

$$\hat{f}(x) \in \left[f(x) - \frac{2\ln\left(\frac{k+1}{\beta}\right)}{\varepsilon} - 1 - \frac{2\ln(3/\delta)}{\varepsilon} - \frac{|S|}{k+1}, f(x) + \frac{2\ln\left(\frac{k+1}{\beta}\right)}{\varepsilon}\right]$$

Moreover, the algorithm outputs all x, such that $\hat{f}(x) > 0$ and there are at most k such elements. For any fixed $x \in U$, the mean squared error is $E[(\hat{f}(x) - f(x))^2] \leq 3\left(1 + \frac{2+2\ln(3/\delta)}{\varepsilon} + \frac{|S|}{k+1}\right)^2$. $\mathrm{PMG}(k, S)$ uses $2k$ words of memory.

Proof. The space complexity is clearly as claimed, as we are storing at any time at most k items and counters. We focus on proving privacy and correctness.

If $f(x) = 0$ we know that $x \notin T$ where T is the keyset after running Algorithm 1. Since Algorithm 2 outputs a subset of T we have $\hat{f}(x) = 0$. The first part of the Theorem follows directly from Fact 4 and Lemmas 9 and 10.

We now bound the mean squared error. There are three sources of error. Let r_1 be the error coming from the Laplace noise, r_2 from the thresholding, and r_3 the error made by the MG sketch. Then

$$E[(\hat{f}(x) - f(x))^2] = E[(r_1 + r_2 + r_3)^2] \leq 3(E[r_1^2] + E[r_2^2] + E[r_3^2])$$

by equivalence of norms (for any dimension n vector v, $\|v\|_1 \leq \sqrt{n}\|v\|_2$). The errors r_2, r_3 are deterministically bounded $r_2 \leq 1 + 2\ln(3/\delta)/\varepsilon$ and $r_3 \leq |S|/(k+1)$. $E[r_1^2]$ is the variance of the Laplace noise; we added two independent noises each with scale $1/\varepsilon$ and thus variance $2/\varepsilon^2$ for a total variance of $4/\varepsilon^2$. This finishes the proof. □

5.1 Privatizing standard versions of Misra-Gries

The privacy of our mechanism as presented in Algorithm 2 relies on our variant of the Misra-Gries algorithm. Our sketch can contain elements with a count of zero. However, elements with a count of zero are removed in the standard version of the sketch. As such, sketches for neighboring datasets can differ for up to k keys if one sketch stores k elements with a count of 1 and the other sketch is empty. It is easy to change Algorithm 2 to handle these implementations. We simply increase the threshold to $1 + 2\ln\left(\frac{k+1}{2\delta}\right)/\varepsilon$ since the probability of outputting any of the k elements with a count of 1 is bounded by δ.

5.2 Tips for practitioners

Here we discuss some technical details to keep in mind when implementing our mechanism.

The output of the Misra-Gries algorithm is an associative array. In Algorithm 2 we add appropriate noise such that the associative array can be released under differential privacy. However, for some implementations of associative arrays such as hash tables the order in which keys are added affects the data structure. Using such an implementation naively violates differential privacy but it is easily solved either by outputting a random permutation of the key-value pairs or using a fixed order e.g. sorted by key.

We present our mechanism with noise sampled from the Laplace distribution. However, the distribution is defined for real numbers which cannot be represented on a finite computer. This is a known challenge and precision-based attacks still exist on popular implementations [20]. Since the output of MG is discrete the distribution can be replaced by the Geometric mechanism [19] or one of the alternatives introduced in [4]. Our mechanism would still satisfy differential privacy but it might be necessary to change the threshold in Algorithm 2 slightly to ensure that Lemma 8 still holds.

Lastly, it is worth noting that the analysis for Lemma 8 is not tight. We bound the probability of outputting a small key by bounding the value of all samples by $\ln(3/\delta)/\varepsilon$ which is sufficient to guarantee that the sum of any two samples does not exceed $2\ln(3/\delta)/\varepsilon$. This simplifies the proof and presentation significantly however one sample could exceed $\ln(3/\delta)/\varepsilon$ without any pair of samples exceeding $2\ln(3/\delta)/\varepsilon$. A tighter analysis would improve the constant slightly which might matter for practical applications.

