Differentially Private Frequency Sketches for Intermittent Queries on Large Data Streams

Sinan Yıldırım*, Kamer Kaya†, Soner Aydın*, Hakan Buğra Erentuğ†
*Industrial Engineering, *Sabancı University*, Istanbul, Turkey
{sinanyildirim, soneraydin}@sabanciuniv.edu
†Computer Science & Engineering, *Sabancı University*, Istanbul, Turkey
{kaya, herentug}@sabanciuniv.edu

2020 IEEE International Conference on Big Data (Big Data) | 978-1-7281-6251-5/20/$31.00 ©2020 IEEE | DOI: 10.1109/BigData50022.2020.9377786

***Abstract*—We propose novel and differentially private versions of Count Sketch, particularly suited for dynamic, intermittent queries for observed frequencies of elements in a universal set. Our algorithms are designed for scenarios where the queries are made intermittently, that is, at different times during the course of the data stream. We explore several approaches, all based on the Laplace mechanism, and ultimately propose an algorithm that is robust and efficiently handles multiple queries at multiple times while keeping its utility at reasonable levels. We demonstrate the performance of the proposed algorithm in various scenarios with a numerical example.**

***Index Terms*—Count Sketch, differential privacy, dynamic and intermittent queries.**

I. Introduction

Sketches are probabilistic data structures that can provide approximate results within mathematically proven error bounds while using orders of magnitude less memory than traditional approaches. They are tailored for streaming data analysis on architectures even with limited memory such as single-board computers that are widely exploited for IoT and edge computing. With the emergence of massive-scale data streams in various fields, the concern of extracting useful information from these data streams while preserving the privacy of individual data becomes an important problem. Differential privacy (DP) provides a framework as a solution in which information about a data set is revealed while, at the same time, the privacy of the individuals that have contributed to the dataset is preserved [1]. This is why leveraging DP for data sketches have been popular in the literature in the last decade [2]–[8]. In this paper, we focus on developing differentially private sketches for count queries. We confine our study to the Count and Count-Min Sketches [9], [10]. The Count Sketch is proposed in [9] as a useful tool for answering frequency queries, by producing unbiased estimators. Similarly, the Count-Min Sketch [10] is proposed for the same task. In the literature, there have been many studies on incorporating either Count Sketch, Count-Min Sketch, or both in a privacy-preserving setting.

In this paper, we tackle the problem of answering intermittent frequency queries regarding the information contained in a data stream while providing DP and keeping the utility of the responses at a reasonable level. What is significant about the setting of "intermittent queries" is the possibility that even between two consecutive *identical* queries, the true answer may have differed due to addition of new individuals' data. The challenge related to data privacy is that the answers to the queries should continually protect the privacy of individuals' data that are included in the data stream at any time of the streaming process, see [18] for a general take on this challenge. Hence, the setting being investigated in this work can be considered as a generalization of the setting which focuses on one-time queries. Overall, the contributions can be summarized as follows:

- We discuss various methods for differentially private counting sketches for intermittent queries that can be considered as reasonable approaches under certain circumstances. However, as our ultimate choice, we propose a single method that we show to be most suitable.
- The algorithms we discuss and propose are based on two different competing approaches. The first approach considers randomly perturbing the cells in the sketch table, while the second group consider perturbing the answer returned from the sketch. We discuss the pros and cons of those approaches in terms of accuracy. We show that while both approaches can compete in a setting where a batch of queries are to be answered at the same time, the first approach is more suited to intermittent queries.
- Despite the algorithms in the paper are presented with the Count Sketch, they can easily be adapted to the Count-Min Sketch, following almost identical steps.

The organization of the paper is as follows: In Section II, we review the definition and basic properties of DP and present the Count and Count-Min Sketches as event-oriented dynamic processes. The main tools and approaches for developing a differentially private Count Sketch are presented in Section III. In Section IV, we present our differentially private sketch tailored for intermittent queries. Section V discusses the related work on sketches providing DP guarantees. In Section VI, we present an experimental study and subsequently comment on its results. In Section VII, we give conclusive remarks and mention some possible extensions of the methodology.

II. Background on Privacy and Sketches

In this section, we provide some background, along with some basic notation, for differential privacy and the count and

This work is supported by TÜBİTAK project number 118E044.

count-min sketches.

A. Differential privacy

In recent years, *differential privacy (DP)* has become a popular framework for achieving privacy-preserving estimates of basic statistics in data sets. We call two data sets $X, X' \in \mathsf{X}$ neighbors if X' is obtained by the addition or deletion of a single entry to or from X. We call $\mathcal{A}$ a randomized algorithm whose output upon taking an input X is a random variable $\mathcal{A}(X)$ taking values from some $\mathcal{S}$.

Definition 1. *We say that $\mathcal{A}$ is (ϵ, δ)-differentially private if, for any pair of neighboring data sets $X, X' \in \mathsf{X}$ from an input set and any subset of output values $S \subseteq \mathcal{S}$, it satisfies [1]*

$$\mathbb{P}\left[\mathcal{A}(X) \in S\right] \leq e^{\epsilon}\mathbb{P}\left[\mathcal{A}(X') \in S\right] + \delta.$$

According to the above inequality, a randomized algorithm is differentially private if the probability distributions for the output obtained from two neighboring databases are '*similar*'. The parameters ϵ and δ determine the *privacy budget*, or *privacy loss*. Those parameters are desired to be as small as possible as far as privacy is concerned.

Assume that a privacy preserving algorithm is required to return the value of a function $\varphi : \mathsf{X} \mapsto \mathbb{R}$ evaluated at the sensitive data set X in a private fashion. One basic way of achieving this is via the *Laplace mechanism* [11, Theorem 1], which relies on the *sensitivity* of this function.

Definition 2. *The sensitivity of $\varphi : \mathsf{X} : \mathbb{R}$ is given by*

$$\nabla_{\varphi} = \sup_{X, X' \in \mathsf{X}} |\varphi(X) - \varphi(X')|.$$

Theorem 1 (Laplace mechanism)**.** *Let $\mathcal{A}$ be an algorithm that returns $\hat{\varphi} = \varphi(X) + v$ on an input $X \in \mathsf{X}$, where $v \sim \text{Laplace}(\nabla_{\varphi}/\epsilon)$. Then $\mathcal{A}$ is ϵ-DP.*

One property of differential privacy is the composition property, which quantifies the privacy loss of multiple uses of a differentially private mechanism.

Theorem 2 (Composition)**.** *Let $\mathcal{S}$ be the output space and X be the input space. Let $\mathcal{A}_1, \ldots, \mathcal{A}_m$ are ϵ-DP algorithms that take $X \in \mathsf{X}$ as input and return random outputs in $\mathcal{S}$ as outputs. Then, let the composition of those algorithms, $\mathcal{A} = (\mathcal{A}_1, \ldots, \mathcal{A}_m)$ return outputs $(S_1, \ldots, S_m)$ is $m\epsilon$-DP. Moreover, this result does not change when $(\mathcal{A}_1, \ldots, \mathcal{A}_m)$ is applied sequentially and each algorithm's output is generated conditional on the outputs of the previous algorithms.*

Another property of differential privacy relevant to our work is that no further privacy loss is suffered by transforming the output through a deterministic algorithm.

