The Sparse Vector Technique, Revisited

Haim Kaplan[*]   HAIMK@TAU.AC.IL
Tel Aviv University and Google Research

Yishay Mansour[†]   MANSOUR.YISHAY@GMAIL.COM
Tel Aviv University and Google Research

Uri Stemmer[‡]   U@URI.CO.IL
Ben-Gurion University and Google Research

Editors: Mikhail Belkin and Samory Kpotufe

Abstract

We revisit one of the most basic and widely applicable techniques in the literature of differential privacy – the *sparse vector technique* [Dwork et al., STOC 2009]. This simple algorithm privately tests whether the value of a given query on a database is close to what we expect it to be. It allows to ask an unbounded number of queries as long as the answer is close to what we expect, and halts following the first query for which this is not the case.

We suggest an alternative, equally simple, algorithm that can continue testing queries as long as any single individual does not contribute to the answer of too many queries whose answer deviates substantially form what we expect. Our analysis is subtle and some of its ingredients may be more widely applicable. In some cases our new algorithm allows to privately extract much more information from the database than the original.

We demonstrate this by applying our algorithm to the *shifting heavy-hitters* problem: On every time step, each of n users gets a new input, and the task is to privately identify all the *current* heavy-hitters. That is, on time step i, the goal is to identify all data elements x such that many of the users have x as their current input. We present an algorithm for this problem with improved error guarantees over what can be obtained using existing techniques. Specifically, the error of our algorithm depends on the maximal number of times that a *single user* holds a heavy-hitter as input, rather than the total number of times in which a heavy-hitter *exists*.

Keywords: Differential privacy, sparse vector, heavy hitters.

1. Introduction

Differential privacy (Dwork et al., 2006) is a mathematical definition for privacy, that aims to enable statistical analyses of databases while providing strong guarantees that individual-level information does not leak. More specifically, consider a database containing data pertaining to individuals, and suppose that we have some data analysis procedure that we would like to apply to this database. We say that this procedure preserves *differential privacy* if no individual's data has a significant effect on the distribution of the outcome of the procedure. Intuitively, this guarantees that whatever is learned about an individual from the outcome of the computation could also be learned with her data arbitrarily modified (or without her data). Formally,

Definition 1 (Dwork et al. (2006)). *A randomized algorithm $\mathcal{A}$ is (ε, δ)-differentially private if for every two databases S, S' that differ on one row, and every set of outcomes F, we have $\Pr[\mathcal{A}(S) \in F] \leq e^{\varepsilon} \cdot \Pr[\mathcal{A}(S') \in F] + \delta$.*

Over the last decade, we have witnessed an explosion of research on differential privacy, and by now it is largely accepted as a gold-standard for privacy preserving data analysis. One of the main properties that helped propel differential privacy to its current state is that differential privacy is closed under composition (Dwork et al., 2010). This important property allows the design of differentially private algorithms to be modular: One can design complex differentially private algorithms by combining several (existing) differentially private algorithms and using them as subroutines. However, the downside with composition is that it degrades the privacy guarantees. Specifically, composing k differentially private algorithms degrades the privacy guarantees by (roughly) a factor of $\sqrt{k}$. As a result, there are cases in which directly applying composition theorems is not the way to obtain the best results, and different techniques are needed. In this work we revisit one of the most basic (and widely applicable) of these techniques, known as the *sparse vector technique* (Dwork et al., 2009).

The sparse vector technique. Consider a large number of *counting queries* $f_1, \ldots, f_m : X \to \{0, 1\}$, which are given one by one to a data curator holding a database $S \in X^n$. In every round, we would like to receive, in a differentially private manner, an approximation to $f_i(S) \triangleq \sum_{x \in S} f_i(x)$. (A *counting query* asks for the number of records in S that satisfy a given predicate, e.g., the number of individuals with blue eyes and diabetes.) This task can be solved by answering each query in a differentially private manner (by adding an appropriately sampled noise), and using composition theorems to argue about the overall privacy guarantees. Due to the costs incurred by composition, this approach requires a database of size $n \gtrsim \sqrt{m}$ in order to answer m such counting queries. In many cases, however, we might only be interested in obtaining answers for the queries f_i whose answers are "meaningful". More specifically, we have some threshold t in mind, and we only care to receive answers to queries s.t. $f_i(S) \geq t$. Dwork et al. (2009) presented a simple (and elegant) tool for such a scenario – algorithm `AboveThreshold`. This algorithm is capable of identifying (and answering) the *first* meaningful query from a stream of queries, while privately ignoring all the (non-meaningful) queries that came before it. The key observation of Dwork et al. is that privately ignoring the non-meaningful queries comes (essentially) for free. Informally, algorithm `AboveThreshold` can be described as follows.

Algorithm `AboveThreshold` (informal)

Input: Database S, threshold t, and a stream of counting queries $f_i : X \to \{0, 1\}$.

1. Let $\hat{t} \leftarrow t + \text{Noise}$.
2. In each round i, when receiving a query f_i, do the following:
    (a) Let $\hat{f}_i \leftarrow f_i(S) + \text{Noise}$.
    (b) If $\hat{f}_i \geq \hat{t}$, then output $\top$ and halt.
    (c) Otherwise, output $\bot$ and proceed to the next iteration.

Once algorithm `AboveThreshold` identifies a "meaningful" query (and halts), estimating the value of this "meaningful" query can be done using standard differentially private noise addition

techniques. In addition, by re-executing the algorithm after every such meaningful query (and using composition theorems to argue about the overall privacy guarantees), we can answer c meaningful queries (out of a stream of m queries) using a database of size $n = \tilde{O}\left(\sqrt{c} \cdot \log m\right)$. When the number of meaningful queries c is much smaller than the total number of queries m (hence the name *sparse vector*), this technique achieves a dramatic improvement compared to what we get when answering *all* queries. The sparse vector technique has proven itself to be one of the most important tools in the literature of differential privacy, with many applications.[1]

1.1. Our contributions

As we mentioned, algorithm `AboveThreshold` of Dwork et al. can be thought of as a technique for bypassing composition barriers (because instead of answering non-meaningful queries, paying in composition, we "privately ignore" them for free). However, in order to answer c meaningful queries, we still use composition theorems when we re-execute the algorithm after every meaningful query. Do we lose something with this approach?

We present a variant of the algorithm `AboveThreshold` in which this transition between one meaningful query to the next is done "more smoothly", without re-executing the algorithm. While we cannot avoid the composition cost altogether, we account for it in a more fine-tuned way, which, in some cases, leads to a significant improvement. Recall that using algorithm `AboveThreshold` we can answer c meaningful queries (out of a stream of m queries) using a database of size $n = \tilde{O}\left(\sqrt{c} \cdot \log m\right)$. Loosely speaking, with our algorithm, the factor $\sqrt{c}$ in the requirement on the database size is replaced with $\sqrt{k^*}$, where k^* is the maximum number of times in which a *single* record in the database *contributes* to a "meaningful" answer. That is,

$$k^* = \max_{x \in S} \left|\{i : f_i(S) \geq t \text{ and } f_i(x) = 1\}\right|.$$

Note that k^* is always at most $c = |\{i : f_i(S) \geq t\}|$, and that it can be significantly smaller than c.

1.1.1. OUR ALGORITHM

At its most basic form, our algorithm can be informally described as follows. We remark that this presentation is oversimplified, and does not capture the full generality of our algorithm (in particular, the connection to k^*, mentioned above, will only become clear later).

Algorithm `ThresholdMonitor` (simplified)

Input: Database S, threshold t, and a stream of counting queries $f_i : X \rightarrow \{0, 1\}$.

1. In each round i, when receiving a query f_i, do the following:
    (a) Let $\hat{f}_i \leftarrow f_i(S) + \text{Noise}$.
    (b) If $\hat{f}_i < t$, then output $a_i = \bot$.
    (c) Otherwise, output $a_i = \top$ and delete from S every x such that $f_i(x) = 1$.

---

1. See, e.g., Hardt and Rothblum (2010); Dwork et al. (2015); Nissim et al. (2016); Nissim and Stemmer (2018); Kaplan et al. (2020b,a); Hassidim et al. (2020); Bun et al. (2019); Stemmer (2016); Bassily et al. (2018); Cummings et al. (2019); Ligett et al. (2017); Barthe et al. (2016a,b); Steinke and Ullman (2016); Cummings et al. (2015); Ullman (2015); Nandi and Bassily (2020); Shokri and Shmatikov (2015); Hsu et al. (2013); Sajed and Sheffet (2019); Feldman and Steinke (2017); Blum et al. (2015).

The main difference from the standard `AboveThreshold` algorithm is that instead of halting after the first $\top$, the algorithm deletes all input elements $x \in S$ who "contribute" to the query that generated this $\top$, (that is, all $x \in S$ such that $f_i(x) = 1$) and continues.

Remark 2 *In the actual construction, instead of deleting elements from the data after the first time they "contribute" to a $\top$ answer, we delete them only after their kth contribution (for a given parameter $k \in \mathbb{N}$). We show that this degrades the privacy guarantees of our algorithm by at most a factor of $\approx \sqrt{k}$. We also extend the construction to handle* linear *queries, i.e., queries that range over $[0, 1]$ (rather than $\{0, 1\}$). This means that the "contribution" of a data element to the value of a query is not binary as we present it here in the introduction.*

The privacy analysis of `ThresholdMonitor` builds on the privacy analysis of the standard `AboveThreshold` algorithm (Dwork et al., 2009), in particular, on the analysis given by Hardt and Rothblum (2010). However, our analysis requires new ideas in order to show that by deleting some of the input elements we can avoid halting the algorithm after the first $\top$. We now give an informal (and simplified) presentation of the privacy analysis of `ThresholdMonitor`. Any informalities made hereafter will be removed in the sections that follow.

Fix two neighboring databases S and $S' = S \cup \{x'\}$, and consider the execution of (the simplified) algorithm `ThresholdMonitor` on S and on S'. Observe that once we reach an iteration i^* in which $a_{i^*} = \top$ and $f_{i^*}(x') = 1$ then x' is deleted from the data (during the execution on S'). After this iteration, we have that S and S' are identical, and the two executions proceed exactly the same. Moreover, time steps $i \leq i^*$ in which $f_i(x') = 0$ are also not very interesting, since in these time steps we have that $f_i(S) = f_i(S')$, and so the two executions behave the same.

Therefore, for the privacy analysis, we only need to argue about time steps $i \leq i^*$ such that $f_i(x') = 1$. At a high level, we partition these time steps into two sets:

$$I_{\text{far}} = \{i \leq i^* : f_i(x') = 1 \text{ and } f_i(S) \ll t\}, \qquad \text{and} \qquad I_{\text{almost}} = \{i \leq i^* : f_i(x') = 1\} \setminus I_{\text{far}}.$$

We show that, essentially, time steps in I_{far} come "for free", and we do not need to pay for them in the privacy analysis. Intuitively, the reason is that in these time steps we have that both $f_i(S)$ and $f_i(S')$ are very far from the threshold t, and in both executions, with overwhelming probability, we will get that $a_i = \bot$.

For time steps in I_{almost} we do need to pay in the privacy analysis (by arguing about the noises that we add to the values of the queries). What we show, however, is that there cannot be too many time steps in I_{almost}. Intuitively, in every time step $i \in I_{\text{almost}}$ there is a noticeable probability of returning $a_i = \top$, in which case $i = i^*$ is the last index in I_{almost}.

First challenge in the analysis. *The partition of the time steps into I_{far} and I_{almost} depends on i^*, which depends on the noises that we add to the values of the queries at each time step. This means that conditioning on i^* (or on I_{far} and I_{almost}) changes the distribution of these noises. Can we still use them to argue about the privacy costs of the time steps in I_{almost}?*

Resolution. We overcome this issue using the following trick. Instead of adding a single (Laplace) noise to the value of each query, we add *two* independent noises: $\hat{f}_i \leftarrow f_i(S) + w_i + v_i$, where w_i and v_i are independent random noises, sampled from the Laplace distribution. This allows us to define the partition of the time steps based on the w_i's, while arguing about the privacy loss based on the v_i's. So now time steps in I_{far} are time steps i such that $f_i(S) + w_i \ll t$. Even

when conditioning on that, v_i is still distributed according to the Laplace distribution, and hence, can mask the difference between $f_i(S)$ and $f_i(S')$ (as is standard in the literature of differential privacy). This idea is general and could be applicable in other cases: It imposes a convenient-to-analyze structure on the probability spaces generated by the noises so that it is easier to argue in conditional subspaces.