6 PURE DIFFERENTIAL PRIVACY

In this section, we discuss how to achieve ε-differential privacy. We cannot use our approach from Section 5 where we add the same noise to all keys because the set of stored keys can differ between sketches for neighboring datasets. Instead, we achieve privacy by adding noise to all elements of $\mathcal{U}$ scaled to the ℓ_1-sensitivity. Chan et al. [11] showed that the sensitivity of Misra-Gries sketches scales with the number of counters. We show that a simple postprocessing step reduces the sensitivity of the sketch to 2 and the worst-case guarantee of the sketch is still $n/(k+1)$ where $n = |S|$. This allows us to achieve an error of $n/(k+1) + O(\log(d)/\varepsilon)$.

The ℓ_1-sensitivity scales with the size of the sketch since the counts can differ by 1 for all k elements between neighboring datasets. This happens when the decrement step is executed on a given input one fewer or one more time than on a neighboring input. We get around this case by post-processing the sketch before adding noise. We first run the Misra-Gries algorithm on the stream but we count how many times the counters were decremented. That is, we count the number of times Branch 2 of Algorithm 1 was executed and denote this count as γ. The Misra-Gries algorithm decrements the counters at most $\lfloor n/(k+1) \rfloor$ times. We use this fact by first adding γ and then subtracting $n/(k+1)$ from each counter in the sketch. We then remove all elements with negative counters. Although we increase the error of the sketch for some datasets, the worst-case error guarantee is still the same as each count has been decremented by at most $n/(k+1)$. Next, we show how this post-processing step reduces the ℓ_1-sensitivity to 2.

Let $S \sim S'$ denote any pair of neighboring streams where S' is obtained by removing one element from S. Consider the effect of running the following procedure on the Misra-Gries sketches for both streams (1) add γ and γ' to the counters of MG_S and $\mathrm{MG}_{S'}$, respectively (2) subtract $|S|/(k+1)$ from the counters in both sketches (3) remove any negative counters from both sketches. It can be shown that the new sketches are either identical or differ by 1 in a single counter. Specifically, we may use the argument from the proof of Lemma 5 to argue that we end in one of the 6 states introduced in that proof before running the procedure. One may verify that the claim holds in all 6 states. Specifically, we get $\gamma = \gamma' + 1$ in the first 2 states and $\gamma = \gamma'$ for the final 4 states. The post-processing step we introduced in the previous paragraph uses the length of the stream which differs by 1 between S and S'. As such, there is an additional difference of $1/(k+1)$ for each counter. The ℓ_1-sensitivity is bounded by 2 since $1 + k/(k+1) < 2$.

We achieve ε-differential privacy by adding noise to our new sketch. We essentially use the same technique as Chan et al. [11] but the noise no longer scale linearly in k as the sensitivity is bounded by 2. Specifically, we add noise sampled from $\mathrm{Laplace}(2/\varepsilon)$ independently to the count of each element from $\mathcal{U}$ and release the top-k noisy counts. A simple union bound shows us that with probability at least $1 - \beta$ the absolute value of all samples is bounded by $2\ln(d/\beta)/\varepsilon$. Note that it might be unfeasible to actually sample noise for each element when $\mathcal{U}$ is large; we refer to previous work on how to implement this more efficiently [4, 11, 12].

It is worth noting that the low sensitivity of the post-processed sketch can also be utilized under (ε, δ)-differential privacy. We can use an approach similar to [21]. They add noise to all non-zero counters and remove noisy counts below a threshold to hide small counters. This would require a threshold with a small dependence on k as neighboring sketches might disagree on all keys. However, [3] extended the technique to real-valued vectors by probabilistically rounding elements with a value less than the ℓ_1-sensitivity. If we apply their technique directly we get a threshold of $4 + 2\ln(1/\delta)/\varepsilon$. This approach has error guarantees that match those from Theorem 11 up to constant factors. However, this approach has worse guarantees than Algorithm 2 when comparing to the non-private Misra-Gries sketch. By Lemma 10 the error of Algorithm 2 is $O(\log(1/\delta)/\varepsilon)$ with high probability (for sufficiently small δ). Here the error is $n/(k+1) + O(\log(1/\delta)/\varepsilon)$ since we subtract up to $n/(k+1)$ from the counters before adding noise.

