# On Differentially Private Counting on Trees

**Badih Ghazi**
Google, Mountain View, CA, US

**Pritish Kamath**
Google, Mountain View, CA, US

**Ravi Kumar**
Google, Mountain View, CA, US

**Pasin Manurangsi**
Google, Bangkok, Thailand

**Kewen Wu**
University of California at Berkeley, CA, US

## Abstract

We study the problem of performing counting queries at different levels in hierarchical structures while preserving individuals' privacy. Motivated by applications, we propose a new error measure for this problem by considering a combination of multiplicative and additive approximation to the query results. We examine known mechanisms in differential privacy (DP) and prove their optimality, under this measure, in the pure-DP setting. In the approximate-DP setting, we design new algorithms achieving significant improvements over known ones.

**2012 ACM Subject Classification** Theory of computation → Theory of database privacy and security

**Keywords and phrases** Differential Privacy, Algorithms, Trees, Hierarchies

**Digital Object Identifier** 10.4230/LIPIcs.ICALP.2023.66

**Category** Track A: Algorithms, Complexity and Games

**Related Version** *Full Version*: `https://arxiv.org/abs/2212.11967`

**Funding** *Kewen Wu*: Most of this work was done while at Google.

**Acknowledgements** KW wants to thank Xin Lyu for helpful references on the sparse vector technique. We thank anonymous ITCS'23 and ICALP'23 reviewers for helpful feedback.

## 1 Introduction

With the increasing need to preserve the privacy of users, differential privacy (DP) [18, 17] has emerged as a widely popular notion that provides strong guarantees on user privacy and satisfies compelling mathematical properties. There have been many deployments of DP in the field of data analytics both in industry [4, 14] and by government agencies [3].

We start by recalling the formal definition of DP, tailored to our setting.

▶ **Definition 1** (Differential Privacy). *Let $\mathcal{A}$ be a randomized algorithm taking an integer vector as input. We say $\mathcal{A}$ is* $(\varepsilon, \delta)$-differentially private *(i.e., $(\varepsilon, \delta)$-DP) if*

$$\mathbf{Pr}\left[\mathcal{A}(\boldsymbol{x}) \in S\right] \leq e^{\varepsilon} \cdot \mathbf{Pr}\left[\mathcal{A}(\boldsymbol{x}') \in S\right] + \delta,$$

*holds for any measurable subset $S$ of $\mathcal{A}$'s range and any two neighboring inputs $\boldsymbol{x}, \boldsymbol{x}'$, where $\boldsymbol{x}, \boldsymbol{x}'$ are considered* neighbors *iff $\|\boldsymbol{x} - \boldsymbol{x}'\|_1 = 1$.*

*When $\delta = 0$, we say $\mathcal{A}$ is $\varepsilon$-DP (aka* pure-DP*); the case $\delta > 0$ is* approximate-DP.

### Estimating Counts in Trees

A fundamental task in data analytics is to aggregate counts over hierarchical subsets (specifically, trees) of the input points. For example, the government might be interested in the number of households, aggregated at the state, country, and city levels. As another example, online advertisers might be interested in the number of user clicks on product ads, when there is a category hierarchy on the products. The tree aggregation problem has been the subject of several previous works in DP including in the context of range queries [13, 42, 21, 43], the continuous release model [20, 10], private machine learning [31, 30], and the US census top-down algorithms [2, 1, 12, 11], to name a few[1]. In this work, we revisit this basic problem and present new perspectives and results.

Let $\mathcal{T}$ be a rooted tree of depth[2] $d$ and arity $k$; the structure of $\mathcal{T}$ is known a priori. Let $\mathsf{nodes}(\mathcal{T})$ be the set of *nodes* and $\mathsf{leaves}(\mathcal{T})$ be the set of *leaves* in $\mathcal{T}$. The problem of private aggregation in trees can be formalized as follows.