Second challenge in the analysis. *Do time steps in I_{far} really come for free?*

Resolution. As we described it, in these time steps, *with overwhelming probability* over sampling v_i we have that $a_i = \perp$ (in both executions). This can be used to show that these time steps increase our privacy parameter δ by at most a factor of m, where m bounds the total number of time steps. While this privacy loss is not too bad (since δ is typically negligible), we want to avoid it. In particular, the privacy guarantees of the standard `AboveThreshold` algorithm do not depend on the number of time steps m, and we would like to preserve this benefit in our analysis. To achieve this we modify the distribution of the noises v_i (by capping them at a certain value) in order to make sure that for the vast majority of time steps $i \in I_{\mathrm{far}}$ we have that $a_i = \perp$ *with probability one* during both executions (over sampling v_i). This means that the vast majority of time steps in I_{far} really come for free (in terms of the privacy analysis). This allows us to avoid blowing up the privacy parameter δ by a factor of m. In addition, we show that this modification to the distribution of the v_i's does not break our privacy analysis for time steps $i \in I_{\mathrm{almost}}$.

Remark 3 *Unlike the traditional* `AboveThreshold` *algorithm, which is $(\varepsilon, 0)$-differentially private, our variant (algorithm* `ThresholdMonitor`*) satisfies (ε, δ)-differential privacy. However, in the context of the shifting heavy-hitters problem, discussed below, our comparison is to the traditional algorithm with advanced composition. Hence, in that context, the traditional algorithm also results in (ε, δ)-differential privacy*

1.1.2. THE SHIFTING HEAVY-HITTERS PROBLEM

We demonstrate our technique by applying it to the *shifting heavy-hitters* problem. In this problem there are n users and a data domain X. On every time step $i \in [m]$, every user $j \in [n]$ gets an input $x_{i,j} \in X$, and the goal is to report all the *current heavy-hitters*. That is, on time step i, the goal is to identify all data elements x such that $w_i(x) \triangleq |\{j \in [n] : x_{i,j} = x\}| \gg 0$.

Definition 4 *We say that an algorithm $\mathcal{A}$ solves the shifting heavy-hitters problem with error parameter τ and failure probability β if it guarantees the following with probability at least $(1 - \beta)$:*

1. *The algorithm never identifies elements with weight zero as being heavy. That is, on every time step $i \in [m]$, if the algorithm identifies an element x then $w_i(x) > 0$.*

2. *On every time step $i \in [m]$ the algorithm identifies every data element x such that $w_i(x) \geq \tau$.*

The shifting heavy-hitters problem generalizes the (standard) heavy-hitters problem, in which $m = 1$. Algorithms for the (standard) heavy-hitters problem have found many applications in both the theory and the practice of differential privacy, including some of the most widespread industrial deployments of differential privacy to date. However, many of these deployments periodically re-collect the data in order to re-identify the heavy-hitters, e.g., on a daily basis. As a result, the problem that actually underlies these deployments is the *shifting heavy-hitters* problem.

A naive approach for privately solving the shifting heavy-hitters problem would be to run a differential private algorithm for histograms on every time step $i \in [m]$. However, such a solution would have error at least $\sqrt{m}$ because of composition. This solution can be improved by using the (standard) sparse vector technique in order to identify the time steps in which a heavy-hitter exists, and only run a private algorithm for histograms during these identified time steps. However, in general, the number k of time steps in which a heavy-hitter exists can be quite large, and the error with this approach scales with $\sqrt{k}$. We present a refined solution in which the error only scales with $\sqrt{k^*}$, where k^* is the maximal number of times that a *single user* holds a heavy-hitter as input.

Example 1 *To illustrate the difference between k and k^*, consider an input stream to the shifting heavy-hitters problem with the following properties:*

1. *There are three partitions of the set of users $[n]$:*

   - $A_1, A_2, \ldots, A_{\tilde{O}(n)} \subseteq [n]$, *where every A_i is of size $\tilde{O}(1)$.*

   - $B_1, B_2, \ldots, B_{\tilde{O}\left(n^{3/4}\right)} \subseteq [n]$, *where every B_i is of size $\tilde{O}\left(n^{1/4}\right)$.*

   - $C_1, C_2, \ldots, C_{\tilde{O}\left(n^{2/3}\right)} \subseteq [n]$, *where every C_i is of size $\tilde{O}\left(n^{1/3}\right)$.*

2. *Every such subset gets a "heavy-hitter" exactly once throughout the stream, at different time steps. Otherwise, users get distinct inputs.*

Observe that every user "participates" in three heavy-hitter occurrences, and hence, $k^* = 3$. Indeed, in this example, our algorithm is capable of identifying all *of the $\tilde{O}(n)$ heavy-hitters throughout the execution. In contrast, using the standard sparse vector technique only allows us to identify the heavy-hitters generated by the C_i's. To see this, observe that in order to identify the C_i's, the standard sparse vector technique needs to halt $\tilde{O}\left(n^{2/3}\right)$ times, which result in error at least $\tilde{O}\left(n^{1/3}\right)$ due to composition. This in turn means that this approach cannot identify the heavy-hitters generated by the A_i's and the B_i's.*

1.2. Related work

Mostly related to our work are the works on the sparse vector technique and its variants.[2] In particular, our analysis for algorithm `ThresholdMonitor` is based on techniques introduced by Hardt and Rothblum (2010). Our variant of the sparse vector technique is similar to that of Hardt and Rothblum (2010) in that both variants do not add noise to the threshold itself, only to the values of the given queries. However, our analysis becomes more complicated in order to avoid halting the algorithm after the first $\top$ answer.

Another related work is the recent work of Feldman and Zrnic (2020), who presented a refinement of standard composition theorems, in which they account separately for the privacy loss of different data elements, and delete elements from the data once they exhaust their privacy budget.

---

2. See, e.g., Dwork et al. (2009); Hardt and Rothblum (2010); Chen and Machanavajjhala (2015); Lyu et al. (2017); Bun et al. (2019).

2. Preliminaries

Notation.   Databases S, S' are called *neighboring* if one is obtained from the other by adding or deleting one element, e.g., $S' = S \cup \{x'\}$. For two random variables Y, Z we write $X \approx_{(\varepsilon, \delta)} Y$ to mean that for every event F it holds that $\Pr[X \in F] \leq e^\varepsilon \cdot \Pr[Y \in F] + \delta$, and $\Pr[Y \in F] \leq e^\varepsilon \cdot \Pr[X \in F] + \delta$. A function f that maps databases to the reals has *sensitivity* ℓ if for every neighboring S, S', it holds that $|f(S) - f(S')| \leq \ell$.

The Laplace mechanism.   The most basic constructions of differentially private algorithms are via the Laplace mechanism as follows. Recall that a random variable has probability distribution $\mathrm{Lap}(b)$ if its probability density function is $f(x) = \frac{1}{2b} \exp\left(-\frac{|x|}{b}\right)$, where $x \in \mathbb{R}$.

Theorem 5 (**The Laplace Mechanism** (Dwork et al., 2006)). *Let f be a function that maps databases to the reals with sensitivity ℓ. The mechanism $\mathcal{A}$ that on input S adds noise with distribution $\mathrm{Lap}(\frac{\ell}{\varepsilon})$ to the output of $f(S)$ preserves $(\varepsilon, 0)$-differential privacy.*

Differential privacy for interactive mechanisms.   In this work we consider *interactive* mechanisms that answer queries presented by a data analyst (or an adversary). Informally, an interactive mechanism M is said to be differentially private if for every adversary A that interacts with M it holds that the algorithm that simulates A and M interacting, and then outputs the transcript of the interaction, is differentially private. We give here a definition presented by Bun et al. (2019).

Definition 6 (**Differential Privacy for Interactive Mechanisms** (Dwork et al., 2006; Bun et al., 2019)). *Let M be a mechanism that takes an input database and answers adaptively chosen queries (by an adversary A from some family Q of possible queries). Mechanism M is (ε, δ)-differentially private if for every adversary A we have that $\mathtt{Adaptive}_{A \rightleftarrows M}$ (defined below) is (ε, δ)-differentially private.*

Algorithm 1 $\mathtt{Adaptive}_{A \rightleftarrows M}$

Input: Database S.

1. The database S is given to the mechanism M.
2. For $i = 1, 2, \ldots$
    (a) The adversary A chooses a query $q_i \in Q$.
    (b) The mechanism M is given q_i and returns a_i.
    (c) a_i is given to A.
3. When M halts or A halts, output $(q_1, a_1, q_2, a_2, \cdots)$.

---

3. Algorithm `ThresholdMonitor` and its Analysis

In this section we present out main result – Algorithm `ThresholdMonitor`, specified in Algorithm 2. The utility properties of the algorithm are straightforward (follow form a simple union bound on the magnitude of the noises throughout the execution).

Algorithm 2 ThresholdMonitor

Input: Database $S \in X^*$, privacy parameters ε, δ, threshold t, parameter k, and an adaptively chosen stream of counting queries $f_i : X \to [0, 1]$.

1. Initiate a counter $c(x) = 0$ for every $x \in X$.

2. Denote $\Delta = \frac{1}{\varepsilon} \log \left(\frac{1}{\delta} \right) \log \left(\frac{1}{\varepsilon} \log \frac{1}{\delta} \right)$.

3. In each round i, when receiving a query f_i, do the following:

    (a) Let $w_i \leftarrow \mathrm{Lap}\left(10\Delta \right)$ and let $v_i \leftarrow \mathrm{Lap}(\frac{1}{\varepsilon} \log \frac{1}{\delta})$.

    (b) Denote $\overline{v}_i = \min \{v_i, \Delta\}$.

    (c) Let $\hat{f}_i = f_i(S) + w_i + \overline{v}_i$.

    (d) If $\hat{f}_i < t$, then output $a_i = \bot$.

    (e) Otherwise:

    - Output $a_i = \top$.
    - For every $x \in X$ set $c(x) \leftarrow c(x) + f_i(x)$.
    - Delete from S every element x such that $c(x) \geq k$.

    (f) Proceed to the next iteration.

Theorem 7 *Consider an execution of* ThresholdMonitor *on a database S and on a stream of m adaptively chosen queries: $f_1, f_2, \ldots, f_m$. Let S_i denote the database S as it is before answering the ith query. With probability at least $1 - \beta$, for every i such that $a_i = \top$ it holds that $f_i(S_i) \geq t - \tilde{O}\left(\frac{1}{\varepsilon} \log \left(\frac{1}{\delta} \right) \log \left(\frac{m}{\beta} \right) \right)$, and for every i such that $a_i = \bot$ it holds that $f_i(S_i) \leq t + \tilde{O}\left(\frac{1}{\varepsilon} \log \left(\frac{1}{\delta} \right) \log \left(\frac{m}{\beta} \right) \right)$.*

We now present the privacy guarantees of the algorithm. Note that, as with AboveThreshold, the number of possible rounds unbounded. Nevertheless, the algorithm is differentially private:

Theorem 8 *Algorithm* ThresholdMonitor *is $(\xi, 3\delta)$-differentially private, for $\xi = \frac{75(k+1)\varepsilon}{\log \frac{1}{\delta}} + 25\varepsilon$. In particular, for $k = \log \frac{1}{\delta}$ we have that algorithm* ThresholdMonitor *is $(O(\varepsilon), 3\delta)$-differentially private.*

Proof Let $\mathcal{B} : X^n \to \{\bot, \top\}^*$ denote the function that simulates ThresholdMonitor interacting with a given adversary A (cf. algorithm 1) and returns the transcript. Without loss of generality, we may assume that our adversary is deterministic.[3] Fix two neighboring databases S and $S' = S \cup \{x'\}$ for some $x' \in X$, and let S_i (or S'_i) denote the database S (or S') as it is before answering the ith query during the execution on S (or on S'). Also let $c_i(\cdot)$ denote the counter $c(\cdot)$ as it is after the ith iteration. Denote $\Delta = \frac{1}{\varepsilon} \log \left(\frac{1}{\delta} \right) \log \left(\frac{1}{\varepsilon} \log \frac{1}{\delta} \right)$. We need the following definition.

---

3. Say that for every deterministic adversary we have that $\Pr[\mathcal{B}(S) \in F] \leq e^\varepsilon \cdot \Pr[\mathcal{B}(S') \in F] + \delta$. Now for a random adversary, let r denote a possible fixture of its random coins. It holds that $\Pr[\mathcal{B}(S) \in F] = \sum_r \Pr[r] \Pr[\mathcal{B}(S) \in F | r] \leq \sum_r \Pr[r](e^\varepsilon \cdot \Pr[\mathcal{B}(S') \in F | r] + \delta) = e^\varepsilon \cdot \Pr[\mathcal{B}(S') \in F] + \delta$.

Definition 9 *We say that an* almost-top *occurs at time step i if $a_i = \perp$ and $c_i(x') < k$ and*

$$f_i(S_i) + w_i \geq t - 2\Delta.$$

Intuitively, almost-tops are time steps in which the noisy value of the query $\hat{f}_i$ *almost* crossed the threshold t, but did not, before the iteration in which $c(x') \geq k$. (This intuitive explanation is somewhat inaccurate because the definition of almost-tops only takes into account the noise w_i, without the noise $\overline{v}_i$.) We also identify the following subset of almost-tops, which we call *special-almost-tops*. In the introduction we mentioned that the vast majority of the time steps i for which $f_i(S_i) + w_i \ll t$ come "for free" in terms of the privacy analysis. Intuitively, special-almost-tops represent the small portion of these time steps that do not come for free.