Theorem 3 (Post-processing)**.** *Let $\mathcal{A}_1$ be an (ϵ, δ)-DP algorithm with inputs from X and outputs from $\mathcal{S}_1$, and let $\mathcal{A}_2 : \mathcal{S}_1 \mapsto \mathcal{S}$ be deterministic algorithm that does not depend on X. Then, the algorithm $\mathcal{A} = (A_2 o \mathcal{A}_1)$ is (ϵ, δ)-DP.*

B. Count and Count-Min Sketches

Suppose we have a data stream from a universal set $\mathcal{X}$ and its i'th element is x_i. Count and Count-Min Sketches are developed in order to answer queries regarding the count of an element $x \in \mathcal{X}$, i.e., its frequency, in such a data set. That is those sketches answer questions in the form of "*how many times has a certain element occurred so far in the data stream?*". The sketching algorithms summarize a large amount of data into a small table by the help of hash functions. Both Count and Count-Min Sketch use similar type of sketch tables.

a) Count-Min Sketch: This sketch uses a table/matrix C with w columns and d rows where row i of the sketch table is governed by a hash function $h_i : \mathcal{X} \to \{1, \ldots, w\}$ which is chosen independently for each row. Let $C(i, j)$ be the value in the ith row and jth column. The table is initialized with all zeroes, i.e., $C(i, j) = 0$ for all $1 \leq i \leq d$ and $1 \leq j \leq w$. Every time an element x from the stream is received, the sketch is updated according to the procedure given in [10]:

$$C(i, h_i(x)) \leftarrow C(i, h_i(x)) + 1, \quad i = 1, \ldots, d.$$

When an element x is queried, the sketch returns the estimation of x's frequency, i.e., f_x, as

$$\hat{f}_x = \min\{C(1, h_1(x)), \ldots, C(d, h_d(x))\}.$$

b) Count Sketch: The difference of Count Sketch from Count-Min Sketch is that the contribution of each element to C is given as either $+1$ or -1 with equal probability, and the sign of increment is determined by using another hash function, shown as $s_i : \mathcal{X} \mapsto \{-1, 1\}$. Accordingly, the estimate of f_x for an item x is given as the median of the values corresponding to x in C, instead of their minimum [12].

The initialization of the table for Count Sketch is the same, that is, $C(i, j) = 0$ for each $1 \leq i \leq d$, $1 \leq j \leq w$. Upon receipt of an element x, the sketch updates the table as

$$C(i, h_i(x)) \leftarrow C(i, h_i(x)) + s_i(x), \quad i = 1, \ldots, d.$$

When queried, the frequency f_x of any $x \in \mathcal{X}$ is estimated by

$$\hat{f}_x = \text{median}\{s_1(x)C(1, h_1(x)), \ldots, s_d(x)C(d, h_d(x))\}.$$

Although our differentially private sketching algorithms can be implemented with both sketches, we will only focus on the Count Sketch to avoid repetitions.

c) Dynamic system view with intermittent queries: In this paper, we are interested in a dynamic system, where queries can be made while the sketch is still being updated by the stream. Moreover, we assume that queries can be made in arbitrary times and for arbitrary subsets of $\mathcal{X}$ of arbitrary size. That is why we prefer to present a sketching algorithm as an event-oriented *dynamic system* that has two types of events: (i) arrival of an element from the stream, and (ii) a query set $Q \subseteq \mathcal{X}$ whose elements are queried for their frequencies. Algorithm 1 models such a dynamic system and employs a Count Sketch to answer intermittent query sets.

Algorithm 1: Count Sketch in a dynamic system

{Initialization of the table}
for $i = 1, \dots, d$ **do**
  for $j = 1, \dots, w$ **do**
    Set $C(i, j) \leftarrow 0$
{Events start}
repeat
  if *stream element* x *arrives* **then**
    for $i = 1, \dots, d$ **do**
      $C(i, h_i(x)) \leftarrow C(i, h_i(x)) + s_i(x)$
  else if *a set* Q *of queries are made* **then**
    for $x \in Q$ **do**
      Set $S \leftarrow \emptyset$
      for $i = 1, \dots, d$ **do**
        Append S with $s_i(x)C(i, h_i(x))$
      return $\hat{f}_x = \textit{median } S$
until *end of events*;

In this paper, we investigate the ways to modify Algorithm 1 so that it would provide DP while keeping the accuracy at a reasonable rate. In the following section, we present a detailed discussion and results on several approaches.

III. Privacy-preserving count sketch

A privacy preserving algorithm protects privacy generally by adding noise to its calculations. When multiple queries are made about sensitive data, as the case in our setting, usually more noise is required not to exceed a privacy budget, see Theorem 2. As a result, the algorithm's utility inevitably deteriorates. Our main concern is to use noise adding mechanisms in an intelligent way so that the utility of the privacy preserving algorithm deteriorates as slowly as possible.

A. Setting

Here, we provide a setting that places the privacy issue in a certain context. For that, it is firstly necessary to define the neighborhood relations between streams. We assume that the stream elements which are processed by the Count Sketch correspond to distinct individuals, and the sketch is constructed to return the frequency values of individuals who are placed in a certain category with respect to a certain feature. For example, when the data stream is composed of individuals with their residence addresses (so $\mathcal{X}$ is a set of certain type of addresses, such as postal codes), the queries are in form of "*How many people reside in address x?*". We assume in this work that the hash functions used by the sketch are publicly known.

In this setting, two ordered data sets are neighbors if they have all the same elements in the same order, except for the presence or absence of a single element. $X = \{x_1, \dots, x_n\}$ and $X' = \{x_1, \dots, x_{k-1}, x_{k+1}, \dots, x_n\}$ can be given as an example of such sets. The regular Count Sketch does not provide differential privacy for two neighboring data sets, because when a new data point arrives, the cells in C which correspond to this data point are increased or decreased by exactly 1. An '*unfortunate*' query will make this difference reflect on the median calculation, hence violate the privacy.

1) Single query and Laplace mechanism: Notice that the sensitivity of the sketch estimate $\hat{f}_x$ for any element $x \in \mathcal{X}$ is 1, since absence or presence of a single individual in the stream can change the estimate by at most 1. Therefore, when a single query for the frequency of an element $x \in \mathcal{X}$ is made, a simple distortion of the output by using a random value from Laplace$(1/\epsilon)$, yielding the noisy output

$$\hat{f}_x + v, \quad v \sim \text{Laplace}(1/\epsilon), \tag{1}$$

is sufficient for providing ϵ-differential privacy.

B. Multiple queries

Assume a set of queries $Q \subseteq \mathcal{X}$ is received and the frequency estimations for all the items in Q are requested. Such *multiple queries* can be quite relevant to practical implementations. For example, a single user may want to know frequencies of a set of elements, which constitute a range or whom the user thinks are *heavy-hitters*, i.e., items that frequently appear in data streams. Alternatively, Q may be the union of different query sets made by different users, i.e.,

$$Q = \bigcup_i Q^{(i)}$$

where $Q^{(i)}$ is the set of elements whose frequencies are queried by user i. If the actual timestamps of those query sets are sufficiently close to each other, they can be considered as simultaneous. For example, the dynamic system may perform batching to answer queries and consider successive periods of a buffer time, Δ, where all the queries made within the same period are considered simultaneous.

1) Median perturbation: Let n_Q denote the size of the set Q. A straightforward application of the composition theorem for DP, Theorem 2, yields a query response for each element in Q that satisfies ϵ/n_Q-DP. This can be achieved by

$$\hat{f}_x = f_x + v_x, \quad v_x \sim \text{Laplace}(n_Q/\epsilon), \quad x \in Q,$$

where v_x is sampled independently for each $x \in Q$. While this basic approach is reasonable when n_Q is small, the estimates get unreliable quickly as n_Q increases. However, Laplace(n_Q/ϵ) may be an (unnecessarily) conservative choice for the amount of noise to be added. We explain why so in the following.