7 PRIVATIZING MERGED SKETCHES

In practice, it is often important that we may merge sketches. This is for example commonly used when we have a dataset distributed over many servers. Each dataset consists of multiple streams in this setting, and we want to compute an aggregated sketch over all streams. We say that datasets are neighboring if we can obtain one from the other by removing a single element from one of the streams. If the aggregator is untrusted we must add noise to each sketch before performing any merges. This is the setting in [11] and we can run their merging algorithm. However, since we add noise to each sketch the error scales with the number of sketches. In particular, the error from the thresholding step of Algorithm 2 scales linearly in the number of sketches for worst-case input. In the rest of this section, we consider the setting where aggregators are trusted and we want to output a differentially private merged sketch.

Agarwal et al. [1] introduced the following simple merging algorithm in the non-private setting. Given two Misra-Gries sketches $T_1, c_1 \leftarrow \mathrm{MG}(k, S_1)$ and $T_2, c_2 \leftarrow \mathrm{MG}(k, S_2)$ they first compute the sum of all counters $c_1 + c_2$. There are up to $2k$ counters at this point. They subtract the value of the $k+1$'th largest counter from all elements. Finally, any non-positive counters are removed leaving at most k counters. They show that merged sketches have the same worst-case guarantee as non-merged Misra-Gries sketches. That is, if we compute a Misra-Gries sketch for each stream $(S_1, \dots, S_m)$ and merge them into a single sketch, the frequency estimate of all elements is at most $N/(k+1)$ less than the true frequency. Here N is the total length of all streams. This holds for any order of merging and the streams do not need to have the same length.

Unfortunately, the structure between neighboring sketches where either a single counter or exactly k counters differ by 1 breaks down when merging. Therefore we cannot run Algorithm 2 on the merged sketch. However, as we show below, the global sensitivity of merged sketches is independent of the number of merges. The sensitivity only depends on the number of counters. We first show a property for a single merge operation; this will allow us to bound the sensitivity for any number of merges. Note that unlike in the previous section, we do not limit the number of keys that differ between sketches and we do not store keys with a count of zero.

Lemma 12. *Let T_1, c_1, T_1', c_1' and T_2, c_2 denote Misra-Gries sketches of size k and denote the sketches merged with the algorithm above as $\hat{T}, \hat{c} \leftarrow \mathrm{Merge}(T_1, c_1, T_2, c_2)$ and $\hat{T}', \hat{c}' \leftarrow \mathrm{Merge}(T_1', c_1', T_2, c_2)$. If $T_1' \subseteq T_1$ and $c_{1i} - c_{1i}' \in \{0, 1\}$ for all $i \in T_1$ then at least one of the following holds (1) $\hat{T}' \subseteq \hat{T}$ and $\hat{c}_i - \hat{c}_i' \in \{0, 1\}$ for all $i \in \hat{T}$ or (2) $\hat{T} \subseteq \hat{T}'$ and $\hat{c}_i' - \hat{c}_i \in \{0, 1\}$ for all $i \in \hat{T}'$.*

Proof. Let $\bar{c}$ and $\bar{c}'$ denote the merged counters before subtracting and removing values. Then clearly $\bar{c}_i - \bar{c}_i' \in \{0, 1\}$ for all $i \in \mathcal{U}$. Therefore we have that $\bar{c}_{k+1} - \bar{c}_{k+1}' \in \{0, 1\}$ where $\bar{c}_{k+1}$ deenotes the value of the $k+1$'th largest element in $\bar{c}$. Note that it does not matter if the $k+1$'th largest value is a different key. Let $\hat{c}$ and $\hat{c}'$ denote the counters after subtracting the $k+1$'th largest element. if $\bar{c}_{k+1} = \bar{c}_{k+1}'$ we subtract the same value from each sketch and we have $\hat{c}_i - \hat{c}_i' \in \{0, 1\}$ for all $i \in \mathcal{U}$. If $\bar{c}_{k+1} - \bar{c}_{k+1}' = 1$ we subtract one more from each count in $\hat{c}$ and we have $\hat{c}_i' - \hat{c}_i \in \{0, 1\}$ for all $i \in \mathcal{U}$. □

Corollary 13. *Let $(S_1, \dots, S_m)$ and $(S_1', \dots, S_m')$ denote two sets of streams such that $S_i \sim S_i'$ for one $i \in [m]$ and $S_j = S_j'$ for any $j \neq i$. Let T, c and T', c' be the result of merging Misra-Gries sketches computed on both sets of streams in any fixed order. Then c and c' differ by 1 for at most k elements and agree on all other counts.*

Proof. It is clearly true for sketches of a pair of neighboring datasets by Lemma 5. It holds by induction after each merging operation by Lemma 12. □

Since the ℓ_1-sensitivity is k we can use the algorithm in [11] that adds noise with magnitude k/ε to all elements in $\mathcal{U}$ and keeps the top-k noisy counts. If we only add noise to non-zero counts we can hide that up to k keys can change between neighboring inputs with a threshold. The two approaches have expected maximum error compared to the non-private sketch of $O(k\log(d)/\varepsilon)$ and $O(k\log(k/\delta)/\varepsilon)$, respectively.