▶ **Problem 2** (Tree Aggregation). *Given a tree $\mathcal{T}$, the input to the problem is a vector $\boldsymbol{x} \in \mathbb{N}^{\mathsf{leaves}(\mathcal{T})}$, where $x_v \in \mathbb{N}$ is a value for $v \in \mathsf{leaves}(\mathcal{T})$. For each node $u \in \mathcal{T}$, define its weight $w_u$ by*

$$w_u = \sum_{v \text{ is a leaf under } u} x_v.$$

*The desired output is a DP estimate vector $\widetilde{\boldsymbol{w}} \in \mathbb{R}^{\mathsf{nodes}(\mathcal{T})}$ of $\boldsymbol{w}$.*

In the above formulation, the input $x_v$ represents the number of individuals that contribute to the leaf $v$, and the weight $w_u$ counts all the number of individuals that contribute to any of its descendants (or itself). As before, $\boldsymbol{x}, \boldsymbol{x}'$ are neighbors iff $\|\boldsymbol{x} - \boldsymbol{x}'\|_1 = 1$.

Besides being a natural problem on its own, algorithms for tree aggregation also serve as subroutines for solving other problems such as range queries [13, 42, 21, 43].

### Linear Queries and Error Measure

Tree aggregation in fact belongs to a class of problems called *linear queries* – one of the most widely studied problems in DP (see, e.g., [15, 19, 28, 5, 39, 9, 37, 6, 24, 38]). In its most general form, the problem can be stated as follows.

▶ **Problem 3** (Linear Queries). *For a given workload matrix $\boldsymbol{W} \in \mathbb{R}^{m \times n}$, the input to the $\boldsymbol{W}$-*linear query *problem is a vector $\boldsymbol{x} \in \mathbb{N}^n$ and the output is a DP estimate of $\boldsymbol{W}\boldsymbol{x}$.*

It is easy to see that the tree aggregation problem can be viewed as a linear query problem, where the binary workload matrix $\boldsymbol{W}^{\mathcal{T}} \in \{0,1\}^{\mathsf{nodes}(\mathcal{T}) \times \mathsf{leaves}(\mathcal{T})}$ encodes if each leaf (corresponding to a column index) is a descendant of (or itself) each node (corresponding to a row index).

Two error measures have been studied in the literature: the (expected) $\ell_2^2$*-error*[3]

$$\ell_2^2\text{-}\mathsf{error}(\mathcal{M}; \boldsymbol{W}) := \max_{\boldsymbol{x} \in \mathbb{N}^n} \sqrt{\frac{1}{m} \mathbb{E}\left[\|\mathcal{M}(\boldsymbol{x}) - \boldsymbol{W}\boldsymbol{x}\|_2^2\right]},$$

[1] We remark that there is a reduction from our problem to that of releasing thresholds, which we discuss in more detail in Section 1.3.

[2] The *depth* is defined to be the maximum number of nodes along a root-to-leaf path of the tree.

[3] In some previous work, $\ell_2^2\text{-}\mathsf{error}(\mathcal{M}; \boldsymbol{W})$ is defined as $\max_{\boldsymbol{x} \in \mathbb{N}^n} \frac{1}{m} \mathbb{E}\left[\|\mathcal{M}(\boldsymbol{x}) - \boldsymbol{W}\boldsymbol{x}\|_2^2\right]$ (without the square root). We use the current version as it is more convenient to deal with in our error analysis. In any case, we can obviously convert a bound in one version to the other.

and the (expected) $\ell_\infty$-*error*

$$\ell_\infty\text{-error}(\mathcal{M}; \boldsymbol{W}) := \max_{\boldsymbol{x} \in \mathbb{N}^n} \mathbb{E}\left[\|\mathcal{M}(\boldsymbol{x}) - \boldsymbol{W}\boldsymbol{x}\|_\infty\right],$$

where $\mathcal{M}$ is a DP mechanism for the $\boldsymbol{W}$-linear query problem. Indeed, previous works have characterized the best possible errors in the approximate-DP case up to polylogarithmic factors for any given workload $\boldsymbol{W}$. (See the discussion in [24] for more details.)