Definition 10 *We say that a* special-almost-top *occurs at time step i if $a_i = \perp$ and $c_i(x') < k$ and*

$$t - f_i(x') - \Delta \leq f_i(S_i) + w_i < t - \Delta.$$

We define the following events.

Event E_1: $$\sum_{i:\text{ almost-top}} f_i(x') \leq 15(k+1) + 5 \cdot \log \frac{1}{\delta}$$

Event E_2: $$|\{i :\ \text{special-almost-top}\}| \leq 30(k+1) + 10 \cdot \log \frac{1}{\delta}$$

Event E_3: $$E_3 = E_1 \wedge E_2$$

Lemma 11 *Consider the execution of $\mathcal{B}(S)$. Event E_3 occurs with probability at least $1 - 3\delta$.*

The formal proof of Lemma 11 is given in Appendix A. Here we only sketch the argument.

Proof sketch Let i^* denote the first time step such that $c_{i^*}(x') \geq k$. Recall that each w_i is sampled from $\mathrm{Lap}(10\Delta)$. Hence, in every time step i, the probability that $f_i(S_i) + w_i \geq t - 2\Delta$ is not much larger than the probability that $f_i(S_i) + w_i \geq t + 2\Delta$, in which case $a_i = \top$ with high probability (because v_i is sampled from $\mathrm{Lap}(b)$ for $b \ll \Delta$). This means that the probability of an almost-top occurring in some time step i is not much larger then the probability of a top occurring in this time step. Hence, it is very unlikely that $\sum_{i:\text{ almost-top}} f_i(x')$ would be much bigger than $\sum_{i \leq i^*:\ a_i = \top} f_i(x')$, which is bounded by $k + 1$ by definition of i^*. This argument can be formalized to show that Event E_1 occurs with high probability.

For Event E_2, recall that for a time step i to be a special-almost-top, the noise w_i must satisfy $w_i \in [t - f_i(x') - \Delta, t - \Delta]$, which happens with probability at most $f_i(x')$, even conditioned on this time step being a (non-special) almost-top. (This is true by the properties of the Laplace distribution.) Therefore, the expected number of special-almost-tops is at most $\sum_{i:\text{ almost-top}} f_i(x')$, which is at most $15(k+1) + 5 \cdot \log \frac{1}{\delta}$ with high probability. Using standard concentration bounds, this shows that Event E_2 occurs with high probability. ∎

We continue with the proof of Theorem 8. Observe that, since we assume that our adversary is deterministic, every query f_i is completely determined by the values of the previous answers $a_1, \ldots, a_{i-1}$. We use the following notation.

Notation 12 *Given a possible outcome vector $\vec{a} \in \{\bot, \top\}^*$, we write $f_{\vec{a},i}$ to denote the query chosen by the adversary after seeing the first $i - 1$ answers in $\vec{a}$.*

An important thing to note is that the outcome vector $\vec{a}$ and the noise vector $W = (w_1, w_2, \dots)$ completely determine whether E_3 occurs. The following lemma is a key in our analysis.

Lemma 13 *The following holds for every outcome vector $\vec{a} \in \{\bot, \top\}^*$ and noise vector $W = (w_1, w_2, \dots)$ for which event E_3 occurs. Let $i^* = i^*(\vec{a})$ denote the first time step such that $c_{i^*}(x') \geq k$. Let $I_{\text{top}} = I_{\text{top}}(\vec{a})$ denote the set of all time steps $i \leq i^*$ such that $a_i = \top$. Let W' denote a vector identical to W, except that every coordinate $i \in I_{\text{top}}$ is decreased by $f_{\vec{a},i}(x')$. That is, $w'_i = w_i - f_{\vec{a},i}(x')$ for every $i \in I_{\text{top}}$, and $w'_i = w_i$ for every other index i. Then,*

$$\Pr\left[\mathcal{B}(S') = \vec{a}\,\big|\,W'\right] \;\leq\; \Pr\left[\mathcal{B}(S) = \vec{a}\,\big|\,W\right] \;\leq\; \exp\left(\frac{75(k+1)\varepsilon}{\log\frac{1}{\delta}} + 25\varepsilon\right)\cdot\Pr\left[\mathcal{B}(S') = \vec{a}\,\big|\,W\right].$$

(Note that since we fixed W and W', these probabilities are only over the noises v_i.)

We now show how Theorem 8 follows from Lemma 13. The proof of Lemma 13 is given in Appendix B. For simplicity we will assume that the noise vector W is discrete; the analysis could be extended to handle continues vectors W.[4] Let $F \subseteq \{\bot, \top\}^*$ be a set of possible outcomes. Denote $\xi = \frac{75(k+1)\varepsilon}{\log\frac{1}{\delta}} + 25\varepsilon$. We have that

$$\Pr[\mathcal{B}(S) \in F] \leq 3\delta + \Pr[\mathcal{B}(S) \in F \wedge E_3] = 3\delta + \sum_{\vec{a} \in F} \Pr[\mathcal{B}(S) = \vec{a} \wedge E_3]$$

$$= 3\delta + \sum_{\vec{a} \in F} \sum_{W} \Pr[W] \cdot \Pr[\mathcal{B}(S) = \vec{a} \wedge E_3 \mid W] = 3\delta + \sum_{\vec{a} \in F} \sum_{\substack{W: \\ E_3 \text{ occurs} \\ \text{for } \vec{a}, W}} \Pr[W] \cdot \Pr[\mathcal{B}(S) = \vec{a} \mid W]$$

$$\leq 3\delta + \sum_{\vec{a} \in F} \sum_{\substack{W: \\ E_3 \text{ occurs} \\ \text{for } \vec{a}, W}} \Pr[W] \cdot e^{\xi} \cdot \Pr[\mathcal{B}(S') = \vec{a} \mid W] \tag{1}$$

$$\leq 3\delta + \sum_{\vec{a} \in F} \sum_{W} \Pr[W] \cdot e^{\xi} \cdot \Pr[\mathcal{B}(S') = \vec{a} \mid W] = 3\delta + e^{\xi} \cdot \Pr[\mathcal{B}(S') \in F],$$

where (1) follows from Lemma 13. The other direction is similar, except that we also need to argue about replacing W with W'. Recall that each W' is obtained from W by decreasing w_i by $f_{\vec{a},i}(x')$ for every coordinate $i \in I_{\text{top}}$. Recall that by Step 3e of the algorithm (and by definition of I_{top}) we have that $\sum_{i \in I_{\text{top}}} f_{\vec{a},i}(x') \leq k + 1$. Also recall that every w_i is sampled from $\text{Lap}(10\Delta)$ for

---

4. In fact, focusing on discrete vectors W is without loss of generality, since we may round each coordinate of W, and each value $f_i(x)$, to the nearest multiplication of $1/N$, for some $N \geq n$. This changes the value of every query $f_i(S)$ by at most 1, which has basically no effect on the utility of the algorithm (since the noise we add to every $f_i(S)$ has variance much larger than 1).

$\Delta = \frac{1}{\varepsilon} \log\left(\frac{1}{\delta}\right) \log\left(\frac{1}{\varepsilon} \log \frac{1}{\delta}\right)$. Therefore, for every such W and its corresponding vector W' we have

$$\Pr[W] = \left(\prod_{i \in I_{\text{top}}} \Pr[w_i]\right) \cdot \left(\prod_{i \notin I_{\text{top}}} \Pr[w_i]\right) \geq \left(\prod_{i \in I_{\text{top}}} e^{-f_{\vec{a},i}(x')/(10\Delta)} \cdot \Pr[w_i']\right) \cdot \left(\prod_{i \notin I_{\text{top}}} \Pr[w_i]\right)$$

$$= \exp\left(-\frac{1}{10\Delta} \sum_{i \in I_{\text{top}}} f_{\vec{a},i}(x')\right) \cdot \Pr[W'] \geq \exp\left(-\frac{k+1}{10\Delta}\right) \cdot \Pr[W'] \geq e^{-\xi} \cdot \Pr[W'].$$

(2)

Hence,

$$\Pr[\mathcal{B}(S) \in F] \geq \Pr[\mathcal{B}(S) \in F \wedge E_3] = \sum_{\vec{a} \in F} \Pr[\mathcal{B}(S) = \vec{a} \wedge E_3]$$

$$= \sum_{\vec{a} \in F} \sum_{W} \Pr[W] \cdot \Pr[\mathcal{B}(S) = \vec{a} \wedge E_3 \mid W] = \sum_{\vec{a} \in F} \sum_{\substack{W: \\ E_3 \text{ occurs} \\ \text{for } \vec{a}, W}} \Pr[W] \cdot \Pr[\mathcal{B}(S) = \vec{a} \mid W]$$

$$\geq \sum_{\vec{a} \in F} \sum_{\substack{W: \\ E_3 \text{ occurs} \\ \text{for } \vec{a}, W}} e^{-\xi} \cdot \Pr[W'] \cdot \Pr[\mathcal{B}(S') = \vec{a} \mid W'] \tag{3}$$

$$= \sum_{\vec{a} \in F} \sum_{\substack{W: \\ E_3 \text{ occurs} \\ \text{for } \vec{a}, W'}} e^{-\xi} \cdot \Pr[W'] \cdot \Pr[\mathcal{B}(S') = \vec{a} \mid W'] \tag{4}$$

$$= e^{-\xi} \cdot \sum_{\vec{a} \in F} \sum_{W} \Pr[W'] \cdot \Pr[\mathcal{B}(S') = \vec{a} \wedge E_3 \mid W'] = e^{-\xi} \cdot \sum_{\vec{a} \in F} \Pr[\mathcal{B}(S') = \vec{a} \wedge E_3]$$

$$= e^{-\xi} \cdot \Pr[\mathcal{B}(S') \in F \wedge E_3]$$

$$\geq e^{-\xi} \cdot \Pr[\mathcal{B}(S') \in F \wedge E_3] + e^{-\xi} \cdot \Pr[\mathcal{B}(S') \in F \wedge \overline{E_3}] - e^{-\xi} \cdot 3\delta$$

$$= e^{-\xi} \cdot \Pr[\mathcal{B}(S') \in F] - e^{-\xi} \cdot 3\delta,$$

where (3) follows from Lemma 13 and from Inequality (2), and where (4) holds because E_3 occurs for $\vec{a}, W$ if and only if it occurs for $\vec{a}, W'$. Specifically, E_3 is determined from $\vec{a}$ (the set I_{top} is also determined from $\vec{a}$) and from $\{w_i : i \notin I_{\text{top}}\}$, regardless of the value of the w_i's in I_{top}. This completes the proof of Theorem 8 using Lemma 13. (The proof of Lemma 13 is given in Appendix B.) ∎

3.1. Composition for k tops

In Theorem 8 we showed that when executing algorithm `ThresholdMonitor` with a parameter k, then the privacy guarantees of the algorithm degrade at most linearly with $k/\log\frac{1}{\delta}$. We give a refined analysis, showing that the privacy guarantees degrade only (roughly) proportionally to $\sqrt{k}$. This refined analysis follows by applying known composition theorems for differential privacy (Dwork et al., 2010) to the construction of algorithm `ThresholdMonitor`. Although standard, this refined analysis requires some technical work, as we need to "break" algorithm `ThresholdMonitor` into smaller components in order to "compose" it more efficiently using composition theorems. The details are given in Appendix C. Here we only state the result.

Theorem 14 *Algorithm* `ThresholdMonitor` *with parameter k is $(\varepsilon_0, 3k\delta)$-differentially private for $\varepsilon_0 = O\left(\sqrt{k}\varepsilon + k\varepsilon^2\right)$.*

4. The Shifting Heavy-Hitters Problem

In the *shifting heavy-hitters* problem there are n users and a data domain X. On every time step $i \in [m]$, every user j gets an input $x_{i,j} \in X$, and the goal is to report all the *current heavy-hitters*. That is, on time step i, the goal is to identify all data elements x such that

$$w_i(x) \triangleq |\{j \in [n] : x_{i,j} = x\}| \gg 0.$$

Definition 15 *We say that an algorithm $\mathcal{A}$ solves the shifting heavy-hitters problem with error parameter τ and failure probability β if it guarantees the following with probability at least $(1 - \beta)$:*

1. *The algorithm never identifies elements with weight zero as being heavy. That is, on every time step i, if the algorithm identifies an element x then $w_i(x) > 0$.*

2. *On every time step i the algorithm identifies every data element x such that $w_i(x) \geq \tau$.*

Of course, we want to solve the shifting heavy-hitters problem while guaranteeing differential privacy. However, as users' data evolve over time, algorithms for the shifting heavy-hitters problem do not exactly fit the standard (non-interactive) definition of differential privacy (Definition 1), nor the interactive variant (Definition 6) in which the queries are chosen adaptively, but users' data do not change. We will use the following definition of Beimel et al. for what it means for a mechanism to be differentially private when the data evolves over time (generalizing the standard definition of differential privacy of Dwork et al. (2006)).