ACKNOWLEDGMENT

The authors are affiliated with Basic Algorithms Research Copenhagen (BARC), supported by the VILLUM Foundation grant 16582.

We would like to thank Rasmus Pagh for helpful discussions. We also thank Martin Aumüller for giving feedback on an early draft of this paper.

REFERENCES

[1] Pankaj K. Agarwal, Graham Cormode, Zengfeng Huang, Jeff M. Phillips, Zhewei Wei, and Ke Yi. Mergeable summaries. *ACM Trans. Database Syst.*, 38(4):26, 2013. doi: 10.1145/2500128. URL https://doi.org/10.1145/2500128.

[2] Apple. Differential privacy overview - apple. URL https://www.apple.com/privacy/docs/Differential_Privacy_Overview.pdf.

[3] Martin Aumüller, Christian Janos Lebeda, and Rasmus Pagh. Differentially private sparse vectors with low error, optimal space, and fast access. In Yongdae Kim, Jong Kim, Giovanni Vigna, and Elaine Shi, editors, *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*, pages 1223–1236. ACM, 2021. doi: 10.1145/3460120.3484735. URL https://doi.org/10.1145/3460120.3484735.

[4] Victor Balcer and Salil P. Vadhan. Differential privacy on finite computers. *J. Priv. Confidentiality*, 9(2), 2019. doi: 10.29012/jpc.679.

[5] Raef Bassily, Kobbi Nissim, Uri Stemmer, and Abhradeep Guha Thakurta. Practical locally private heavy hitters. *Advances in Neural Information Processing Systems*, 30, 2017.

[6] Jeremiah Blocki, Elena Grigorescu, Tamalika Mukherjee, and Samson Zhou. How to make your approximation algorithm private: A black-box differentially-private transformation for tunable approximation algorithms of functions with low sensitivity. *arXiv preprint arXiv:2210.03831*, 2022.

[7] Jonas Böhler and Florian Kerschbaum. Secure multi-party computation of differentially private heavy hitters. In Yongdae Kim, Jong Kim, Giovanni Vigna, and Elaine Shi, editors, *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*, pages 2361–2377. ACM, 2021. doi: 10.1145/3460120.3484557. URL https://doi.org/10.1145/3460120.3484557.

[8] Prosenjit Bose, Evangelos Kranakis, Pat Morin, and Yihui Tang. Bounds for frequency estimation of packet streams. In Jop F. Sibeyn, editor, *SIROCCO 10: Proceedings of the 10th Internaltional Colloquium on Structural Information Complexity, June 18-20, 2003, Umeå Sweden*, volume 17 of *Proceedings in Informatics*, pages 33–42. Carleton Scientific, 2003.

[9] Mark Bun, Jelani Nelson, and Uri Stemmer. Heavy hitters and the structure of local privacy. *ACM Transactions on Algorithms (TALG)*, 15(4):1–40, 2019.

[10] Ricardo Silva Carvalho, Ke Wang, Lovedeep Gondara, and Chunyan Miao. Differentially private top-k selection via stability on unknown domain. In *Conference on Uncertainty in Artificial Intelligence*, pages 1109–1118. PMLR, 2020.

[11] T.-H. Hubert Chan, Mingfei Li, Elaine Shi, and Wenchang Xu. Differentially private continual monitoring of heavy hitters from distributed streams. *IACR Cryptol. ePrint Arch.*, page 218, 2012. URL http://eprint.iacr.org/2012/218.

[12] Graham Cormode, Cecilia M. Procopiuc, Divesh Srivastava, and Thanh T. L. Tran. Differentially private summaries for sparse data. In *ICDT*, pages 299–311. ACM, 2012. doi: 10.1145/2274576.2274608.

[13] David Durfee and Ryan M Rogers. Practical differentially private top-k selection with pay-what-you-get composition. *Advances in Neural Information Processing Systems*, 32, 2019.

[14] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3-4):211–407, 2014. doi: 10.1561/0400000042. URL https://doi.org/10.1561/0400000042.