It is worth noting that these measures focus only on the *additive error* of the query, i.e., $\mathcal{M}(\boldsymbol{x}) - \boldsymbol{W}\boldsymbol{x}$. In many scenarios, however, this is not the only possible measure of error. Specifically, in this work, we seek to expand the error measure by additionally incorporating *multiplicative error*. Intuitively, multiplicative errors are meaningful when the true answer (i.e., $(\boldsymbol{W}\boldsymbol{x})_i$) is quite large; e.g., if the true error is $10^6$, then we should not be distinguishing whether the additive error is 10 or 100 as both of them are very small compared to $10^6$. In addition to this intuition, multiplicative errors have also been used in other contexts such as in empirical evaluations of range queries (e.g., [13, 40, 43]).

With the above discussion in mind, we now proceed to define the error measure.

▶ **Definition 4** (Multiplicative Root Mean Squared Error). *Given parameter $\alpha > 0$, we define an* $\alpha$-multiplicative root mean squared error ($\alpha$-RMSE) *of an estimate $\widetilde{z}$ of the true answer $z \geq 0$ as*

$$\mathsf{RMSE}_\alpha(\widetilde{z}, z) := \sqrt{\mathop{\mathbb{E}}_{\widetilde{z}}\left[\left(\max\left\{|\widetilde{z} - z| - \alpha \cdot z, 0\right\}\right)^2\right]}.$$

*For $\boldsymbol{W}$-linear query, we define an* $\alpha$-multiplicative maximum root mean squared error ($\alpha$-mRMSE) *of a mechanism $\mathcal{M}$ to be*

$$\mathsf{mRMSE}_\alpha(\mathcal{M}; \boldsymbol{W}) := \max_{\boldsymbol{x} \in \mathbb{Z}^n} \max_{i \in [m]} \mathsf{RMSE}_\alpha\left(\mathcal{M}(\boldsymbol{x})_i, (\boldsymbol{W}\boldsymbol{x})_i\right).$$

Note that when $\alpha = 0$ (i.e., the error is only additive), our notion of $\alpha$-RMSE coincides with that of the standard RMSE. By taking the maximum error across all queries when defining the error for linear queries, we mitigate the weakness of $\ell_2^2$-error bound, which allows some queries to incur huge errors, while still avoiding the "union bound issue" faced in the $\ell_\infty$-error. The latter can be significant as the number of queries here can be exponential in the depth $d$.

We remark that our algorithms also achieve the usual with high probability guarantees, i.e., with probability at most $\eta$, $|\widetilde{z} - z| \leq \alpha \cdot \max\{z, \tau\}$ for some threshold $\tau$. We defer such a statement to later sections for simplicity of comparing the bounds. Furthermore, our error notion implies upper bounds on "smoothed relative errors" used for empirical evaluations in previous works [40, 43]. We provide a formal statement in the full version.

When $\alpha = 0$, we drop the "$\alpha$-multiplicative" or "$\alpha$-" prefixes and refer to the errors simply as maximum RMSE or mRMSE. Similarly, we also drop $\alpha$ from the subscript and simply write mRMSE instead of $\mathsf{mRMSE}_0$.

## 1.1 Our Results

Two known baselines for tree aggregation are the $\varepsilon$-DP Laplace mechanism and $(\varepsilon, \delta)$-DP Gaussian mechanism, which achieve mRMSE of $O(d/\varepsilon)$ and $O(\sqrt{d\log(1/\delta)}/\varepsilon)$ respectively. We start by showing that these are already tight for the additive-only errors:

▶ **Theorem 5** (Informal; see Theorem 28). *There is no $\varepsilon$-DP algorithm for tree aggregation with* mRMSE *$o(d/\varepsilon)$, even for binary trees.*

**Table 1** Overview of results; entries indicate upper/lower bounds on $\alpha$-mRMSE. Upper bounds corresponding to Laplace and Gaussian mechanisms are formally stated in Corollary 15. For simplicity we omit the dependence on $\alpha$ in the additive-multiplicative bounds here.

| Type of error | $\varepsilon$-DP | $(\varepsilon, \delta)$-DP |
|---|---|---|
| Additive-only ($\alpha = 0$) | $O(d/\varepsilon)$ : Laplace | $O_{\varepsilon,\delta}(\sqrt{d})$ : Gaussian |
| | $\Omega(d/\varepsilon)$ : Theorem 28 | $\Omega_{\varepsilon,\delta}(\sqrt{d})$ : Theorem 29 |
| Additive-Multiplicative ($0 < \alpha < 1$) | $\Omega(d/\varepsilon)$ : Theorem 28 | $O_{\varepsilon,\delta}(\log d)$ : Theorem 21 |