Notation 16 *Two databases S, S' (of the same size) are called j-neighboring if they agree on every coordinate $i \neq j$.*

Definition 17 (**Differential Privacy for Interactive Mechanisms with Evolving Data** (Dwork et al., 2006; Beimel et al.)). *Let M be a mechanism with the following properties. On each round $i \in [m]$, the mechanism gets a database S_i (of size n) and a query q_i (from some family Q of possible queries), and responds with an answer a_i. Mechanism M is (ε, δ)-differentially private in the evolving setting if for every adversary A and every $j \in [n]$ we have that $\mathtt{Game}_{A,M,j}$, defined below, is (ε, δ)-differentially private (w.r.t. the input bit b).*

Towards designing a private algorithm for the shifting heavy-hitters problem, we generalize algorithm `ThresholdMonitor` for the evolving data setting. We call the modified construction `ThresholdMonitorEvolving`. The details are given in Appendix D. Let us now return to the shifting heavy-hitters problem. A technical issue here is that there is a certain circularity between τ (the error parameter of the algorithm) and k (the number of times in which a heavy-hitter exists). The reason is that τ defines a threshold such the algorithm must identify every element with weight at least τ as being a "heavy-hitter". Therefore, when τ gets smaller, then there might be more time steps in which a heavy-hitter exists, which increases k. This, in turn, increases τ, because the error of the algorithm grows with k.

Algorithm 3 $\texttt{Game}_{A,M,j}$

Input: A bit $b \in \{0, 1\}$. (The bit b is unknown to A and M.)

1. For $i = 1, 2, \ldots, m$

   (a) The adversary A chooses two j-neighboring databases S_i^0, S_i^1 and a query $q_i \in Q$.

   (b) The mechanism M is given S_i^b and q_i and returns a_i.

   (c) a_i is given to A.

2. Output A's view of the game, that is $S_1^0, S_1^1, q_1, a_1, \ldots, S_m^0, S_m^1, q_m, a_m$, and the internal randomness of A.

Notation 18 *For $k \in \mathbb{N}$ we denote $\tau(k) = \tilde{O}\left(\frac{\sqrt{k}}{\varepsilon} \cdot \log\left(\frac{1}{\delta}\right) \cdot \log\left(\frac{m \cdot |X|}{\beta}\right)\right)$.*

Notation 19 *Let k^* be such that the maximal number of times that a single user holds an element with weight $\tau(k^*)$ is at most k^*. That is, $k^* \geq \max_{j \in [n]} |\{i \in [m] : w_i(x_{i,j}) \geq \tau(k^*)\}|$.*

Using algorithm $\texttt{ThresholdMonitorEvolving}$, we present an algorithm for the shifting heavy-hitters problem in which the error scales roughly as $\tau(k^*) \approx \sqrt{k^*}$. We obtain the following theorem; the details are given in Appendix D.

Theorem 20 *There exists an (ε, δ)-differentially private algorithm for solving the shifting heavy-hitters problem with failure probability β and error $O\left(\tau(k^*)\right)$.*

Qualitatively, the error in Theorem 20 is always at most the error obtained using the standard sparse vector technique, and it can be significantly lower in some cases (see Example 1 in the introduction). We remark that the assumption on k^* is only used in the utility analysis, and that the privacy of our algorithm is guaranteed even if this assumption does not hold.

Acknowledgments

References

Gilles Barthe, Noémie Fong, Marco Gaboardi, Benjamin Grégoire, Justin Hsu, and Pierre-Yves Strub. Advanced probabilistic couplings for differential privacy. In *CCS*, pages 55–67, 2016a.

Gilles Barthe, Marco Gaboardi, Benjamin Grégoire, Justin Hsu, and Pierre-Yves Strub. Proving differential privacy via probabilistic couplings. In *LICS*, pages 749–758, 2016b.

Raef Bassily, Abhradeep Guha Thakurta, and Om Dipakbhai Thakkar. Model-agnostic private learning. In *NeurIPS*, pages 7102–7112, 2018.

Amos Beimel, Kobbi Nissim, and Uri Stemmer. Personal communication.

Avrim Blum, Jamie Morgenstern, Ankit Sharma, and Adam Smith. Privacy-preserving public information for sequential games. In *ITCS*, pages 173–180, 2015.

Mark Bun, Thomas Steinke, and Jonathan Ullman. Make up your mind: The price of online queries in differential privacy. *Journal of Privacy and Confidentiality*, 9(1), 2019.

Yan Chen and Ashwin Machanavajjhala. On the privacy properties of variants on the sparse vector technique. *CoRR*, abs/1508.07306, 2015.

Rachel Cummings, Michael J. Kearns, Aaron Roth, and Zhiwei Steven Wu. Privacy and truthful equilibrium selection for aggregative games. In *WINE*, pages 286–299, 2015.

Rachel Cummings, Sara Krehbiel, Yuliia Lut, and Wanrong Zhang. Privately detecting changes in unknown distributions. *CoRR*, abs/1910.01327, 2019.

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, pages 265–284, 2006.

Cynthia Dwork, Moni Naor, Omer Reingold, Guy N. Rothblum, and Salil P. Vadhan. On the complexity of differentially private data release: efficient algorithms and hardness results. In *STOC*, pages 381–390, 2009.

Cynthia Dwork, Guy N. Rothblum, and Salil P. Vadhan. Boosting and differential privacy. In *FOCS*, pages 51–60, 2010.

Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Leon Roth. Preserving statistical validity in adaptive data analysis. In *STOC*, pages 117–126, 2015.

Vitaly Feldman and Thomas Steinke. Generalization for adaptively-chosen estimators via stable median. In *COLT*, pages 728–757, 2017.

Vitaly Feldman and Tijana Zrnic. Individual privacy accounting via a renyi filter. *CoRR*, abs/2008.11193, 2020.

Anupam Gupta, Katrina Ligett, Frank McSherry, Aaron Roth, and Kunal Talwar. Differentially private combinatorial optimization. In *SODA*, pages 1106–1125, 2010.

Moritz Hardt and Guy N. Rothblum. A multiplicative weights mechanism for privacy-preserving data analysis. In *FOCS*, pages 61–70, 2010.

Avinatan Hassidim, Haim Kaplan, Yishay Mansour, Yossi Matias, and Uri Stemmer. Adversarially robust streaming algorithms via differential privacy. *CoRR*, abs/2004.05975, 2020.

Justin Hsu, Aaron Roth, and Jonathan Ullman. Differential privacy for the analyst via private equilibrium computation. In *STOC*, pages 341–350, 2013.

Haim Kaplan, Katrina Ligett, Yishay Mansour, Moni Naor, and Uri Stemmer. Privately learning thresholds: Closing the exponential gap. In *COLT*, pages 2263–2285, 2020a.

Haim Kaplan, Micha Sharir, and Uri Stemmer. How to find a point in the convex hull privately. In *SoCG*, pages 52:1–52:15, 2020b.

Katrina Ligett, Seth Neel, Aaron Roth, Bo Waggoner, and Steven Z. Wu. Accuracy first: Selecting a differential privacy level for accuracy constrained ERM. In *NIPS*, pages 2566–2576, 2017.

Min Lyu, Dong Su, and Ninghui Li. Understanding the sparse vector technique for differential privacy. *VLDB*, pages 637–648, 2017.

Anupama Nandi and Raef Bassily. Privately answering classification queries in the agnostic PAC model. In *ALT*, pages 687–703, 2020.

Kobbi Nissim and Uri Stemmer. Clustering algorithms for the centralized and local models. In *ALT*, pages 619–653, 2018.

Kobbi Nissim, Uri Stemmer, and Salil Vadhan. Locating a small cluster privately. In *PODS*, pages 413–427, 2016.

Touqir Sajed and Or Sheffet. An optimal private stochastic-mab algorithm based on optimal private stopping rule. In *ICML*, pages 5579–5588, 2019.

Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In *CCS*, pages 1310–1321, 2015.

Thomas Steinke and Jonathan Ullman. Between pure and approximate differential privacy. *Journal of Privacy Confidentiality*, 7(2), 2016.

Uri Stemmer. *Individuals and privacy in the eye of data analysis*. PhD thesis, Ben-Gurion University of the Negev, 2016. Supervisors – Amos Beimel and Kobbi Nissim.

Jonathan Ullman. Private multiplicative weights beyond linear queries. In *PODS*, pages 303–312, 2015.

Appendix A. Proof of Lemma 11

In this section we prove Lemma 11, which states that Event E_3 occurs with high probability. As we mentioned, the intuitive explanation is that in every round i, the probability of an almost-top is roughly the same as the probability of an actual top, and hence, the almost-tops and the actual tops should be "balanced". The analysis we give here is an adaptation of the analysis of Gupta et al. (2010) for a different scenario.

Towards proving Lemma 11, let us consider the following m-round game against an adversary.

A m-round game

In each round i:

1. The adversary chooses $0 \leq p_i \leq \frac{1}{2}$ and $\frac{p_i}{4} \leq q_i \leq 1 - p_i$, and $\gamma_i \in [0, 1]$, possibly based on the first $(i-1)$ rounds.

2. A random variable $X_i \in \{0, 1, 2\}$ is sampled (and the outcome is given to the adversary), where $\Pr[X_i = 1] = p_i$ and $\Pr[X_i = 2] = q_i$ and $\Pr[X_i = 0] = 1 - p_i - q_i$.

We refer to the values γ_i as *weights*. For $k \in \mathbb{R}$ and $i \in [m]$, let $Z_i^{(k)}$ be the indicator for the event that

$$\sum_{j=1}^{i} \mathbb{1}\{X_j = 2\} \cdot \gamma_j \leq k.$$

That is, $Z_i^{(k)} = 1$ if and only if the sum of weights γ_j for time steps $1 \leq j \leq i$ such that $X_j = 2$ is at most k. Note that for $k < 0$ we have that $Z_i^{(k)} \equiv 0$. For $k \in \mathbb{R}$ let

$$W^{(k)} = \sum_{i=1}^{m} \mathbb{1}\{X_i = 1\} \cdot \gamma_i \cdot Z_i^{(k)}.$$

This can be interpreted as follows. Let i^* denote the first time step in which $Z_{i^*}^{(k)} = 0$. Then $W^{(k)}$ is the sum of weights γ_i for time steps $1 \leq i < i^*$ in which $X_i = 1$.

Remark 21 *Intuitively, we can think of k as the "budget" of the adversary – every time that $X_i = 2$ then the budget is decreased by γ_i. The adversary's goal is to maximize the sum of weights γ_i for time steps in which $X_i = 1$ before the budget runs out. That is, every time that $X_i = 1$ the adversary gets γ_i as a "reward" (before the budget runs out), and every time that $X_i = 2$ the budget is decreased by γ_i. The random variable $W^{(k)}$ represents the sum of rewards obtained by the adversary before the budget runs out.*

Towards analyzing $W^{(k)}$, we define the following partial sums. For $j \in [m]$ and $k \in \mathbb{R}$, let

$$W_j^{(k)} = \sum_{i=j}^{m} \mathbb{1}\{X_i = 1\} \cdot \gamma_i \cdot Z_i^{(k)}.$$

Observe that $W^{(k)} = W_1^{(k)}$.

Claim 22 *For every adversary's strategy, every $k \geq 0$, every $\lambda \in \mathbb{R}$, and every $j \in [m]$ we have that* $\Pr[W_j^{(k)} > \lambda] \leq \exp\left(-\frac{\lambda}{5} + 3(k+1)\right)$. *In particular,* $\Pr[W^{(k)} > \lambda] \leq \exp\left(-\frac{\lambda}{5} + 3(k+1)\right)$.