Let X, X' be neighbour data sets with the different element denoted by x^*. We will call $\{(1, h_1(x^*)), \dots, (d, h_d(x^*))\}$ the set of *sensitive cells* for this pair X, X'. One can argue that it is possible that not all queries in Q require a *sensitive cell*. Given Q, we should instead add noise based on *the maximum possible number of sensitive queries*, i.e., queries for which at least one sensitive cell is to be used for the median calculation. Let this number be m_Q. Then, each query in Q can be responded by

$$\hat{f}_x = f_x + v_x, \quad v_x \sim \text{Laplace}(m_Q/\epsilon), \quad x \in Q. \tag{2}$$

providing ϵ-DP by the composition theorem, Theorem 2.

We provide a proposition below that characterizes m_Q. Let $j^x_{1:d} = (j^x_1, \ldots, j^x_d)$ be the vector of column indices of the table C's cells for x, i.e., the hash functions on x yield reads from cells $(1, j^x_1), \ldots, (d, j^x_d)$ in C. The number m_Q can be written as[1]

$$m_Q = \max_{j_{1:d} \in \{1,\ldots,w\}^d} \sum_{x \in Q} [(j_1 = j^x_1) \vee \ldots \vee (j_d = j^x_d)] \quad (3)$$

The problem of finding m_Q in (3) can be shown to be a generalized version of maximum satisfiability (MAXSAT) problem, where the variables are non-binary but integers, and therefore, it is NP-hard. For moderate values of n_Q, one can compute m_Q with a brute-force grid search over a maximum of $(n_Q)^d$ combinations for $j_{1:d}$ in (3). However, for large n_Q, we should resort to an approximation to find m_Q, which has to be an upper bound in order to satisfy ϵ-DP. An upper bound can be computed as in Algorithm 2.

Algorithm 2: An upper bound estimate for m_Q

Input: Query set Q
Output: Estimate of m_Q, $\hat{m}_Q$
{Initialize an auxiliary $d \times w$ table N}
$N(i,j) = 0$ for all $1 \le i \le d$, $1 \le j \le w$.
for $i = 1, \ldots, d$ **do**
  Compute the maximum frequency for the i'th row:
  for $x \in Q$ **do**
    Increment $N(i, h_i(x)) = N(i, h_i(x)) + 1$.
  Calculate $m_i = \max_{j=1,\ldots,w} N(i,j)$.
return $\hat{m}_Q = \sum_{i=1}^d m_i$.

Proposition 1. *We have $m_Q \le \hat{m}_Q$, where $\hat{m}_Q$ is computed in Algorithm 2. Furthermore, if $n_Q \le d$ we have $m_Q = n_Q$.*

Proof. Let $j^*_i = \arg\max_{j=1,\ldots,w} N(i,j)$, where $N(i,j)$ is calculated in Algorithm 2. The worst case scenario, that is the scenario where a maximum number of the elements in Q touch a sensitive cell, is realized when $(1, j^*_1), \ldots, (d, j^*_d)$ are the sensitive cells, there are m_i queries in Q that touch the cell (i, j^*_i), for $i = 1, \ldots, d$, and those queries are all distinct (that is, no query touches more than one sensitive cell). This leads to $m_Q = \hat{m}_Q$. In the case of $n_Q \le d$, it is trivial by definition that we have $m_Q \le n_Q$. For the equality itself, letting $x_1, \ldots, x_{n_Q}$ be the elements in Q, note that one can always select $j_1, \ldots, j_d$ such that j_i coincides with $h_i(x_i)$ for $i = 1, \ldots, n_Q \le d$, making $m_Q = n_Q$. □

The expected value of $\hat{m}_Q$ in Algorithm 2 can be computed exactly by using the method in [13], upon observing that m_i in Algorithm 2 is the highest frequency for a multinomial sample with size n_Q and w bins. From the properties of m_i, m_Q can be claimed to be much less than n_Q when n_Q is large. This is demonstrated in Figure 1 which shows $\mathbb{E}(\hat{m}_Q)$ computed by using the recursion in [14].

[1]Strictly speaking, (3) is an upper bound, with equality holding if for all $j_{1:d}$ satisfying the condition in (3), there is an $x \in \mathcal{X}$ such that $j^x_{1:d} = j_{1:d}$.

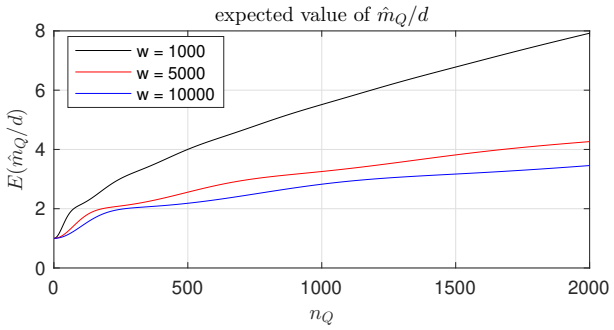

Fig. 1: $E(m_i) = E(\hat{m}_Q/d)$ vs query size n_Q, where m_i is given in Algorithm 2. Observe the sub-linear increase.

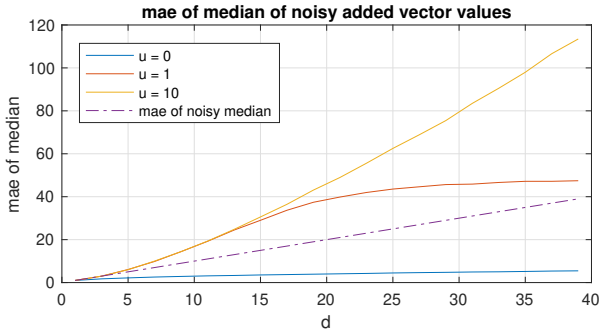

Fig. 2: A comparison between the mean absolute values of noise added median of the cell values (median perturbation) and the median of noisy cell values (cell perturbation).

2) Cell perturbation: While m_Q can be significantly smaller than n_Q, by Proposition 1, when $n_Q \ge d$, the parameter of the Laplace noise is d/ϵ at best.

An effective alternative to the median perturbation approach is observed in [6]. This approach is based on adding noise to the C's cells once and for all. Observe that there are only d cells in C that are affected by a presence/absence of an item in the stream. Namely, construct the noisy sketch

$$\tilde{C}(i,j) = C(i,j) + v_{i,j}, \quad v_{i,j} \overset{\text{i.i.d.}}{\sim} \text{Laplace}(d/\epsilon). \quad (4)$$

and answer all the queries by using the noisy sketch $\tilde{C}$. That is, for all the elements $x \in Q$, we return

$$\hat{f}_x = \text{median}\{\tilde{C}(1, h_1(x))s_1(x), \ldots, \tilde{C}(d, h_d(x))s_d(x)\} \quad (5)$$

In $\tilde{C}$, each of those sensitive cells are corrupted by a Laplace(d/ϵ) noise, which preserves ϵ/d privacy. By the composition theorem, the privacy level of constructing $\tilde{C}$ as such is ϵ. Moreover, by the post-processing theorem for DP, returning any number of query responses calculated from $\tilde{C}$ as in (5), which is a deterministic operation given $\tilde{C}$, is also ϵ-DP. In fact, it suffices to add noise to the cells that are relevant to the queries in Q. $C(i,j)$ is added noise if there is an $x \in Q$ such that $h_i(x) = j$. While this is a straightforward observation, it plays an important role in case of intermittent queries.

3) A comparison: A natural question is which method is better over the other one; the median perturbation in (2), or cell perturbation in (4) followed by (5)? Some probabilistic error bounds are available for both approaches; see Theorem 6 regarding cell perturbation and the theorems in Appendix C for median perturbation. Since those bounds only enable a qualitative elaboration, we instead resort to a numerical comparison and show that one method is not uniformly better than the other, and the outcome of the comparison depends on d, m_Q, and the variability among the true frequency values in the d cells to be used for the median calculation.