▶ **Theorem 6** (Informal; see Theorem 29). *There is no $(\varepsilon, \delta)$-DP algorithm for tree aggregation with* mRMSE $o_{\varepsilon,\delta}(\sqrt{d})$*, even for binary trees.*

Given the above results, it is therefore natural to ask whether multiplicative errors can help reduce the error bound. For pure-DP, we show that this unfortunately is not the case.

▶ **Theorem 7** (Informal; see Theorem 28). *For any constant $\alpha < 1$, there is no $\varepsilon$-DP algorithm for tree aggregation with $\alpha$-*mRMSE *$o(d/\varepsilon)$, even for binary trees.*

Our next – and perhaps the most surprising – result is that, unlike in the pure-DP case, allowing multiplicative approximation in approximate-DP allows us to reduce the upper bounds exponentially from $O_{\varepsilon,\delta}(\sqrt{d})$ to $O_{\varepsilon,\delta}(\log d)$:

▶ **Theorem 8** (Informal; see Theorem 21). *For any constant $\alpha > 0$, there is an efficient $(\varepsilon, \delta)$-DP algorithm for tree aggregation with $\alpha$-*mRMSE *$O(\log(d/\delta)/\varepsilon)$.*

We remark that Theorem 8 has worse dependency on $\delta$ than the $(\varepsilon, \delta)$-DP Gaussian mechanism. Indeed, the former has $\log(1/\delta)$ whereas the latter only has $\sqrt{\log(1/\delta)}$. However this gap is somewhat unavoidable as we will discuss in Remark 33 when $\delta$ is small, say, $\delta = 2^{-\Omega(d)}$. Our results are summarized in Table 1.

Our bounds do *not* depend on the arity $k$ of $\mathcal{T}$. This is immediate for the lower bounds (Theorems 5–7) since it suffices to prove it for binary trees $k = 2$. The reason for the upper bounds (Theorem 8) is less clear, relying on the fact that the weights of two nodes are correlated iff they are on the same root-to-leaf path, which is irrelevant of the arity.

## 1.2 Proof Overview

### Probabilistic Utility Guarantee

Recall that we defined the error $\alpha$-mRMSE as a variant of RMSE but with a multiplicative error subtracted out. While this gives us a nice scalar quantity (once $\alpha$ is fixed) to work with and state the results, it will be useful in the subsequent analyses to define a probabilistic version of the guarantee with additional fixed thresholds.

In this different utility guarantee (formalized in (1)), every node $u$ is given an additional threshold $\tau_u$, and a randomized output $\widetilde{\boldsymbol{w}}$ is accurate if for all $u \in \mathsf{nodes}(\mathcal{T})$ we have with probability at least $1 - \eta$ that

$$|\widetilde{w}_u - w_u| \leq \alpha \cdot \max\{w_u, \tau_u\}.$$

The smaller the thresholds $\tau_u$'s are, the better the accuracy will be.

The benefit of having $\tau_u$ is that it is independent of $w_u$ and is explicitly available to the algorithms; this formulation may be of independent interest. On the other hand, this probabilistic guarantee is closely related to the original $\alpha$-mRMSE in Definition 4:

1. Any algorithm with low $\alpha$-mRMSE has good probabilistic guarantee (Lemma 19).
2. Any algorithm with good probabilistic guarantee can be converted into one with low $\alpha$-mRMSE (Lemma 20).

### Improved Approximate-DP Algorithm

Given Item 2, it suffices to design an algorithm with a good probabilistic guarantee, i.e., works for thresholds $\tau_u$'s as small as possible. Our algorithm can be decomposed into two parts: a reduction step and a classification algorithm.