Proof The proof is by reverse induction on j. For the base case ($j = m$) observe that $W_m^{(k)} \in [0, 1]$, and hence if $\lambda > 1$ then $\Pr[W_m^{(k)} > \lambda] = 0$. Also observe that if $\lambda \leq 0$ then $\Pr[W_m^{(k)} > \lambda] \leq 1 \leq \exp(-\frac{\lambda}{5} + 3(k+1))$. Assume therefore that $0 < \lambda \leq 1$. Then,

$$
\begin{aligned}
\Pr[W_m^{(k)} > \lambda] &= \Pr[\mathbb{1}\{X_m = 1\} \cdot \gamma_m \cdot Z_m^{(k)} > \lambda] \\
&\leq \Pr[X_m = 1] \\
&= p_m \\
&\leq \frac{1}{2} \\
&< \exp(-1/5) \\
&\leq \exp(-\lambda/5 + 3(k+1)).
\end{aligned}
$$

This completes the analysis for the base case ($j = m$). Now suppose that for any adversary's strategy, for every $\lambda \in \mathbb{R}$, and for every $k \geq 0$ it holds that $\Pr[W_{j+1}^{(k)} > \lambda] \leq \exp\left(-\frac{\lambda}{5} + 3(k+1)\right)$. We will show the same for j. Fix $k \geq 0$ and fix $\lambda \in \mathbb{R}$. We may assume that $\lambda > 0$, as otherwise $\Pr[W_j^{(k)} > \lambda] \leq 1 \leq \exp(-\frac{\lambda}{5} + 3(k+1))$. Suppose that the adversary chooses p_j, q_j in round j. Then,

$$
\begin{aligned}
\Pr[W_j^{(k)} > \lambda] &= \Pr[\mathbb{1}\{X_j = 1\} \cdot \gamma_j \cdot Z_j^{(k)} + W_{j+1}^{(k)} > \lambda] \\
&= \Pr[X_j = 0] \cdot \Pr[\mathbb{1}\{X_j = 1\} \cdot \gamma_j \cdot Z_j^{(k)} + W_{j+1}^{(k)} > \lambda | X_j = 0] \\
&\quad + \Pr[X_j = 1] \cdot \Pr[\mathbb{1}\{X_j = 1\} \cdot \gamma_j \cdot Z_j^{(k)} + W_{j+1}^{(k)} > \lambda | X_j = 1] \\
&\quad + \Pr[X_j = 2] \cdot \Pr[\mathbb{1}\{X_j = 1\} \cdot \gamma_j \cdot Z_j^{(k)} + W_{j+1}^{(k)} > \lambda | X_j = 2] \\
&\leq \Pr[X_j = 0] \cdot \Pr[W_{j+1}^{(k)} > \lambda] \\
&\quad + \Pr[X_j = 1] \cdot \Pr[W_{j+1}^{(k)} > \lambda - \gamma_j] \\
&\quad + \Pr[X_j = 2] \cdot \Pr[W_{j+1}^{(k-\gamma_j)} > \lambda] \\
&\leq (1 - p_j - q_j) \cdot \exp\left(-\frac{\lambda}{5} + 3(k+1)\right) \\
&\quad + p_j \cdot \exp\left(-\frac{\lambda - \gamma_j}{5} + 3(k+1)\right) \\
&\quad + q_j \cdot \exp\left(-\frac{\lambda}{5} + 3(k - \gamma_j + 1)\right) \tag{5}
\end{aligned}
$$

$$= (1 - p_j - q_j) \cdot \exp\left(-\frac{\lambda}{5} + 3(k+1)\right)$$

$$+ p_j \cdot e^{\gamma_j/5} \cdot \exp\left(-\frac{\lambda}{5} + 3(k+1)\right)$$

$$+ q_j \cdot e^{-3\gamma_j} \cdot \exp\left(-\frac{\lambda}{5} + 3(k+1)\right)$$

$$= \left(1 + p_j\left(e^{\gamma_j/5} - 1\right) + q_j\left(e^{-3\gamma_j} - 1\right)\right) \cdot \exp\left(-\frac{\lambda}{5} + 3(k+1)\right)$$

$$\leq \left(1 + p_j\left(e^{\gamma_j/5} - 1\right) + \frac{p_j}{4}\left(e^{-3\gamma_j} - 1\right)\right) \cdot \exp\left(-\frac{\lambda}{5} + 3(k+1)\right)$$

$$= \left(1 + p_j\left(e^{\gamma_j/5} + \frac{e^{-3\gamma_j}}{4} - \frac{5}{4}\right)\right) \cdot \exp\left(-\frac{\lambda}{5} + 3(k+1)\right)$$

$$\leq \exp\left(-\frac{\lambda}{5} + 3(k+1)\right). \tag{6}$$

Inequality (6) follows from the fact that $e^{\gamma/5} + \frac{e^{-3\gamma}}{4} - \frac{5}{4} \leq 0$ for every $\gamma \in [0,1]$. Inequality (5) follows from the inductive assumption.[5]   ∎

We next show how Lemma 11 follows from Claim 22. Recall that Lemma 11 states that $\Pr[E_3] \geq 1 - 3\delta$, where $E_3 = E_1 \wedge E_2$. We analyze E_1 and E_2 separately.

Claim 23 $\Pr[E_1] \geq 1 - \delta$.

Proof To map algorithm ThresholdMonitor to the setting of Claim 22, recall that we denote $\Delta = \frac{1}{\varepsilon}\log\left(\frac{1}{\delta}\right)\log\left(\frac{1}{\varepsilon}\log\frac{1}{\delta}\right)$, and consider an execution of ThresholdMonitor. In every round i let $\gamma_i = f_i(x')$ and define a random variable X_i as follows:

1. If $a_i = \top$ then $X_i = 2$

2. If $a_i = \bot$ and $f_i(S_i) + w_i \geq t - 2\Delta$ then $X_i = 1$

3. Otherwise $X_i = 0$

To apply Claim 22 we need to show that for every i

$$\Pr[X_i = 1] \leq \frac{1}{2} \qquad \text{and} \qquad \Pr[X_i = 2] \geq \frac{1}{4} \cdot \Pr[X_i = 1].$$

---

5. For the edge case where $-1 \leq k - \gamma_j < 0$, in which we cannot apply the inductive assumption, observe that $W_{j+1}^{(k-\gamma_j)} \equiv 0$ and hence $\Pr[W_{j+1}^{(k-\gamma_j)} > \lambda] = 0$.

Indeed,

$$
\begin{aligned}
\Pr[X_i = 1] &= \Pr\left[a_i = \bot \text{ and } f_i(S_i) + w_i \geq t - 2\Delta\right] \\
&= \Pr\left[f_i(S_i) + w_i + \overline{v}_i < t \text{ and } f_i(S_i) + w_i \geq t - 2\Delta\right] \\
&= \Pr\left[t - f_i(S_i) - 2\Delta \leq w_i < t - f_i(S_i) - \overline{v}_i\right] \\
&\leq \Pr[v_i < -\Delta] + \Pr\left[t - f_i(S_i) - 2\Delta \leq w_i < t - f_i(S_i) - \overline{v}_i | v_i \geq -\Delta\right] \\
&\leq \frac{1}{4} + \Pr\left[t - f_i(S_i) - 2\Delta \leq w_i < t - f_i(S_i) + \Delta\right] \\
&\leq \frac{1}{4} + \frac{1}{4} = \frac{1}{2},
\end{aligned}
$$

where the last inequality follows from the fact that w_i is sampled from $\mathrm{Lap}(10\Delta)$. Also,

$$
\begin{aligned}
\Pr[X_i = 2] &\geq \Pr[f_i(S_i) + w_i \geq t - 2\Delta] \cdot \Pr[X_i = 2 | f_i(S_i) + w_i \geq t - 2\Delta] \\
&\geq \Pr[X_i = 1] \cdot \Pr[X_i = 2 | f_i(S_i) + w_i \geq t - 2\Delta] \\
&= \Pr[X_i = 1] \cdot \Pr[a_i = \top | f_i(S_i) + w_i \geq t - 2\Delta] \\
&= \Pr[X_i = 1] \cdot \Pr[f_i(S_i) + w_i + \overline{v}_i \geq t | f_i(S_i) + w_i \geq t - 2\Delta] \\
&\geq \Pr[X_i = 1] \cdot \Pr[v_i \geq -\Delta] \cdot \Pr[f_i(S_i) + w_i + \overline{v}_i \geq t | f_i(S_i) + w_i \geq t - 2\Delta \text{ and } v_i \geq -\Delta] \\
&\geq \Pr[X_i = 1] \cdot \Pr[v_i \geq -\Delta] \cdot \Pr[w_i \geq t - f_i(S_i) + \Delta | w_i \geq t - f_i(S_i) - 2\Delta] \\
&\geq \Pr[X_i = 1] \cdot \frac{1}{2} \cdot \frac{1}{2} = \Pr[X_i = 1] \cdot \frac{1}{4},
\end{aligned}
$$

where the last inequality follows again from the fact that w_i is sampled from $\mathrm{Lap}(10\Delta)$. Now, let i^* denote the first time step in which $k \leq \sum_{i=1}^{i^*} \mathbb{1}\{X_i = 2\} \cdot \gamma_i = \sum_{i=1}^{i^*} \mathbb{1}\{a_i = \top\} \cdot f_i(x')$. Also let

$$
W = \sum_{i=1}^{i^*} \mathbb{1}\{X_i = 1\} \cdot \gamma_i = \sum_{i=1}^{i^*} \mathbb{1}\{i \text{ is an almost-top}\} \cdot f_i(x').
$$

By Claim 22 we have that

$$
\Pr[\,\overline{E_1}\,] \leq \Pr\left[W \geq 5 \cdot \log\frac{1}{\delta} + 15(k+1)\right] \leq \delta.
$$

∎

Claim 24 $\Pr[E_2] \geq 1 - 2\delta$.

Proof Consider an execution of `ThresholdMonitor` on a database S, and fix a possible outcome vector $\vec{a} = (a_1, a_2, \ldots, a_m)$. Recall that $\vec{a}$ determines all of the queries $f_{\vec{a},i}$ throughout the execution. Note, however, that fixing $\vec{a}$ does not completely determine the noisy values w_i throughout the execution. (Rather, for every query i it only determines whether $\hat{f}_i < t$ or not.) Let $F_1, F_2, \ldots, F_m \in \mathbb{R}$ be random variables denoting the values of $w_1, w_2, \ldots, w_m$, respectively. Observe that, conditioned on $\vec{a}$, we have that $F_1, F_2, \ldots, F_m$ are independent. In addition, for every fixture I of the time steps in which an almost-top occurs, we have that $F_1, F_2, \ldots, F_m$ are conditionally independent given $\vec{a}$ and I.

Now, let $G_1, G_2, \ldots, G_m \in \{0, 1\}$ be random variables defined as $G_i = 1$ if a special-almost-top occurs in time i. As each G_i is a function (only) of F_i, we have that $G_1, G_2, \ldots, G_m$ are also conditionally independent given $\vec{a}$ and I. Observe that $\mathbb{E}\left[\sum_i G_i \mid \vec{a}, I\right] \leq \sum_{i \in I} f_{\vec{a},i}(x')$, because every w_i is sampled from $\mathrm{Lap}(10\Delta)$, and hence, even conditioned on i being an almost-top, the probability that w_i falls inside an interval of length $f_{\vec{a},i}(x')$ is at most $f_{\vec{a},i}(x')$. Therefore,

$$
\Pr[E_2] = \Pr\left[\sum_{i=1}^{m} G_i \leq 30(k+1) + 10\log\frac{1}{\delta}\right]
$$

$$
= \sum_{\vec{a},I} \Pr[\vec{a}, I] \cdot \Pr\left[\sum_{i=1}^{m} G_i \leq 30(k+1) + 10\log\frac{1}{\delta}\,\middle|\, \vec{a}, I\right]
$$

$$
\geq \sum_{\vec{a},I} \Pr[\vec{a}, I] \cdot \Pr\left[E_1 \wedge \sum_{i=1}^{m} G_i \leq 30(k+1) + 10\log\frac{1}{\delta}\,\middle|\, \vec{a}, I\right]
$$

$$
= \sum_{\vec{a},I} \Pr[\vec{a}, I] \cdot \Pr\left[E_1 \wedge \sum_{i=1}^{m} G_i \leq 2 \cdot \max\left\{\sum_{i \in I} f_{\vec{a},i}(x'),\ 15(k+1) + 5\log\frac{1}{\delta}\right\}\,\middle|\, \vec{a}, I\right]
\tag{7}
$$

$$
= \sum_{\vec{a},I} \Pr[\vec{a}, I] \cdot \Pr\left[E_1 \wedge \sum_{i=1}^{m} G_i \leq 2 \cdot \max\left\{\mathbb{E}\left[\sum G_i \,\middle|\, \vec{a}, I\right],\ 15(k+1) + 5\log\frac{1}{\delta}\right\}\,\middle|\, \vec{a}, I\right]
\tag{8}
$$

$$
\geq \sum_{\vec{a},I} \Pr[\vec{a}, I] \cdot \Pr\left[\sum_{i=1}^{m} G_i \leq 2 \cdot \max\left\{\mathbb{E}\left[\sum G_i \,\middle|\, \vec{a}, I\right],\ 15(k+1) + 5\log\frac{1}{\delta}\right\}\,\middle|\, \vec{a}, I\right] - \delta
\tag{9}
$$

$$
\geq \sum_{\vec{a},I} \Pr[\vec{a}, I] \cdot (1 - \delta) - \delta
\tag{10}
$$

$$
= 1 - 2\delta.
$$

Equality (7) holds since whenever E_1 occurs we have that $\sum_{i \in I} f_{\vec{a},i}(x') \leq 15(k+1) + 5\log\frac{1}{\delta}$. Equality (8) holds since $\mathbb{E}\left[\sum_i G_i \mid \vec{a}, I\right] \leq \sum_{i \in I} f_{\vec{a},i}(x')$. Inequality (9) holds since $\Pr[E_1] \geq 1 - \delta$. Inequality (10) follows from the Chernoff bound. ∎

Lemma 11 now follows by combining Claim 23 and Claim 24.