Figure 2 compares both methods for a hypothetical query set Q of size $n_Q \geq d$ (so that it is not too small) and assumes the best scenario $m_Q = d$ for the median perturbation method. We made this assumption so that the same amount of noise, drawn from Laplace(d/ϵ), is added to the median in the output perturbation method and to each of the cells in the cell perturbation method. Under this setting, we compare the performances of $\hat{f}_x$ in (2) and $\hat{f}_x$ in (5), when both provide ϵ-DP. The non-noisy values that are subject to the median calculation are designed as $(0, \ldots, 0, u, \ldots, u)$ where there are $(d+1)/2$ zeroes and $(d-1)/2$ u's, so that the median is always 0 and u resembles the amount of variation among the cell values.

The plots suggest that one method is not uniformly better than the other; and the first method can be more advantageous when the variation is higher. However, note that the assumption $m_Q = d$ may be a big favor for the first method and may not be even close to the truth when n_Q is large; see Figure 2.

While the approach of adding noise to C once and for all, or shortly the cell perturbation approach, is quite efficient when multiple queries are made at the same time, it is not tailored for a dynamic system where different query sets have to be handled at different times. This latter challenge, by which this work is motivated, is tackled in the next section.

IV. Queries at different times

Suppose we have a stream of data and query sets are made at different times, while the sketch is still being updated by the stream. Suppose query sets $Q_1, Q_2, \ldots$ are made at times $t_1 < t_2 < \ldots$. Crucially, new elements from the stream may arrive between successive query times. That is why the sketch and/or the noisy outputs of the previous time are not up-to-date. Here, we will analyze two classes of methods; *use-and-forget* and *use-and-keep*.

A. Use-and-forget methods

We first describe two methods, in both of which the sketch is carried on exactly and fresh noise is added to the medians or the cells themselves at each time t_k.

1) Median perturbation: The first method is based on median perturbation. As the total number of times a query is made is unknown beforehand, we can preserve ϵ-DP by preserving ϵ_k-DP for query time t_k, where ϵ_k is the k'th member of a geometric series,

$$\epsilon_k = \epsilon(1-\eta)\eta^{k-1}, \quad k = 1, 2, \ldots. \tag{6}$$

for some $\eta < 1$. Therefore, one way to preserve ϵ_k-DP privacy for the k'th set of queries, Q_k is made at time t_k. However, this scheme is hardly useful since the geometric increase in the amount of noise will soon destroy the accuracy in the responses.

2) Cell perturbation: An alternative method can be built up on an extension of the method for adding noise to the cells themselves, which allows a noisy cell value to be used more than one queries made at the same time. Since there are d sensitive cells, each has to be protected with ϵ/d-DP. For this, we need to keep track on the number of times each cell is used in median calculation. Let that information be kept in a $d \times w$ matrix U. Specifically, assume that x is queried and

$$(1, j_1^x), (2, j_2^x), \ldots, (d, j_d^x)$$

are the indices of the cells of the sketch table x is hashed. The numbers of times the those cells have been used is given by

$$U(1, j_1^x), U(2, j_2^x), \ldots, U(d, j_d^x).$$

Then, the algorithm uses noisy cell values $\tilde{C}(i, j_i^x) = C(i, j_i^x) + V_i$, where $V_i \overset{\text{i.i.d.}}{\sim}$ Laplace(d/ϵ_i') is adjusted to preserve ϵ_i'-DP, which be chosen as

$$\epsilon_i' = \frac{\epsilon}{d}(1-\eta)\eta^{U(i, j_i^x)}, \quad i = 1, \ldots, d.$$

The noisy counts are then used to produce the final answer as

$$\text{median}\{\tilde{C}(1, j_1^x)s_1(x), \ldots, \tilde{C}(d, j_d^x)s_d(x)\}.$$

Again, the reason for division by d is due to the existence of d sensitive cells. This algorithm guarantees that at the k'th use of the cell, we provide ϵ_k-DP. In total, each sensitive cell's information is protected with $\sum_k \epsilon_k = \epsilon/d$-DP, yielding ϵ-DP in total.

This approach can be implemented as in Algorithm 3. Note that any noise added to the cells of C is thrown away after being used in the median calculation. When a next set of queries is received at a future time, a fresh noise has to be added to the relevant cells. Because of that, we call this approach the use-and-forget approach.

B. Use-and-keep methods

The use-and-forget approach is designed to accommodate the requirement that the exact sketch be kept without corruption; observe that the non-noisy sketch table C is carried on in Algorithm 3. Since each noisy value leaks information about the true value, the algorithm has to apply a noise schedule with a geometrically increasing variance, from which it will eventually suffer.

When the sketch table need not be kept as exact, an alternative scheme is possible with superior statistical properties. This alternative scheme is based on the observation that the different item between X and X' can appear only in one of the intervals $(0, t_1], (t_1, t_2], \ldots$, where we recall that t_i is the time of the query set Q_i is made. Before explaining our new scheme, we will present a lemma on the DP of a very simple counting algorithm; a proof is given in Appendix A.

Algorithm 3: Use-and-forget DP Count Sketch

{Initialization of the table}
for $i = 1, \dots, d$ **do**
  for $j = 1, \dots, w$ **do**
    Set $C(i,j) \leftarrow 0$ and $U(i,j) \leftarrow 0$.
{Events start}
repeat
  if *stream element* x *arrives* **then**
    for $i = 1, \dots, d$ **do**
      $C(i, h_i(x)) \leftarrow C(i, h_i(x)) + s_i(x)$.
  else if *a set* Q *of queries are made* **then**
    Set $E = \emptyset$.
    for $x \in Q$ **do**
      Set $S \leftarrow \emptyset$.
      for $i = 1, \dots, w$ **do**
        Set $j = h_i(x)$.
        if $(i,j) \notin E$ **then**

$$\epsilon' = \frac{\epsilon}{d}(1-\eta)\eta^{U(i,j)}$$
$$v \sim \text{Laplace}(1/\epsilon')$$
$$\tilde{C}(i,j) = C(i,j) + v,$$
$$U(i,j) \leftarrow U(i,j) + 1$$
$$E \leftarrow E \cup \{(i,j)\}$$

        Append S with $\{\tilde{C}(i,j)s_i(x)\}$.
      return $\hat{f}_x = \textit{median}(S)$.
until *end of events*;

Algorithm 4: Use-and-keep DP count sketch

{Initialization of the table}
for $i = 1, \dots, d$ **do**
  for $j = 1, \dots, w$ **do**
    Set $C(i,j) \leftarrow 0$.
{Events start}
repeat
  if *stream element* x *arrives* **then**
    for $i = 1, \dots, d$ **do**
      $C(i, h_i(x)) \leftarrow C(i, h_i(x)) + s_i(x)$.
  else if *a set* Q *of queries are made* **then**
    Initialize $E = \emptyset$
    for $x \in Q$ **do**
      Set $S = \emptyset$.
      for $i = 1, \dots, d$ **do**
        Set $j = h_i(x)$.
        if $(i,j) \notin E$ **then**

$$v \sim \text{Laplace}(d/\epsilon)$$
$$C(i,j) \leftarrow C(i,j) + v$$
$$E \leftarrow E \cup \{(i,j)\}$$

        Append S with $\{C(i,j)s_i(x)\}$.
      return $\hat{f}_x = \textit{median}(S)$.
until *end of events*;

Lemma 1. *Suppose we have a simple counter, c, that counts the number of occurrences of a certain value, x, in a stream. Let the times of queries about c be $t_1, t_2, \dots$. Also, assume a noisy counter, $\tilde{c}$, which is incremented by 1 on every occurrence of x, just like c, however with the difference that at the time t_i of each query it is updated as*

$$\tilde{c} \leftarrow \tilde{c} + v_i, \quad v_i \overset{i.i.d.}{\sim} \text{Laplace}(1/\epsilon).$$

A mechanism that returns $\tilde{c}$ at each query time $t_1, t_2, \dots$ immediately after the noise adding step is ϵ-DP.