**Reduction.** Since the error measure is multiplicative when $w_u$ is large, we design a geometric sequence of thresholds and classify each $w_u$ into the correct interval created by the thresholds. To do so, every time we use a classification algorithm to find nodes that are above the current threshold, and in the next round we only focus on the ones below the threshold. Assuming previous classifications are all correct, the weights of the nodes above the current threshold are actually below the previous threshold. Such a sandwiching relation provides a good approximation if the granularity of the thresholds is not too large compared with the ratio $\alpha$ (Lemma 26).

Moreover, we assign privacy and error parameters in the same geometric fashion, thus their telescoping sum (from composition theorems) converges.

**Classification.** Given any fixed threshold $\tau$, the goal is to correctly classify each $w_u$ to be either *above* or *below* $\tau$. Naively, to ensure every node is correctly classified, we need to apply a union bound over all the nodes. This will incur a $\mathsf{poly}(d)$ overhead if we use Laplace noise or Gaussian noise as in the standard mechanisms (Lemmas 13 and 14) since the tree can have exponential size. To deal with this issue, we use a *truncated Laplace mechanism* where the Laplace noise is truncated to be bounded (Lemma 16); this ensures that the estimation error is always at most the truncation range and thus can obviate a union bound. However, we still need to pay the privacy loss from compositions. If one node is the ancestor of another, then the input leaves they depend on must overlap, which means simply estimating every node's weight will incur $d$ rounds of composition. To improve this, our classification algorithm will find the transition nodes in the tree: a *transition node* is one whose weight exceeds $\tau$ but none of its children has weight above $\tau$. Given the locations of the transition nodes, we can easily classify the other nodes: a node is above $\tau$ iff it is the ancestor of (or itself) a transition node. Assuming the previous classification with threshold $\tau' > \tau$ succeeds, none of the nodes' weights should exceed $\tau'$ now. Since there are at most $\tau'/\tau$ transition nodes in a tree, we can use the *sparse vector technique* [22] to find them, and incur fewer rounds of composition.

We remark that the idea of using increasing thresholds and sparse vector techniques has been used in [7, 26, 25].

### Pure-DP Lower Bounds

Given Item 1, it suffices to rule out DP algorithms with very strong probabilistic guarantees, i.e., works for thresholds $\tau_u$'s that are too small.

Our proof uses the *packing argument* [28]: we construct extremal datasets where any two datasets have a large distance. Then if the output has small error, we can correctly identify the input dataset. On the other hand, by the privacy guarantee, the output distribution for different datasets, though having a large distance, should not be too different, contradicting the fact that they decode to different input datasets.

Not surprisingly, our extremal datasets place the maximal value on a leaf and keep other leaves empty. But the key issue is the decoding step. Indeed, previous packing arguments work with $\ell_\infty$-error, where the error on *all* output coordinates is small with high probability, thus admitting simple decoding algorithms. We, however, can only guarantee the error on any *fixed* node is small with high probability; we also cannot use a union bound since the output size can be exponential.

The way we circumvent this is by designing a novel probabilistic decoding algorithm where we will correctly decode to the input dataset with probability large enough to derive a contradiction. The decoding algorithm itself performs a random walk on the tree where each step favors the larger estimated weight. Then the success probability of decoding can be lower bounded in terms of the number of correctly classified nodes, which in turn can be lower bounded by its expectation and our probabilistic guarantee suffices.

#### Additive-Only Lower Bounds

The above packing argument only works for pure-DP setting (or $(\varepsilon, \delta)$-DP but with exponentially small $\delta$). Indeed, as shown by our improved approximate-DP algorithm (Theorem 8), the bound can be exponentially small if we allow both approximate-DP and $\alpha > 0$. Therefore we now turn to the only remaining case: approximate-DP and $\alpha = 0$. In this case, by Definition 4, $\alpha$-mRMSE is an additive-only error.

Our proof starts by slightly modifying the error characterization of linear queries from [24]. This shows that mRMSE for $\boldsymbol{W}$-linear query is characterized by a factorization norm of $\boldsymbol{W}$. Thus proving Theorem 6 boils down to showing a lower bound on this factorization norm of the binary tree matrix. Following previous works on range queries (e.g., [36]), we do so by invoking a dual (maximum-based) characterization of the factorization norm from [35, 33] and give an explicit solution to this dual formulation.