Appendix B. Proof of Lemma 13

In this section we restate and prove Lemma 13.

Lemma 25 (restating Lemma 13). *The following holds for every outcome vector $\vec{a} \in \{\bot, \top\}^*$ and noise vector $W = (w_1, w_2, \ldots)$ for which event E_3 occurs. Let $i^* = i^*(\vec{a})$ denote the first time step such that $c_{i^*}(x') \geq k$. Let $I_{\mathrm{top}} = I_{\mathrm{top}}(\vec{a})$ denote the set of all time steps $i \leq i^*$ such that $a_i = \top$. Let W' denote a vector identical to W, except that every coordinate $i \in I_{\mathrm{top}}$ is decreased*

by $f_{\vec{a},i}(x')$. *That is, $w_i' = w_i - f_{\vec{a},i}(x')$ for every $i \in I_{\text{top}}$, and $w_i' = w_i$ for every other index i. Then,*

$$\Pr\left[\mathcal{B}(S') = \vec{a} \,|\, W'\right] \;\leq\; \Pr\left[\mathcal{B}(S) = \vec{a} \,|\, W\right] \;\leq\; \exp\left(\frac{75(k+1)\varepsilon}{\log\frac{1}{\delta}} + 25\varepsilon\right) \cdot \Pr\left[\mathcal{B}(S') = \vec{a} \,|\, W\right].$$

(Note that since we fixed W and W', these probabilities are only over the noises v_i.)

Proof of Lemma 13 Fix an outcome vector $\vec{a}$ and a vector of noises $W = (w_1, w_2, \dots)$ for which Event E_3 occurs. Let $i^* = i^*(\vec{a})$ denote the first time step such that $c_{i^*}(x') \geq k$. This is the time step in which x' is deleted from the data. Denote $\Delta = \frac{1}{\varepsilon} \log\left(\frac{1}{\delta}\right) \log\left(\frac{1}{\varepsilon} \log\frac{1}{\delta}\right)$. Consider the following partition of the time steps (this partition depends on $\vec{a}$ and W):

I_{top}	All indices $i \leq i^*$ such that $a_i = \top$
I_{after}	All indices $i > i^*$
I_{far}	All indices $i \leq i^*$ such that $a_i = \bot$ and $f_{\vec{a},i}(S) + w_i < t - f_{\vec{a},i}(x') - \Delta$
$I_{\text{special-almost}}$	All indices $i \leq i^*$ such that $a_i = \bot$ and $t - f_{\vec{a},i}(x') - \Delta \leq f_{\vec{a},i}(S_i) + w_i < t - \Delta$
$I_{\text{upper-almost}}$	All indices $i \leq i^*$ such that $a_i = \bot$ and $f_{\vec{a},i}(S) + w_i \geq t - \Delta$

Recall that we write W' to denote a vector identical to W, except that coordinates $i \in I_{\text{top}}$ are decreased by $f_{\vec{a},i}(x')$. For convenience, let $\mathcal{B} = \mathcal{B}(S)$ and $\mathcal{B}' = \mathcal{B}(S')$. We may now decompose

$$\begin{aligned}
\Pr\left[\mathcal{B}(S) = \vec{a} \,|\, W\right] &= \Pr\left[(\mathcal{B}_1, \mathcal{B}_2, \dots) = (a_1, a_2, \dots) \,|\, W\right] \\
&= \prod_{i=1,2,\dots} \Pr\left[\mathcal{B}_i = a_i \,|\, (\mathcal{B}_1, \dots, \mathcal{B}_{i-1}) = (a_1, \dots, a_{i-1}) \wedge W\right] \\
&= \prod_{i \in I_{\text{top}}} \Pr\left[\mathcal{B}_i = a_i \,|\, (\mathcal{B}_1, \dots, \mathcal{B}_{i-1}) = (a_1, \dots, a_{i-1}) \wedge W\right] \\
&\quad \cdot \prod_{i \in I_{\text{after}}} \Pr\left[\mathcal{B}_i = a_i \,|\, (\mathcal{B}_1, \dots, \mathcal{B}_{i-1}) = (a_1, \dots, a_{i-1}) \wedge W\right] \\
&\quad \cdot \prod_{i \in I_{\text{far}}} \Pr\left[\mathcal{B}_i = a_i \,|\, (\mathcal{B}_1, \dots, \mathcal{B}_{i-1}) = (a_1, \dots, a_{i-1}) \wedge W\right] \\
&\quad \cdot \prod_{i \in I_{\text{special-almost}}} \Pr\left[\mathcal{B}_i = a_i \,|\, (\mathcal{B}_1, \dots, \mathcal{B}_{i-1}) = (a_1, \dots, a_{i-1}) \wedge W\right] \\
&\quad \cdot \prod_{i \in I_{\text{upper-almost}}} \Pr\left[\mathcal{B}_i = a_i \,|\, (\mathcal{B}_1, \dots, \mathcal{B}_{i-1}) = (a_1, \dots, a_{i-1}) \wedge W\right] \quad (11)
\end{aligned}$$

We next analyze each of the expressions in Equality (11) separately, and relate them to the corresponding expressions with S' instead of S. Recall that we need to bound these expressions both from above and from below.

Analysis for $i \in I_{\text{top}}$. Recall that $S' = S \cup \{x'\}$, and that $f_{\vec{a},i}(S'_i) = f_{\vec{a},i}(S_i) + f_{\vec{a},i}(x')$ for every $i \in I_{\text{top}}$. Therefore, for every $i \in I_{\text{top}}$ we have

$$
\Pr\left[\mathcal{B}_i = a_i \,|\, (\mathcal{B}_1, \ldots, \mathcal{B}_{i-1}) = (a_1, \ldots, a_{i-1}) \wedge W\right]
$$
$$
= \Pr\left[\mathcal{B}_i = \top \,|\, (\mathcal{B}_1, \ldots, \mathcal{B}_{i-1}) = (a_1, \ldots, a_{i-1}) \wedge W\right]
$$
$$
= \Pr_{v_i}\left[f_{\vec{a},i}(S_i) + w_i + \overline{v}_i \geq t\right]
$$
$$
= \Pr_{v_i}\left[f_{\vec{a},i}(S'_i) - f_{\vec{a},i}(x') + w_i + \overline{v}_i \geq t\right]
$$
$$
= \Pr\left[\mathcal{B}'_i = a_i \,|\, (\mathcal{B}'_1, \ldots, \mathcal{B}'_{i-1}) = (a_1, \ldots, a_{i-1}) \wedge W'\right],
$$

Observe that one of the directions holds also without replacing W with W'. Specifically,

$$
\Pr\left[\mathcal{B}_i = a_i \,|\, (\mathcal{B}_1, \ldots, \mathcal{B}_{i-1}) = (a_1, \ldots, a_{i-1}) \wedge W\right]
$$
$$
= \Pr\left[\mathcal{B}_i = \top \,|\, (\mathcal{B}_1, \ldots, \mathcal{B}_{i-1}) = (a_1, \ldots, a_{i-1}) \wedge W\right]
$$
$$
= \Pr_{v_i}\left[f_{\vec{a},i}(S_i) + w_i + \overline{v}_i \geq t\right]
$$
$$
\leq \Pr_{v_i}\left[f_{\vec{a},i}(S'_i) + w_i + \overline{v}_i \geq t\right]
$$
$$
= \Pr\left[\mathcal{B}'_i = a_i \,|\, (\mathcal{B}'_1, \ldots, \mathcal{B}'_{i-1}) = (a_1, \ldots, a_{i-1}) \wedge W\right].
$$

Analysis for $i \in I_{\text{after}}$. Recall that during the execution on S' we have that x' is deleted from the input database during time $i = i^*$. Therefore, for every $i > i^*$ we have that $S_i = S'_i$. Hence,

$$
\prod_{i \in I_{\text{after}}} \Pr\left[\mathcal{B}_i = a_i \,|\, (\mathcal{B}_1, \ldots, \mathcal{B}_{i-1}) = (a_1, \ldots, a_{i-1}) \wedge W\right]
$$

$$
= \left(\prod_{\substack{i \in I_{\text{after}} \\ a_i = \bot}} \Pr_{v_i}\left[f_{\vec{a},i}(S_i) + w_i + \overline{v}_i < t\right]\right) \cdot \left(\prod_{\substack{i \in I_{\text{after}} \\ a_i = \top}} \Pr_{v_i}\left[f_{\vec{a},i}(S_i) + w_i + \overline{v}_i \geq t\right]\right)
$$

$$
= \left(\prod_{\substack{i \in I_{\text{after}} \\ a_i = \bot}} \Pr_{v_i}\left[f_{\vec{a},i}(S'_i) + w_i + \overline{v}_i < t\right]\right) \cdot \left(\prod_{\substack{i \in I_{\text{after}} \\ a_i = \top}} \Pr_{v_i}\left[f_{\vec{a},i}(S'_i) + w_i + \overline{v}_i \geq t\right]\right)
$$

$$
= \prod_{i \in I_{\text{after}}} \Pr\left[\mathcal{B}'_i = a_i \,|\, (\mathcal{B}'_1, \ldots, \mathcal{B}'_{i-1}) = (a_1, \ldots, a_{i-1}) \wedge W\right]
$$

$$
= \prod_{i \in I_{\text{after}}} \Pr\left[\mathcal{B}'_i = a_i \,|\, (\mathcal{B}'_1, \ldots, \mathcal{B}'_{i-1}) = (a_1, \ldots, a_{i-1}) \wedge W'\right],
$$

where the last equality is because once we fix $(a_1, \ldots, a_{i-1})$ and w_i then $\mathcal{B}'_i$ is independent of w_j for $j \neq i$ (it depends only on v_i).

Analysis for $i \in I_{\text{far}}$. For every $i \in I_{\text{far}}$ we have that $f_{\vec{a},i}(S_i) + w_i < t - f_{\vec{a},i}(x') - \Delta$. Therefore, we also have that
$$
f_{\vec{a},i}(S'_i) + w_i = f_{\vec{a},i}(S_i) + f_{\vec{a},i}(x') + w_i < t - \Delta.
$$

Recall that, by definition, $\overline{v}_i$ is always at most Δ. Hence,

$$\prod_{i \in I_{\text{far}}} \Pr\left[\mathcal{B}_i = a_i \,|\, (\mathcal{B}_1, \ldots, \mathcal{B}_{i-1}) = (a_1, \ldots, a_{i-1}) \wedge W\right]$$

$$= \prod_{i \in I_{\text{far}}} \Pr\left[\mathcal{B}_i = \bot \,|\, (\mathcal{B}_1, \ldots, \mathcal{B}_{i-1}) = (a_1, \ldots, a_{i-1}) \wedge W\right]$$

$$= \prod_{i \in I_{\text{far}}} \Pr_{v_i}\left[f_{\vec{a},i}(S_i) + w_i + \overline{v}_i < t\right]$$

$$= 1$$

$$= \prod_{i \in I_{\text{far}}} \Pr_{v_i}\left[f_{\vec{a},i}(S_i') + w_i + \overline{v}_i < t\right]$$

$$= \prod_{i \in I_{\text{far}}} \Pr\left[\mathcal{B}_i' = a_i \,\big|\, (\mathcal{B}_1', \ldots, \mathcal{B}_{i-1}') = (a_1, \ldots, a_{i-1}) \wedge W\right]$$

$$= \prod_{i \in I_{\text{far}}} \Pr\left[\mathcal{B}_i' = a_i \,\big|\, (\mathcal{B}_1', \ldots, \mathcal{B}_{i-1}') = (a_1, \ldots, a_{i-1}) \wedge W'\right].$$