[15] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.

[16] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, pages 1054–1067, 2014.

[17] Quan Geng, Peter Kairouz, Sewoong Oh, and Pramod Viswanath. The staircase mechanism in differential privacy. *IEEE Journal of Selected Topics in Signal Processing*, 9(7):1176–1184, 2015. doi: 10.1109/JSTSP.2015.2425831.

[18] Badih Ghazi, Noah Golowich, Ravi Kumar, Rasmus Pagh, and Ameya Velingker. On the power of multiple anonymous messages. *IACR Cryptol. ePrint Arch.*, page 1382, 2019. URL https://eprint.iacr.org/2019/1382.

[19] Arpita Ghosh, Tim Roughgarden, and Mukund Sundararajan. Universally utility-maximizing privacy mechanisms. *SIAM Journal on Computing*, 41(6):1673–1693, 2012.

[20] Samuel Haney, Damien Desfontaines, Luke Hartman, Ruchit Shrestha, and Michael Hay. Precision-based attacks and interval refining: how to break, then fix, differential privacy on finite computers. *CoRR*, abs/2207.13793, 2022. doi: 10.48550/arXiv.2207.13793. URL https://doi.org/10.48550/arXiv.2207.13793.

[21] Aleksandra Korolova, Krishnaram Kenthapadi, Nina Mishra, and Alexandros Ntoulas. Releasing search queries and clicks privately. In *WWW*, pages 171–180. ACM, 2009. doi: 10.1145/1526709.1526733.

[22] Darakhshan J. Mir, S. Muthukrishnan, Aleksandar Nikolov, and Rebecca N. Wright. Pan-private algorithms via statistics on sketches. In Maurizio Lenzerini and Thomas Schwentick, editors, *Proceedings of the 30th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2011, June 12-16, 2011, Athens, Greece*, pages 37–48. ACM, 2011. doi: 10.1145/1989284.1989290. URL https://doi.org/10.1145/1989284.1989290.

[23] J. Misra and David Gries. Finding repeated elements. *Science of Computer Programming*, 2(2):143–152, 1982. ISSN 0167-6423. doi: https://doi.org/10.1016/0167-6423(82)90012-0. URL https://www.sciencedirect.com/science/article/pii/0167642382900120.

[24] Rasmus Pagh and Mikkel Thorup. Improved utility analysis of private countsketch. *CoRR*, abs/2205.08397, 2022. doi: 10.48550/arXiv.2205.08397. URL https://doi.org/10.48550/arXiv.2205.08397.

[25] Gang Qiao, Weijie Su, and Li Zhang. Oneshot differentially private top-k selection. In *International Conference on Machine Learning*, pages 8672–8681. PMLR, 2021.

[26] Zhan Qin, Yin Yang, Ting Yu, Issa Khalil, Xiaokui Xiao, and Kui Ren. Heavy hitter estimation over set-valued data with local differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 192–203, 2016.

[27] Jakub Tětek. Additive noise mechanisms for making randomized approximation algorithms differentially private. *arXiv preprint arXiv:2211.03695*, 2022.

[28] Tianhao Wang, Ninghui Li, and Somesh Jha. Locally differentially private heavy hitter identification. *IEEE Transactions on Dependable and Secure Computing*, 18 (2):982–993, 2019.

[29] Hao Wu and Anthony Wirth. Asymptotically optimal locally private heavy hitters via parameterized sketches. In *International Conference on Artificial Intelligence and Statistics*, pages 7766–7798. PMLR, 2022.

[30] Dan Zhao, Suyun Zhao, Hong Chen, Ruixuan Liu, Cuiping Li, and Wenjuan Liang. Efficient protocols for heavy hitter identification with local differential privacy. *Frontiers of Computer Science*, 16(5):1–11, 2022.

[31] Fuheng Zhao, Dan Qiao, Rachel Redberg, Divyakant Agrawal, Amr El Abbadi, and Yu-Xiang Wang. Differentially private linear sketches: Efficient implementations and applications. *CoRR*, abs/2205.09873, 2022. doi: 10.48550/arXiv.2205.09873. URL https://doi.org/10.48550/arXiv.2205.09873.

[32] Wennan Zhu, Peter Kairouz, Brendan McMahan, Haicheng Sun, and Wei Li. Federated heavy hitters discovery with differential privacy. In *International Conference on Artificial Intelligence and Statistics*, pages 3837–3847. PMLR, 2020.