## 1.3 Relation Between Tree Aggregation and Releasing Thresholds

There is a simple reduction from the tree aggregation problem (Problem 18) to the problem of releasing thresholds. Recall that the problem of releasing thresholds is the same as linear queries with the workload matrix $\boldsymbol{W}^{\mathrm{th}} \in \{0,1\}^{m \times m}$ being the matrix with upper-triangular entries (including the diagonal entries) equal to one.

The reduction works as follows. First, we may index the leaves from $1, \ldots, |\mathsf{leaves}(\mathcal{T})|$ based, say, on their order in the DFS traversal of the tree. It is not hard to see that $w_u$ of any node in the tree corresponds to $(\boldsymbol{W}^{\mathrm{th}}\boldsymbol{x})_b - (\boldsymbol{W}^{\mathrm{th}}\boldsymbol{x})_{a-1} = \sum_{v \in [a,b]} x_v$ for some $a, b \in \{1, \ldots, |\mathsf{leaves}(\mathcal{T})|\}$. Therefore, we may run any DP threshold releasing algorithm (with $m = |\mathsf{leaves}(\mathcal{T})|$) and solve the tree aggregation problem.

The above reduction yields an mRMSE error for the tree aggregation problem that is of the same order as that of releasing thresholds (with $m = |\mathsf{leaves}(\mathcal{T})|$). For the latter, it is known that the tight error is $\Theta_{\varepsilon,\delta}(\log m)$ [29][4]. Assuming that each node at depth less than $d$ has at least two children, $|\mathsf{leaves}(\mathcal{T})| \geq 2^d$ and therefore, the error yielded by this reduction is at least $\Omega_{\varepsilon,\delta}(d)$. In other words, this is not even as good as the straightforward Gaussian mechanism for our problem, which yields an error of $O_{\varepsilon,\delta}(\sqrt{d})$ (and we have shown this to be tight in Theorem 29).

We note that there is another line of research that studies privately learning threshold functions. Recent work has shown that learning threshold functions with $(\varepsilon, \delta)$-DP in the PAC model only requires $O_{\alpha,\varepsilon,\delta}\left((\log^* m)^{O(1)}\right)$ samples [32]. Due to the connection

[4] In fact, the lower bound in [29] is even stronger as it holds against the $\ell_2^2$-error.

between learning thresholds and releasing threshold functions presented in [8], this gives an algorithm for the latter with an error bound of $O_{\varepsilon,\delta}\left((\log^* m)^{O(1)} \cdot (\log n)^{2.5}\right)$, where $n$ denotes $\|\boldsymbol{x}\|_1$ (i.e., the total count across all leaves in our setting). When combining this bound with the above reduction, one gets a tree aggregation algorithm with error $O_{\varepsilon,\delta}\left((\log^* |\mathsf{leaves}(\mathcal{T})|)^{O(1)} \cdot (\log n)^{2.5}\right)$. Although the term $(\log^* |\mathsf{leaves}(\mathcal{T})|)^{O(1)}$ is very small, this error bound is not directly comparable to the lower and upper bounds achieved in our paper because our bounds are *independent* of $n$ whereas there is a dependency of $(\log n)^{2.5}$ in this releasing threshold-based bound. (Note also that the dependency on $n$ cannot be removed while keeping the dependency on $m$ sublogarithmic, as this would contradict with the aforementioned lower bound of [29].)

Finally, we remark that the reduction does *not* work if we allow the threshold releasing algorithm to incur multiplicative errors. This is because we need to subtract two thresholds to get $w_u$ for each node $u$ in the tree, and subtraction does not preserve multiplicative approximation guarantees.

### Paper Organization

We formalize the notation in Section 2. Then in Section 3 we introduce the error measure we will actually use in designing algorithms and proving lower bounds; and relate it to the $\alpha$-mRMSE measure. The improved approximate-DP algorithm is presented in Section 4 and the corresponding lower bounds are in Section 5. The concluding remarks are in Section 6.

## 2 Preliminaries

We use $\ln(\cdot)$ and $\log(\cdot)$ to denote the logarithm with base $e$ and 2 respectively. For a positive integer $n$, let $[n]$ denote the set $\{1, \ldots, n\}$. Let $\mathbb{N}$ denote the set of non-negative integers.