The theorem can easily be applied to the cells of the sketch table. Namely, one can keep the noise in the cells and carry on sketching the stream with the noisy sketch table. When a cell that has been used before (and hence it is noisy) is to be used for the next time, an independent Laplace(d/ϵ) noise is added to the current value of the cell. The new noisy value goes in the calculations needed to estimate the count, and it is kept as the current value of the cell.

With the new scheme, the total noise on each cell is the sum of k independent Laplace(d/ϵ) noises, where k is the number of times the value of that cell is used. Therefore, the accumulation of noise on a cell value is much less severe than the noise in Algorithm 3 whose variance geometrically increases in k. With this motivation, we present Algorithm 4. Furthermore, Algorithm 4 does not need to know how many times a cell is used before, therefore does not need to keep a matrix such as U in Algorithm 3. At each query x, each cell which x is hashed is used after its value is updated with the addition of an independent Laplace(d/ϵ) noise.

Differential privacy property of Algorithm 4 is indicated in the following theorem.

Theorem 4. *Algorithm 4 provides ϵ-differential privacy.*

Proof. First, note that, for queries made at the same time, a cell is added noise at most once. Secondly, for queries made at multiple times, we update the noisy values by merely adding an independent noise to the noisy count, therefore imitating the counting process in Lemma 1. This ensures that the value of each used cell in the sketch table $C(i,j)$ is protected with ϵ/d privacy throughout the whole process. Next, observe that there are only d sensitive cells: Letting X and X' be two neighboring data sets, and the differing element be x, the indices of C where X and X' differ after using the same hash functions are

$$I_x = \{(1, h_1(x)), \dots, (d, h_d(x))\}.$$

Therefore, the total privacy is $d\epsilon/d = \epsilon$, by the composition theorem, Theorem 2. □

1) Error analysis for the use-and-keep algorithm: We can quantify the error in the responses of Algorithm 4 with the following theorem. A proof is given in the Appendix.

Theorem 5. *Assume that Algorithm 4 is run with d rows and w columns and the required privacy level is ϵ. Suppose that a query is made for an element x, and for $1 \leq i \leq d$, the cell that it is hashed to in the i'th row has been used $u_i - 1$ times prior to the query. Then, for any $\kappa^2 > 2\max_{i=1,\dots,d}\left(\frac{||f||_2^2}{w} + \frac{d^2}{\epsilon^2}u_i\right)$, we have*

$$\mathbb{P}(|\hat{f}_x - f_x| > \kappa) \leq e^{-\frac{d}{2}(1-2\lambda)}[2(1-\lambda)]^{d/2}$$

where $\lambda = \frac{||f||_2^2}{\kappa^2 w} + \frac{d}{\kappa^2\epsilon^2}\sum_{i=1}^{d} u_i$, and $||f||_2^2 = \sum_x f_x^2$.

Theorem 5 has two important implications.

- Fix x, the element to be queried. Fix a small probability ρ and let λ_ρ be such that $e^{-\frac{d}{2}(1-2\lambda^*)}[2(1-\lambda^*)]^{d/2} = \rho$. Then, the absolute error which is exceeded with probability ρ is $\left(\frac{||f||_2^2}{\lambda_\rho w} + \frac{d}{\lambda_\rho\epsilon^2}\sum_{i=1}^{d} u_i\right)^{1/2}$. This grows sublinearly with $\sum_{i=1}^{d} u_i$, the sum of the total number of uses of the cells that x is hashed to. This suggests that, if x is queried as frequently as its occurrence in the data stream (and similarly for the other elements that are also hashed in the same cells as x), we expect the relative error $\hat{f}_x$ to decrease in time.
- The error bound in the theorem indicates the two sided effect, in terms of performance, of the number of rows d in the sketch table. Observe that the error probability may not be monotonic in d; instead, increasing d up to a certain value may improve the error bound until worsening it beyond that value. This can be explained as follows: while a larger d helps the accuracy of the median, it also corrupts the cell values with more noise. Therefore, the optimum value d in terms of performance is in general somewhere in the middle.

V. Related Work

The related research in the literature can be grouped into three categories: (i) those that modify the standard count-based sketches to make them privacy-preserving, (ii) those that combine count based sketches and privacy-preserving mechanisms as separate steps of a master algorithm, (iii) those that argue that the inherent noise of the count-based sketches provides privacy under certain special conditions. These works are generally based on using the Laplace mechanism or its variations. As an example of (i), Count Sketch and Count-Min Sketch are used for the succinct representation of user data in several different settings (such as recommendation systems, user location prediction, etc.) and Laplace noise is used to enhance the privacy of estimates obtained from these sketches [6]. There exist examples for (ii) which utilize Count (or Count-Min) Sketches and privacy-preserving mechanisms as distinct sub-procedures of a master algorithm, without modifying these sketches [15]. Similarly, [5] use the Geometric mechanism (a discrete analog of the Laplace mechanism) to perturb the inputs that are later passed into Count Sketch (or alternatively, other summarisation methods which are based on sampling and filtering) . Hence, they do not incorporate this mechanism directly into the sketch; they use it in the preliminary steps of their main algorithms. Aside from these, [16] and [17], which are in category (iii), the Count Sketch is argued to be inherently providing DP in a special setting, where the inputs of the sketch are the gradient updates of an optimization problem. Under some strict assumptions given in those studies, such as the inputs of sketches being Gaussian distribution and their norms being bounded with high probability; Count Sketch satisfies DP by itself. But the authors still use output perturbation (by either adding Laplace noise to gradient updates or clipping the log-likelihood function of the optimization problem) when the inherent noise of sketch is not sufficient to provide ϵ-DP. Also, the assumptions that are used in these studies are not applicable when the input is frequency data which is discrete, nonnegative, and in the case of data streams, where the distribution is generally observed to be a power-law distribution.

All of these existing approaches mentioned above can be thought of as addressing the problem of answering one or multiple queries *at a single time* in a privacy preserving way. Our primary focus is protecting privacy when the data in question are in the form of a data stream that is formed by sensitive information from individuals, and queries are made dynamically and intermittently. The concept of providing privacy under continual observation was originally laid out in [18] where several algorithms were proposed for that objective as well. Although counters, or, more generally, statistics monotonic in time, are considered in [18], their methods can be adopted for the cells of the Count Sketch. However, those methods require additional memory.

VI. Experiments

In this section, we focus on the performance of Algorithm 4. We generated a data stream of size 2^{27} where each item is a random draw from the Zipfian distribution with domain $\{1, \dots, 2^{29}\}$ and parameter α, so that $p(X = k) \propto k^{-\alpha}$ for $k = 1, \dots, 2^{29}$. We tested Algorithm 4 under several scenarios constructed by the combination of several parameters. These parameters are as follows:

- privacy level $\epsilon \in \{0.01, 0.05, 1\}$;
- $\alpha = 1$ of the Zipfian;
- the number of rows $d \in \{3, 9, 15\}$;
- the number of columns $w = 5000$;
- query set size $n_Q \in \{1, 10^3, 10^4, 10^5, 10^6\}$;
- user type parameter $u \in \{50, 5000, 500000\}$ which denotes that the query items are coming from the top u most frequent items.