Analysis for $i \in I_{\text{special-almost}}$. First observe that

$$\prod_{i \in I_{\text{special-almost}}} \Pr\left[\mathcal{B}_i = a_i \,|\, (\mathcal{B}_1, \ldots, \mathcal{B}_{i-1}) = (a_1, \ldots, a_{i-1}) \wedge W\right]$$

$$= \prod_{i \in I_{\text{special-almost}}} \Pr\left[\mathcal{B}_i = \bot \,|\, (\mathcal{B}_1, \ldots, \mathcal{B}_{i-1}) = (a_1, \ldots, a_{i-1}) \wedge W\right]$$

$$= \prod_{i \in I_{\text{special-almost}}} \Pr_{v_i}\left[f_{\vec{a},i}(S_i) + w_i + \overline{v}_i < t\right]$$

$$\geq \prod_{i \in I_{\text{special-almost}}} \Pr_{v_i}\left[f_{\vec{a},i}(S_i') + w_i + \overline{v}_i < t\right]$$

$$= \prod_{i \in I_{\text{special-almost}}} \Pr\left[\mathcal{B}_i' = a_i \,\big|\, (\mathcal{B}_1', \ldots, \mathcal{B}_{i-1}') = (a_1, \ldots, a_{i-1}) \wedge W\right]$$

$$= \prod_{i \in I_{\text{special-almost}}} \Pr\left[\mathcal{B}_i' = a_i \,\big|\, (\mathcal{B}_1', \ldots, \mathcal{B}_{i-1}') = (a_1, \ldots, a_{i-1}) \wedge W'\right].$$

For the other direction, fix $i \in I_{\text{special-almost}}$. Recall that v_i is sampled from $\text{Lap}(\frac{1}{\varepsilon} \log \frac{1}{\delta})$, and that we denote $\Delta = \frac{1}{\varepsilon} \log\left(\frac{1}{\delta}\right) \log\left(\frac{1}{\varepsilon} \log \frac{1}{\delta}\right)$. Using the fact that $e^{\gamma} \cdot \left(1 - \frac{\gamma}{2}\right) \geq 1$ for every $0 \leq \gamma \leq 1$ we have that

$$\Pr\left[\mathcal{B}_i = a_i \,\middle|\, (\mathcal{B}_1, \ldots, \mathcal{B}_{i-1}) = (a_1, \ldots, a_{i-1}) \wedge W\right]$$
$$\leq 1$$
$$\leq e^{\varepsilon / \log \frac{1}{\delta}} \cdot \left(1 - \frac{\varepsilon}{2 \log \frac{1}{\delta}}\right)$$
$$\leq e^{\varepsilon / \log \frac{1}{\delta}} \cdot \Pr_{v_i} \left[v_i < \Delta\right]$$
$$\leq e^{\varepsilon / \log \frac{1}{\delta}} \cdot \Pr_{v_i} \left[f_{\vec{a},i}(S_i) + w_i + v_i < t\right] \tag{12}$$
$$\leq e^{\varepsilon / \log \frac{1}{\delta}} \cdot e^{\varepsilon / \log \frac{1}{\delta}} \cdot \Pr_{v_i} \left[f_{\vec{a},i}(S_i') + w_i + v_i < t\right]$$
$$\leq e^{2\varepsilon / \log \frac{1}{\delta}} \cdot \Pr_{v_i} \left[f_{\vec{a},i}(S_i') + w_i + \overline{v}_i < t\right]$$
$$= e^{2\varepsilon / \log \frac{1}{\delta}} \cdot \Pr\left[\mathcal{B}_i' = a_i \,\middle|\, (\mathcal{B}_1', \ldots, \mathcal{B}_{i-1}') = (a_1, \ldots, a_{i-1}) \wedge W\right],$$

where (12) follows since for $i \in I_{\text{special-almost}}$ we have that $f_{\vec{a},i}(S_i) + w_i + \Delta < t$, and hence, if $v_i < \Delta$ then $f_{\vec{a},i}(S_i) + w_i + v_i < t$. Now, by Event E_2 we have that $|I_{\text{special-almost}}| \leq 30(k+1) + 10 \log \frac{1}{\delta}$, and therefore,

$$\prod_{i \in I_{\text{special-almost}}} \Pr\left[\mathcal{B}_i = a_i \,\middle|\, (\mathcal{B}_1, \ldots, \mathcal{B}_{i-1}) = (a_1, \ldots, a_{i-1}) \wedge W\right]$$
$$\leq \prod_{i \in I_{\text{special-almost}}} e^{2\varepsilon / \log \frac{1}{\delta}} \cdot \Pr\left[\mathcal{B}_i' = a_i \,\middle|\, (\mathcal{B}_1', \ldots, \mathcal{B}_{i-1}') = (a_1, \ldots, a_{i-1}) \wedge W\right]$$
$$\leq \exp\left(\frac{60(k+1)\varepsilon}{\log \frac{1}{\delta}} + 20\varepsilon\right) \cdot \prod_{i \in I_{\text{special-almost}}} \Pr\left[\mathcal{B}_i' = a_i \,\middle|\, (\mathcal{B}_1', \ldots, \mathcal{B}_{i-1}') = (a_1, \ldots, a_{i-1}) \wedge W\right].$$

Analysis for $i \in I_{\text{upper-almost}}$. As in the analysis for $i \in I_{\text{special-almost}}$, we have that

$$\prod_{i \in I_{\text{upper-almost}}} \Pr\left[\mathcal{B}_i = a_i \,\middle|\, (\mathcal{B}_1, \ldots, \mathcal{B}_{i-1}) = (a_1, \ldots, a_{i-1}) \wedge W\right]$$
$$\geq \prod_{i \in I_{\text{upper-almost}}} \Pr\left[\mathcal{B}_i' = a_i \,\middle|\, (\mathcal{B}_1', \ldots, \mathcal{B}_{i-1}') = (a_1, \ldots, a_{i-1}) \wedge W\right]$$
$$= \prod_{i \in I_{\text{upper-almost}}} \Pr\left[\mathcal{B}_i' = a_i \,\middle|\, (\mathcal{B}_1', \ldots, \mathcal{B}_{i-1}') = (a_1, \ldots, a_{i-1}) \wedge W'\right].$$

For the other direction,

$$\prod_{i \in I_{\text{upper-almost}}} \Pr\left[\mathcal{B}_i = a_i \,\middle|\, (\mathcal{B}_1, \ldots, \mathcal{B}_{i-1}) = (a_1, \ldots, a_{i-1}) \wedge W\right]$$

$$= \prod_{i \in I_{\text{upper-almost}}} \Pr\left[\mathcal{B}_i = \bot \,\middle|\, (\mathcal{B}_1, \ldots, \mathcal{B}_{i-1}) = (a_1, \ldots, a_{i-1}) \wedge W\right]$$

$$= \prod_{i \in I_{\text{upper-almost}}} \Pr_{v_i}\left[f_{\vec{a},i}(S_i) + w_i + \overline{v}_i < t\right]$$

$$= \prod_{i \in I_{\text{upper-almost}}} \Pr\left[f_{\vec{a},i}(S_i) + w_i + v_i < t\right] \tag{13}$$

$$\leq \prod_{i \in I_{\text{upper-almost}}} e^{f_{\vec{a},i}(x') \cdot \varepsilon / \log \frac{1}{\delta}} \cdot \Pr\left[f_{\vec{a},i}(S_i) + f_{\vec{a},i}(x') + w_i + v_i < t\right]$$

$$= \prod_{i \in I_{\text{upper-almost}}} e^{f_{\vec{a},i}(x') \cdot \varepsilon / \log \frac{1}{\delta}} \cdot \Pr\left[f_{\vec{a},i}(S_i') + w_i + v_i < t\right]$$

$$\leq \prod_{i \in I_{\text{upper-almost}}} e^{f_{\vec{a},i}(x') \cdot \varepsilon / \log \frac{1}{\delta}} \cdot \Pr\left[f_{\vec{a},i}(S_i') + w_i + \overline{v}_i < t\right]$$

$$= \prod_{i \in I_{\text{upper-almost}}} e^{f_{\vec{a},i}(x') \cdot \varepsilon / \log \frac{1}{\delta}} \cdot \Pr\left[\mathcal{B}_i' = a_i \,\middle|\, (\mathcal{B}_1', \ldots, \mathcal{B}_{i-1}') = (a_1, \ldots, a_{i-1}) \wedge W\right]$$

$$= \exp\left(\sum_{i \in I_{\text{upper-almost}}} \frac{f_{\vec{a},i}(x') \cdot \varepsilon}{\log(1/\delta)}\right) \cdot \prod_{i \in I_{\text{upper-almost}}} \Pr\left[\mathcal{B}_i' = a_i \,\middle|\, (\mathcal{B}_1', \ldots, \mathcal{B}_{i-1}') = (a_1, \ldots, a_{i-1}) \wedge W\right]$$

$$\leq \exp\left(\frac{15\varepsilon(k+1)}{\log(1/\delta)} + 5\varepsilon\right) \cdot \prod_{i \in I_{\text{upper-almost}}} \Pr\left[\mathcal{B}_i' = a_i \,\middle|\, (\mathcal{B}_1', \ldots, \mathcal{B}_{i-1}') = (a_1, \ldots, a_{i-1}) \wedge W\right],$$

$$\tag{14}$$

where (14) follows from Event E_1, and where (13) holds since $\overline{v}_i = \min\{v_i, \Delta\}$ and since for $i \in I_{\text{upper-almost}}$ we have $f_{\vec{a},i}(S) + w_i + \Delta \geq t$. Hence, if $f_{\vec{a},i}(S_i) + w_i + \overline{v}_i < t$ then $f_{\vec{a},i}(S_i) + w_i + v_i < t$.

This concludes the analysis of the different expressions in Equality (11). Returning to Equality (11), we now get that

$$\Pr\left[\mathcal{B}(S') = \vec{a} \,\middle|\, W'\right] \leq \Pr\left[\mathcal{B}(S) = \vec{a} \,\middle|\, W\right] \leq \exp\left(\frac{75(k+1)\varepsilon}{\log \frac{1}{\delta}} + 25\varepsilon\right) \cdot \Pr\left[\mathcal{B}(S') = \vec{a} \,\middle|\, W\right].$$

This completes the proof of Lemma 13. ∎

Appendix C. Proof of Theorem 14

In this section we prove Theorem 14. We begin with a short overview of the proof (the formal proof is given afterwards).

Proof overview. Fix two neighboring databases S, S'. We want to show that the outcome distributions of `ThresholdMonitor`(S) and `ThresholdMonitor`(S') are similar. To that end, we "break" algorithm `ThresholdMonitor` into several "smaller" components, analyze the privacy guarantees of each component, and then argue about the overall privacy guarantees of `ThresholdMonitor` using composition theorems. Technically, the definition of these components depends on S and S', but this still allows the argument to go through. These components should not be thought of as standalone algorithms (they only exist as part of the proof of Theorem 14).

Before presenting the formal proof, we state the necessary preliminaries regarding adaptive composition for differential privacy (Dwork et al., 2010). Let ε, δ be parameters. For $b \in \{0, 1\}$, define the following ℓ-round game against an adversary A, denoted as `Experiment`$^b_{\ell, \varepsilon, \delta}(A)$.

`Experiment`$^b_{\ell, \varepsilon, \delta}(A)$

    1. For $i = 1, \ldots, \ell$:

        (a) The adversary A outputs two distributions $\mathcal{D}_i^0$ and $\mathcal{D}_i^1$ such that $\mathcal{D}_i^0 \approx_{(\varepsilon, \delta)} \mathcal{D}_i^1$.

        (b) The adversary receives a sample $y_i \sim \mathcal{D}_i^b$.

    2. Output $(y_1, \ldots, y_\ell)$ and the internal randomness of the adversary A.

In every round of this experiment, the adversary specifies two (similar) distributions $\mathcal{D}_i^0$ and $\mathcal{D}_i^1$ and gets a sample from $\mathcal{D}_i^b$. Intuitively, the adversary's goal is to guess the bit b. However, since it is restricted to choose distributions $\mathcal{D}_i^0$ and $\mathcal{D}_i^1$ such that $\mathcal{D}_i^0 \approx_{(\varepsilon, \delta)} \mathcal{D}_i^1$, then its ability to guess the bit b is limited. Formally, the following theorem states that A's view of the experiment (and in particular, A's guess for b) is distributed roughly the same for both values of b.

Theorem 26 (Dwork et al. (2010)). *For every $\varepsilon, \delta, \hat{\delta} \geq 0$, every $\ell \in \mathbb{N}$, and every adversary A it holds that*

$$Experiment^0_{\ell, \varepsilon, \delta}(A) \approx_{(\hat{\varepsilon}, \ell\delta + \hat{\delta})} Experiment^1_{\ell, \varepsilon, \delta}(A),$$

where $\hat{\varepsilon} = \sqrt{2\ell \ln\left(\frac{1}{\hat{\delta}}\right)} \cdot \varepsilon + \ell\varepsilon(e^\varepsilon - 1)$.

We are now ready to present the proof of Theorem 14.

Proof of Theorem 14 Fix two neighboring databases S and $S' = S \cup \{x'\}$ for some $x' \in X$. We partition the execution of `ThresholdMonitor` into $\ell = k/\log\frac{1}{\delta}$ epochs based on the value of the counter $c(x')$, where x' is the data element that appears in S' but not in S. Informally, during each epoch, the value of $c(x')$ lies within an interval of length $\log\frac{1}{\delta}$, which ensures that the outcome distribution of each epoch is similar when executing on S or on S' (as in Section 3). Theorem 14 then follows from Theorem 26, since `ThresholdMonitor` can be viewed as the composition of these epochs.