For each combination of $(\epsilon, \alpha, d, w, n_Q, u)$ with the components taking values in their ranges stated above, we simulated the dynamic system in Algorithm 4. In each simulation, the data stream is generated gradually with the arrival of an element with regular time intervals. Along with the arrivals, queries are scheduled periodically at evenly spaced times. The

periods for queries are arranged in such a way that the total number of queried elements is the same and equal to 10^6 for all the scenarios. That is, the period for queries when $n_Q = 10$ is 10 times the period for queries when $n_Q = 1$. This is ensured for fair comparison of scenarios in terms of accuracy. For each scenario, the queries start after 5×10^5 items arrive. Finally, all the scenarios are simulated 20 times with the same data stream but independent random seeds to generate the hash functions, query sets, and the Laplace noises. The average errors over those independent simulations are reported. Figure 3 shows the plots for the cumulative mean relative error vs time for combinations of ϵ, n_Q. For each combination, error plots for different values of d are superimposed. All the other parameters are averaged out.

Comparing the plots in Figure 3 along each parameter is informative: First, as expected, the error increases with decreasing ϵ. Second, we have smaller errors as n_Q increases. This is because answering a set queries at the same time is more beneficial than answering parts of them at separate times, since the former has the advantage of using the same cell noise for more queries.

A further deduction from Figure 3 is that the best value of d, among the ones compared, decreases as the scenario becomes more challenging. Observe, e.g., the last column, where $\epsilon = 0.01$. As n_Q decreases, which necessitates more frequent noise adding, the value of d that gives the minimum error also decreases: it changes from $d = 15$ to $d = 9$ (at $n_Q = 10000$) and then to $d = 3$ as n_Q further decreases. This may be explained as follows: Smaller n_Q implies more frequent use of a single cell. Recall that we add a Laplace(d/ϵ) noise to a cell value each time the cell is required, making the cells noisier. There is a certain value of d beyond which the effect of taking the median over d rows is overwhelmed by the error due to the amount of noise added to the cells. This value of d is smaller when the cells are noisier (n_Q is smaller). This transition of the best d towards small values occurs earlier for smaller ϵ, since smaller ϵ means a more noisy sketch table.

Figure 4 shows plots for cumulative mean relative errors vs time for combinations of ϵ, u and with plots for different values of d superimposed. All the other parameters are averaged out. Observe that the relative error is larger as the queries are made from a larger pool of top elements. This behavior is also typical of the regular Count Sketch. Besides that, similar conclusions can be drawn about the behavior of the error curves as d changes. Once again, smaller values d is preferred as the cells get more noisy.

In both Figure 3 and Figure 4, we observe the cumulative relative error (left) grows sub-linearly. This supports our claim that in Section IV-B1 stemming from Theorem 5 that the absolute error is expected to grow if queries are made as frequently as the arrival of new elements to the stream.

VII. Conclusion

In this paper, we proposed differentially private versions of the count sketch for frequency estimation. We both discussed median perturbation and cell perturbation methods. While in

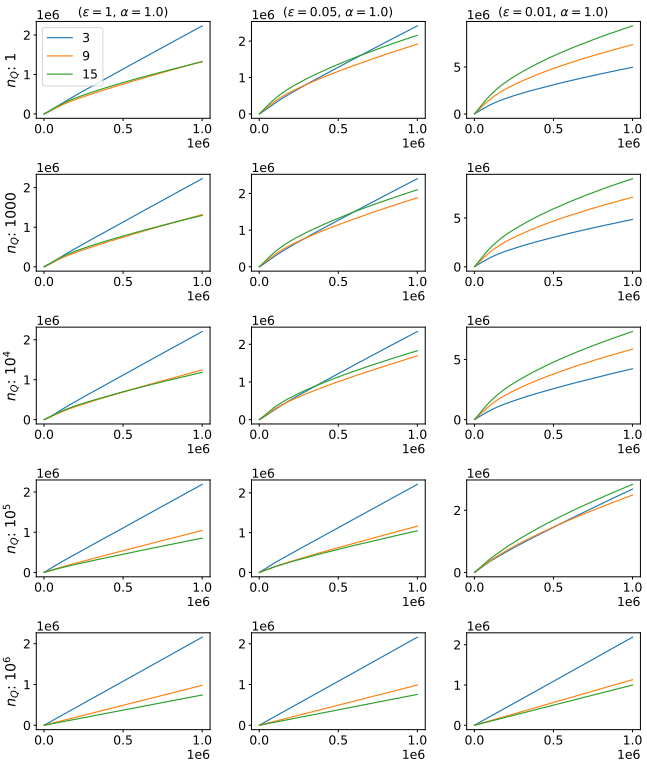

Fig. 3: Cumulative mean relative error of Algorithm 4 for different combinations of ϵ, n_Q and d.

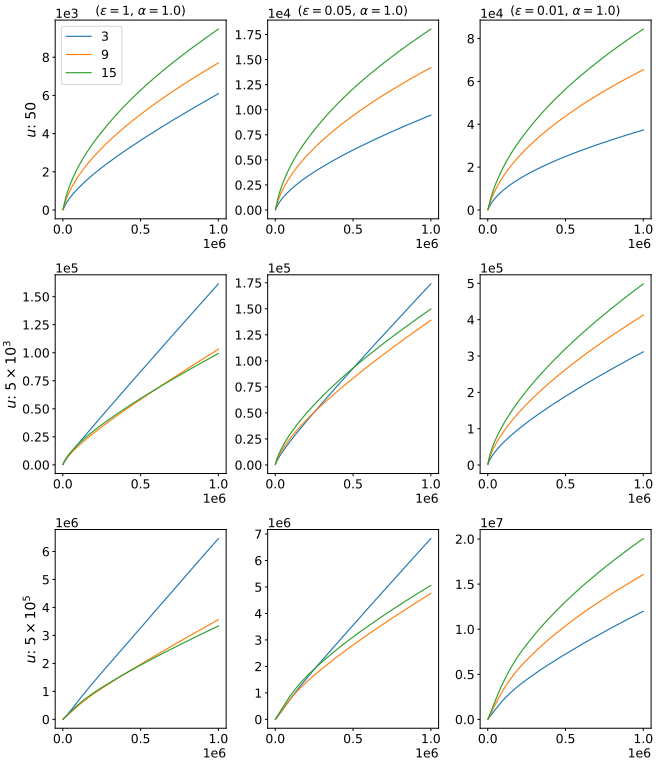

Fig. 4: Cumulative mean relative error of Algorithm 4 for different combinations of ϵ, u and d.

the static case, where all queries are made at the same time, it is not certain which method will prevail, for the dynamic case we propose using the cell perturbation technique as it is able to produce less noisy estimates. The ultimate algorithm proposed for the dynamic case was Algorithm 4, where the noise is used and kept in the cell value so that the DP noise variance increases only linearly, as opposed to geometrically which would happen with a naive implementation of the composition theorem for differential privacy.

Possible extensions to the proposed approach are as follows:

a) Subsampling: It is possible to increase the privacy level of the algorithms by deciding to process each element in the stream with a certain probability $0 < \rho < 1$ [19]. This increases the privacy level by reducing ϵ to roughly an order of $\rho\epsilon$ for small ρ, see [19] for the exact expression. The final estimates are then to be rescaled by $1/\rho$ to preserve unbiasedness. This comes, of course, with the expense of reducing the accuracy by means of multiplying the variance of the estimates by $1/\rho^2$.

b) Algorithms for Count-Min Sketch: Just like Count Sketch, Count-Min Sketch can be modified in a similar fashion to provide privacy. Note that like the Count Sketch, the sensitivity of each cell and the minimum of those cells is 1. In our implementations, we used only Count Sketch, since it provides an unbiased estimator of the true frequency values, unlike Count-Min Sketch which overestimates the true counts.

c) Pan-privacy: One direction for future research can be to modify the use-and-keep approach so as to satisfy pan-privacy [3]. It is not difficult to satisfy pan-privacy against a single intrusion (into the state of the algorithm, which is the sketch C) followed by a set of queries made at a single time. This is achieved by simply starting C with $C(i,j) \overset{\text{i.i.d.}}{\sim} \text{Laplace}(d/\epsilon)$. However, providing pan-privacy against multiple intrusions seems difficult and needs more investigation.

d) Alternatives to the Laplace mechanism: More advanced techniques than the Laplace mechanism for preserving the same amount of privacy were offered in [20]. We remark that the choice for the probability distribution of the DP noise is not the main focus of our algorithm. Any noise adding technique is equally applicable and can substitute the Laplace mechanism as long as providing the same level of privacy. Moreover, light-tailed noise mechanisms such as the Gaussian mechanism can be considered instead of Laplace mechanism if one is willing to weaken the privacy requirements (and welcome $\delta > 0$).

REFERENCES

[1] C. Dwork, "Differential privacy," in *Automata, Languages and Programming*, M. Bugliesi, B. Preneel, V. Sassone, and I. Wegener, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 1–12.

[2] N. Mishra and M. Sandler, "Privacy via pseudorandom sketches," in *Proceedings of the Twenty-Fifth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, ser. PODS '06. New York, NY, USA: Association for Computing Machinery, 2006, pp. 143–152. [Online]. Available: https://doi.org/10.1145/1142351.1142373

[3] C. Dwork, M. Naor, T. Pitassi, G. Rothblum, and S. Yekhanin, "Pan-private streaming algorithms," in *Proceedings of The First Symposium on Innovations in Computer Science (ICS 2010)*. Tsinghua University Press, January 2010. [Online]. Available: https://www.microsoft.com/en-us/research/publication/pan-private-streaming-algorithms/

[4] D. Mir, S. Muthukrishnan, A. Nikolov, and R. N. Wright, "Pan-private algorithms via statistics on sketches," in *Proceedings of the Thirtieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, ser. PODS '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 37?48. [Online]. Available: https://doi.org/10.1145/1989284.1989290

[5] G. Cormode, C. Procopiuc, D. Srivastava, and T. T. L. Tran, "Differentially private summaries for sparse data," in *Proceedings of the 15th International Conference on Database Theory*, ser. ICDT '12. New York, NY, USA: ACM, 2012, pp. 299–311. [Online]. Available: http://doi.acm.org/10.1145/2274576.2274608

[6] L. Melis, G. Danezis, and E. D. Cristofaro, "Efficient private statistics with succinct sketches," in *23rd Annual Network and Distributed System Security Symposium, NDSS 2016, San Diego, California, USA, February 21-24, 2016*. The Internet Society, 2016. [Online]. Available: http://wp.internetsociety.org/ndss/wp-content/uploads/sites/25/2017/09/efficient-private-statistics-with-succinct-sketches.pdf

[7] H. Sparka, F. Tschorsch, and B. Scheuermann, "P2kmv: A privacy-preserving counting sketch for efficient and accurate set intersection cardinality estimations," Cryptology ePrint Archive, Report 2018/234, 2018. [Online]. Available: https://eprint.iacr.org/2018/234

[8] S. N. von Voigt and F. Tschorsch, "Rrtxfm: Probabilistic counting for differentially private statistics," in *IACR Cryptol. ePrint Arch.*, 2019.

[9] M. Charikar, K. Chen, and M. Farach-Colton, "Finding frequent items in data streams," in *Proceedings of the 29th International Colloquium on Automata, Languages and Programming*, ser. ICALP '02. Berlin, Heidelberg: Springer-Verlag, 2002, pp. 693–703.

[10] G. Cormode and S. Muthukrishnan, "An improved data stream summary: the count-min sketch and its applications," *Journal of Algorithms*, vol. 55, no. 1, pp. 58 – 75, 2005. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0196677403001913

[11] C. Dwork, "Differential privacy: A survey of results," in *International Conference on Theory and Applications of Models of Computation*. Springer, 2008, pp. 1–19.

[12] M. Charikar, K. Chen, and M. Farach-Colton, "Finding frequent items in data streams," *Theoretical Computer Science*, vol. 312, no. 1, pp. 3 – 15, 2004, automata, Languages and Programming. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0304397503004006

[13] P. R. Freeman, "Algorithm as 145: Exact distribution of the largest multinomial frequency," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, no. 3, pp. 333–336, 1979. [Online]. Available: http://www.jstor.org/stable/2347220

[14] M. V. Ramakrishna, "An exact probability model for finite hash tables," in *Proceedings. Fourth International Conference on Data Engineering*, Feb 1988, pp. 362–368.

[15] A. Monreale, W. H. Wang, F. Pratesi, S. Rinzivillo, D. Pedreschi, G. Andrienko, and N. Andrienko, "Privacy-preserving distributed movement data aggregation," in *Geographic Information Science at the Heart of Europe*. Springer, 2013, pp. 225–245.

[16] R. Balu and T. Furon, "Differentially private matrix factorization using sketching techniques," in *Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security*. ACM, 2016, pp. 57–62.

[17] T. Li, Z. Liu, V. Sekar, and V. Smith, "Privacy for free: Communication-efficient learning with differential privacy using sketches," *arXiv preprint arXiv:1911.00972*, 2019.

[18] C. Dwork, M. Naor, T. Pitassi, and G. N. Rothblum, "Differential privacy under continual observation," in *Proceedings of the Forty-Second ACM Symposium on Theory of Computing*, ser. STOC '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 715?724. [Online]. Available: https://doi.org/10.1145/1806689.1806787

[19] B. Balle, G. Barthe, and M. Gaboardi, "Privacy amplification by subsampling: Tight analyses via couplings and divergences," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, ser. NIPS'18. Red Hook, NY, USA: Curran Associates Inc., 2018, p. 6280?6290.

[20] G. Cormode, T. Kulkarni, and D. Srivastava, "Constrained differential privacy for count data," *ArXiv*, vol. abs/1710.00608, 2017.

APPENDIX

A. Proof of Lemma 1

Proof. (Lemma 1) Given a stream, let c_k^X be the true count for the k'th query at time t_k. Also, let $\tilde{C}_0^X = 0$ and $\tilde{C}_k^X$ be the reported noisy count for the k'th query at time t_k (capital letter is used to emphasize the randomness in $\tilde{C}_k^X$). Note that $\tilde{C}_k^X - \tilde{C}_{k-1}^X$, $k \geq 1$, are i.i.d. with $\tilde{C}_k^X - \tilde{C}_{k-1}^X - (c_k^X - c_{k-1}^X) \sim$ Laplace$(1/\epsilon)$. For any $n > 0$, the joint density of $\tilde{C}_0^X, \ldots, \tilde{C}_n^X$ at the values $\tilde{c}_0, \ldots, \tilde{c}_n$ is

$$p_X(\tilde{c}_0, \ldots, \tilde{c}_n) = \prod_{k=1}^{n} \frac{\epsilon}{2} \exp\{-\epsilon|(\tilde{c}_k - \tilde{c}_{k-1}) - (c_k^X - c_{k-1}^X)|\}$$

For neighbour X and X', $(c_k^X - c_{k-1}^X)$ and $(c_k^{X'} - c_{k-1}^{X'})$ differ by 1 for only one $k \geq 1$. For that k. We have

$$|(\tilde{c}_k - \tilde{c}_{k-1}) - (c_k^X - c_{k-1}^X)| - |(\tilde{c}_k - \tilde{c}_{k-1}) - (c_k^{X'} - c_{k-1}^{X'})| \leq 1$$

As a result, $e^{-\epsilon} \leq p_X(\tilde{c}_0, \ldots, \tilde{c}_n)/p_{X'}(\tilde{c}_0, \ldots, \tilde{c}_n) \leq e^{\epsilon}$. □

B. Proof of Theorem 5

Theorem 5 is based on a standard result on the median of probabilistically bounded random variables, which is restated here with some adaptation to our setting.