We now give the formal details. Consider algorithm `TM`$_{x'}$ (presented in Algorithm 4). This algorithm is similar to algorithm `ThresholdMonitor`, except that the counters $c(\cdot)$ are initialized differently. (As we mentioned, this algorithm should not be thought of as a standalone algorithm; it is only part of the analysis.) Observe that in `TM`$_{x'}$ we initialize $c(x') \geq k - \log\frac{1}{\delta}$, and that x'

Algorithm 4 $\mathtt{TM}_{x'}$

Input: Database $D \in X^*$, privacy parameters ε, δ, threshold t, parameter k, counter values $c : X \to \mathbb{R}$, and an adaptively chosen stream of counting queries $f_i : X \to [0, 1]$.

1. Set $c(x') \leftarrow \max \left\{ c(x'), \ k - \log \frac{1}{\delta} \right\}$. Delete from D every element x such that $c(x) \geq k$.

2. Denote $\Delta = \frac{1}{\varepsilon} \log \left(\frac{1}{\delta} \right) \log \left(\frac{1}{\varepsilon} \log \frac{1}{\delta} \right)$.

3. In each round i, when receiving a query f_i, do the following:

   (a) Let $w_i \leftarrow \mathrm{Lap}\,(10\Delta)$ and let $v_i \leftarrow \mathrm{Lap}(\frac{1}{\varepsilon} \log \frac{1}{\delta})$.

   (b) Denote $\overline{v}_i = \min \{ v_i, \ \Delta \}$.

   (c) Let $\hat{f}_i = f_i(D) + w_i + \overline{v}_i$.

   (d) If $\hat{f}_i < t$, then output $a_i = \bot$.

   (e) Otherwise:
   - Output $a_i = \top$.
   - For every $x \in X$ set $c(x) \leftarrow c(x) + f_i(x)$.
   - Delete from D every element x such that $c(x) \geq k$.

   (f) Proceed to the next iteration.

---

is deleted from the data once $c(x') \geq k$. Hence, an identical analysis to that of Section 3 shows that for every neighboring databases that differ on x', say D and $D' = D \cup \{x'\}$, we have that the outcome distribution of $\mathtt{TM}_{x'}(D)$ and of $\mathtt{TM}_{x'}(D')$ are similar. Specifically, for every adversary $\widetilde{A}$ interacting with $\mathtt{TM}_{x'}$ we have that

$$\widetilde{\mathcal{B}}(D) \approx_{(O(\varepsilon), 3\delta)} \widetilde{\mathcal{B}}(D'),$$

where $\widetilde{\mathcal{B}}$ denotes the function that simulates the interaction and returns the transcript (cf. algorithm 1). Fix an adversary A that interacts with $\mathtt{ThresholdMonitor}$, and let $\mathcal{B} : X^n \to \{\bot, \top\}^*$ denote the function that simulates the interaction. We need to show that $\mathcal{B}(S) \approx_{(\varepsilon_0, 3k\delta)} \mathcal{B}(S')$.

We now reformulate algorithm $\mathcal{B}$ (ie., the interaction between $\mathtt{ThresholdMonitor}$ and the adversary A) as $\ell = k / \log \frac{1}{\delta}$ applications of algorithm $\mathtt{TM}_{x'}$. Specifically, consider algorithm $\mathtt{TailoredComposition}$, defined below.

Now observe that $\mathtt{TailoredComposition}$ is identical to $\mathcal{B}$, that is, to the function that simulates $\mathtt{ThresholdMonitor}$ (with parameter k) interacting with the adversary A. Specifically, algorithm $\mathtt{TailoredComposition}$ keeps track of the counters $\overline{c}$ in a way that is identical to the values of these counters in an execution of $\mathtt{ThresholdMonitor}$, and these counters are given to algorithm $\mathtt{TM}_{x'}(T)$ when it is re-executed. Therefore, in order to show that $\mathcal{B}(S) \approx_{(\varepsilon_0, 3k\delta)} \mathcal{B}(S')$ it suffices to show that

$$\mathtt{TailoredComposition}(S) \approx_{(\varepsilon_0, 3k\delta)} \mathtt{TailoredComposition}(S').$$

This follows by mapping algorithm $\mathtt{TailoredComposition}$ to the settings of Theorem 26 (the composition theorem). Specifically, for $b \in \{0, 1\}$ and $g \in \{1, 2, \ldots, \ell\}$, let z_g^b denote the portion of the output vector $(q_1, a_1, q_2, a_2, \ldots)$ that corresponds to time steps in which $G = g$ during the execution of $\mathtt{TailoredComposition}$ on S_b, where $S_0 = S$ and $S_1 = S'$. Also, for $g \in$

Algorithm `TailoredComposition`

Input: Database T (where T is either S or S')

1. Let $G = 1$, and instantiate a counter $\bar{c}(x) = 0$ for every $x \in X$.

2. Instantiate algorithm $\text{TM}_{x'}$ on the database T with the counters $\bar{c}$.

3. For $i = 1, 2, \ldots$ do

    (a) Obtain the next query f_i from the adversary A.

    (b) Give f_i to algorithm $\text{TM}_{x'}$ and obtain an answer a_i.

    (c) Give a_i to the adversary A.

    (d) If $a_i = \top$ set $\bar{c}(x) \leftarrow \bar{c} + f_i(x)$ for every $x \in X$.

    (e) If $\bar{c}(x') \geq G \cdot \log \frac{1}{\delta}$ and $\bar{c}(x') < k$ then
        - Set $G \leftarrow G + 1$.
        - End the current execution of $\text{TM}_{x'}$.
        - Re-instantiate $\text{TM}_{x'}(T)$ with the counters $\bar{c}$.

4. When A halts, output $z = (q_1, a_1, q_2, a_2, \ldots)$.

$\{1, 2, \ldots, \ell\}$ let Z_g^b be a random variable representing the value of z_g^b. Every such Z_g^b is the outcome of the interaction between the adversary and a single execution of algorithm $\text{TM}_{x'}(S_b)$. Therefore, for every $g \in \{1, 2, \ldots, \ell\}$ we have that $Z_g^0 \approx_{(O(\varepsilon), 3\delta)} Z_g^1$. Thus, by Theorem 26 we get that

$$\texttt{TailoredComposition}(S) = (Z_1^0, \ldots, Z_\ell^0) \approx_{(\varepsilon_0, 3k\delta)} (Z_1^1, \ldots, Z_\ell^1) = \texttt{TailoredComposition}(S'),$$

for $\varepsilon_0 = O\left(\sqrt{k}\varepsilon + k\varepsilon^2\right)$.

■

Appendix D. The Shifting Heavy-Hitters Problem – Missing Details

Towards designing a private algorithm for the shifting heavy-hitters problem, we generalize algorithm `ThresholdMonitor` for the evolving data setting. Consider algorithm `ThresholdMonitorEvolving`, or `TME` in short, given in Figure 5. Observe that in the special case where $S_i = S$ for all i (i.e., when the database does not change over time) then this algorithm is identical to `ThresholdMonitor`.

Theorem 27 *Algorithm* `ThresholdMonitorEvolving` *is* (ε, δ)-*differentially private in the evolving setting.*

Recall that an adversary for algorithm `ThresholdMonitorEvolving` chooses (on every time step) both the next query and the next inputs to the users, while an adversary for algorithm `ThresholdMonitor` only chooses the queries. To prove Theorem 27, we will show that the adversary's choices for the

---

Algorithm 5 `ThresholdMonitorEvolving`

Input: Privacy parameters ε, δ, threshold t, and parameter k.

Setting: On every time step $i \in [m]$ we get a database $S_i \in X^n$, and a query $f_i : X \to [0, 1]$.

1. Initiate a counter $c(j) = 0$ for every $j \in [n]$.

2. Denote $\Delta_1 = O\left(\frac{\sqrt{k}}{\varepsilon} \log\left(\frac{k}{\delta}\right) \log\left(\frac{k}{\varepsilon} \log \frac{k}{\delta}\right)\right)$ and $\Delta_2 = O\left(\frac{\sqrt{k}}{\varepsilon} \log\left(\frac{k}{\delta}\right)\right)$.

3. In each round i, when receiving a database $S_i = (x_{i,1}, \ldots, x_{i,n})$ and a query f_i, do the following:

   (a) Let $w_i \leftarrow \text{Lap}\left(10\Delta_1\right)$ and let $v_i \leftarrow \text{Lap}\left(\Delta_2\right)$.

   (b) Denote $\overline{v}_i = \min\{v_i, \Delta_1\}$.

   (c) Let $\hat{f}_i = \left(\sum_{j \in [n] : c(j) < k} f_i(x_{i,j})\right) + w_i + \overline{v}_i$.

   (d) If $\hat{f}_i < t$, then output $a_i = \bot$.

   (e) Otherwise:

   - Output $a_i = \top$.
   - For every $j \in [n]$ set $c(j) \leftarrow c(j) + f_i(x_{i,j})$.

   (f) Proceed to the next iteration.

---

users' inputs can be encoded in the choice for the next query, without actually changing users' inputs. In other words, will map an execution of `ThresholdMonitorEvolving` (in which the data evolves over time) into an execution of `ThresholdMonitor` in which the data remains fixed.

Proof of Theorem 27 Fix an adversary A and an index $j \in [n]$. We need to show that algorithm $\text{Game}_{A,\text{TME},j}$ is (ε, δ)-differentially private. To that end, let us consider a fictional algorithm $\text{FicGame}_{A,\text{TME},j}$ in which the data given to TME does not evolve over time. (Algorithm $\text{FicGame}_{A,\text{TME},j}$ is only part of the analysis.)

Observe that by Theorem 14 we have that algorithm $\text{FicGame}_{A,\text{TME},j}$ is (ε, δ)-differentially private (w.r.t. the bit b). In addition, for every iteration i and for every $r \in [n]$ we have that $f_i(d_r) = q_i(x_{i,r}^b)$. Hence, the answers a_i are distributed exactly the same during the execution of $\text{FicGame}_{A,\text{TME},j}$ and $\text{Game}_{A,\text{TME},j}$. As a result, the view of the adversary A is also distributed the same. Therefore, algorithm $\text{Game}_{A,\text{TME},j}$ is (ε, δ)-differentially private. ■

We now restate and prove Theorem 20.

Theorem 28 (**restating Theorem 20**). *There exists an (ε, δ)-differentially private algorithm for solving the shifting heavy-hitters problem with failure probability β and error $O\left(\tau(k^*)\right)$.*

Proof We use `ThresholdMonitorEvolving` with parameters $t = O\left(\tau(k^*)\right)$ and k^* in order to identify the heavy hitters on every time step. Specifically, on every time step $i \in [m]$ we query `ThresholdMonitorEvolving` with the functions $f_x(y) = \mathbb{1}_{\{y=x\}}$ for every $x \in X$, and report x as a heavy-hitter if the corresponding answer is $\top$. The privacy properties of this algorithm follow directly from the privacy properties of `ThresholdMonitorEvolving` (see Theorem 27). The utility guarantees follow from the fact that throughout the execution we sample at most $O(m \cdot |X|)$ Laplace

---

Algorithm 6 $\texttt{FicGame}_{A,\text{TME},j}$

Input: A bit $b \in \{0, 1\}$.

1. Let $D = (d_1, \ldots, d_n) \in ([n] \times \{0, 1\})^n$ be such that $d_j = (j, b)$ and $d_r = (r, 0)$ for every $r \neq j$.

2. Instantiate algorithm $\texttt{ThresholdMonitor}$ on the database D with parameters t, k and with privacy parameters $\tilde{\varepsilon} \leftarrow O\left(\varepsilon/\sqrt{k}\right)$ and $\tilde{\delta} \leftarrow O\left(\delta/k\right)$.

3. For $i = 1, 2, \ldots, m$

   (a) The adversary A chooses two j-neighboring databases $S_i^0 = (x_{i,1}^0, \ldots, x_{i,n}^0)$ and $S_i^1 = (x_{i,1}^1, \ldots, x_{i,n}^1)$ and a query $q_i : X \rightarrow [0, 1]$.

   (b) Define $f_i : ([n] \times \{0, 1\}) \rightarrow [0, 1]$ as follows: $f_i(r, z) = q_i(x_{i,r}^z)$.

   (c) Give f_i to algorithm $\texttt{ThresholdMonitor}$ and obtain an answer a_i.

   (d) a_i is given to A.

4. Output A's view of the game, that is $S_1^0, S_1^1, q_1, a_1, \ldots, S_m^0, S_m^1, q_m, a_m$, and the internal randomness of A.

---

random variables. By the properties of the Laplace distribution, with probability at least $1 - \beta$, all of these random variables are at most $\tau(k^*)$ in absolute value (see Theorem 7). ∎