Lemma 2. *Let $X_1, \ldots, X_d$ be random variables satisfying*

$$\mathbb{P}(|X_i - \mu| > \kappa) < \lambda_i, \quad i = 1, \ldots, d$$

and let $\lambda = \sum_{i=1}^{d} \lambda_i/d$. If $\lambda < 1/2$, then,

$$\begin{aligned}\mathbb{P}(|\tilde{X} - \mu| > \kappa) &\leq e^{-\frac{d}{2}(1-2\lambda)}[2(1-\lambda)]^{d/2} \\ &\leq \exp\left\{-\frac{d}{8}\frac{(1-2\lambda)^2}{1-\lambda}\right\}.\end{aligned}$$

where $\tilde{X}$ is the median of $X_1, \ldots, X_d$.

Proof. Let $Y_i = \mathbb{I}(|X_i - \mu| < \kappa)$ for $i = 1, \ldots, d$, and $Y = \sum_i Y_i$. Then $\mathbb{E}(Y_i) > 1 - \lambda_i$ and $\mathbb{E}(Y) = d(1-\lambda)$. The event $|\tilde{X} - \mu| > \kappa$ implies that at least $d/2$ of $X_1, \ldots, X_d$ are outside $(\mu - \kappa, \mu + \kappa)$, which is equivalent to $Y < d/2$. Also,

$$\begin{aligned}\{Y < d/2\} &\Leftrightarrow \left\{Y < \mathbb{E}(Y)\left[1 - \left(1 - \frac{d}{2\mathbb{E}(Y)}\right)\right]\right\} \\ &\Rightarrow \left\{Y < \mathbb{E}(Y)\left[1 - \left(1 - \frac{d}{2d(1-\lambda)}\right)\right]\right\} \\ &\Leftrightarrow \{Y < \mathbb{E}(Y)\,(1-\delta)\}\end{aligned}$$

where $\delta = \frac{1-2\lambda}{2(1-\lambda)}$. Therefore, by the Chernoff bound, we have

$$\begin{aligned}\mathbb{P}(|\tilde{X} - \mu| > \kappa) &\leq \left(\frac{e^{-\delta}}{(1-\delta)^{1-\delta}}\right)^{E(Y)} \\ &\leq \left(\frac{e^{-\delta}}{(1-\delta)^{1-\delta}}\right)^{d(1-\lambda)} = e^{-\frac{d}{2}(1-2\lambda)}[2(1-\lambda)]^{d/2}\end{aligned}$$

where the first line is by Chernoff, the second line is due to the fact that $\mathbb{E}(Y) > d(1-\lambda)$ and the base of the exponentiation is bounded by 1. □

We can use Lemma 2 to prove Theorem 5 for the use-and-keep method for dynamic queries in Algorithm 4.

Proof. (Theorem 5) In Algorithm 4, if the d cells corresponding to the queried element x are used $u_1 - 1, \ldots, u_d - 1$ times prior to the query, they will have been used $u_1, \ldots, u_d$ just before the response. Therefore, the values in the related cells can be written as $C(i, j_i^x) = C_0(i, j_i^x) + V_i$ for $i = 1, \ldots, d$, where $C_0(i, j_i^x)$ is the cell value we would have if the regular count sketch were used and $V_i = \sum_{k=1}^{u_i} V_{i,k}$ where each $V_{i,j}$ are independent with $V_{i,j} \sim$ Laplace(d/ϵ). We have $\mathbb{E}(C(i, j_i^x)) = f_x$ and $\sigma_i^2 = \text{var}(C(i, j_i^x))$ is bounded by

$$\sigma_i^2 \leq \frac{||f||_2^2}{w} + \frac{u_i d^2}{\epsilon^2}.$$

where the first term is due to hashing. For $\kappa^2 > 2\max_i \sigma_i^2$, using the Tchebyshev's bound, we have

$$\mathbb{P}(|\hat{C}_i - f_x| > \kappa) \leq \frac{\sigma_i^2}{\kappa^2}, \quad i = 1, \ldots, d.$$

By Lemma 2, an upper bound on the error of the median $\hat{f}_x^k = \text{median}(C(1, j_1^x), \ldots, C(1, j_d^x))$ is given by

$$\mathbb{P}(|\hat{f}_x - f_x| > \kappa) \leq e^{-\frac{d}{2}(1-2\lambda)}[2(1-\lambda)]^{d/2}$$

where $\lambda = \frac{1}{d}\sum_{i=1}^{d} \frac{\sigma_i^2}{\kappa^2} = \frac{||f||_2^2}{w\kappa^2} + \frac{d}{\epsilon^2\kappa^2}\sum_{i=1}^{d} u_i$ as claimed. □

C. Error bounds for output perturbation

Let f_x^c denote the response which is returned by standard count sketch for element x, and $\hat{f}_x^\epsilon$ be the value obtained by adding a Laplace noise to f_x^c to preserve ϵ privacy. Also, define $||f||_2^2 := \sum_x f_x^2$. The first result on the error is a bound for the probability of a fixed error.

Theorem 6. *Let $\kappa > 2||f||_2/w$ be a constant. Then, we have $\mathbb{P}(|\hat{f}_x^\epsilon - f_x| > \kappa) < \lambda$ where, with $\lambda^* = ||f_{-x}||_2/(w\kappa^*)$,*

$$\lambda = \min_{\frac{2||f||_2}{w} < \kappa^* < \kappa} e^{-\frac{d}{2}[(1-2\lambda^*) - \ln(2(1-\lambda^*))]} + e^{-\epsilon(\kappa - \kappa^*)}$$

Proof. For any $2||f_{-x}||_2/w < \kappa^* < \kappa$, the simple triangular inequality for probability statements can be applied as

$$\mathbb{P}(|\hat{f}_x^\epsilon - f_x| > \kappa) < \mathbb{P}(|\hat{f}_x - f_x| > \kappa - \kappa^*) + \mathbb{P}(|\hat{f}_x^\epsilon - \hat{f}_x| > \kappa^*).$$

While the first probability can be bounded using Lemma 2, the second one can be calculated exactly using the cdf of the Laplace distribution. This inequality holds for any $\kappa > \kappa^* > 2||f_{-x}||_2/w$, hence the minimum. □

Now, let us fix λ and examine how the amount of error varies with respect to d, with probability $1 - \lambda$.

Corollary 1. *For a noisy median response that satisfies ϵ_0-DP and $\lambda > (e/2)^{-d/2}$, we have $\mathbb{P}(|\hat{f}_x^\epsilon - f_x| > \kappa) < \lambda$ where*

$$\kappa = \max_{(e/2)^{-d/2} < \lambda^* < \lambda} \frac{||f||_2}{\sqrt{w\lambda_0}} - \frac{1}{\epsilon_0 \ln(\lambda - \lambda^*)}.$$

where λ_0 is such that $e^{-\frac{d}{2}[(1-2\lambda_0) - \ln(2(1-\lambda_0))} = \lambda^*$.