### 2.1 Norms

We use boldface uppercase (e.g., $\boldsymbol{A}$) to denote matrices and boldface lowercase (e.g., $\boldsymbol{x}$) to denote vectors. We use $\mathbf{0}, \mathbf{1}$ to denote the all-zeros and all-ones vectors / matrices.

For $\boldsymbol{x} \in \mathbb{R}^m$, its *$\ell_p$-norm* is defined as $\|\boldsymbol{x}\|_p := \left(\sum_{i\in[m]} |x_i|^p\right)^{1/p}$ for any $1 \le p < \infty$. Its *$\ell_\infty$-norm* is defined as $\|\boldsymbol{x}\|_\infty := \max_{i\in[m]} |x_i|$.

### 2.2 Tools from Differential Privacy

Here we note some useful facts regarding differential privacy [18, 17, 41].

▶ **Fact 9** (Post-Processing). *Let $\mathcal{A}_1$ be an $(\varepsilon, \delta)$-DP algorithm and $\mathcal{A}_2$ be a (randomized) post-processing algorithm. Then the algorithm $\mathcal{A}(\boldsymbol{x}) = \mathcal{A}_2(\mathcal{A}_1(\boldsymbol{x}))$ is still an $(\varepsilon, \delta)$-DP algorithm.*

▶ **Fact 10** (Group Privacy). *Let $\mathcal{A}$ be an $(\varepsilon, \delta)$-DP algorithm and $\boldsymbol{x}, \boldsymbol{x}'$ be two arbitrary inputs. Define $k = \|\boldsymbol{x} - \boldsymbol{x}'\|_1$. Then for any measurable subset $S$ of $\mathcal{A}$'s range, we have*

$$\mathbf{Pr}\left[\mathcal{A}(\boldsymbol{x}) \in S\right] \le e^{k\cdot\varepsilon} \cdot \mathbf{Pr}\left[\mathcal{A}(\boldsymbol{x}') \in S\right] + \delta \cdot \frac{e^{k\cdot\varepsilon} - 1}{e^{\varepsilon} - 1}.$$

▶ **Fact 11** (Basic Composition). *Let $\mathcal{A}_1$ be an $(\varepsilon_1, \delta_1)$-DP algorithm and $\mathcal{A}_2$ be an $(\varepsilon_2, \delta_2)$-DP algorithm. Then $\mathcal{A}(\boldsymbol{x}) = (\mathcal{A}_1(\boldsymbol{x}), \mathcal{A}_2(\mathcal{A}_1(\boldsymbol{x}), \boldsymbol{x}))$ is an $(\varepsilon_1 + \varepsilon_2, \delta_1 + \delta_2)$-DP algorithm.*

▶ **Fact 12** (Parallel Composition). *Let $\mathcal{A}_1$ be an $(\varepsilon_1, \delta_1)$-DP algorithm and $\mathcal{A}_2$ be an $(\varepsilon_2, \delta_2)$-DP algorithm. Assume $\mathcal{A}_1$ and $\mathcal{A}_2$ depend on disjoint subsets of input coordinates. Then the algorithm $\mathcal{A}(\boldsymbol{x}) = (\mathcal{A}_1(\boldsymbol{x}), \mathcal{A}_2(\mathcal{A}_1(\boldsymbol{x}), \boldsymbol{x}))$ is a $(\max\{\varepsilon_1, \varepsilon_2\}, \max\{\delta_1, \delta_2\})$-DP algorithm.*

Two of the most ubiquitous mechanisms in DP are the Laplace and Gaussian mechanisms [16, 17, 23]. We use $\mathsf{Lap}(\sigma)$ to denote the *Laplace distribution* with parameter $\sigma$, whose density function is $\frac{1}{2\sigma}\exp\left(-\frac{|x|}{\sigma}\right)$. We use $\mathsf{N}(\mu, \sigma^2)$ to denote the *Gaussian distribution* with mean $\mu$ and variance $\sigma^2$, whose density function is $\frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\right)$.

For matrix $\boldsymbol{A}$ and $p \geq 1$, we use $\|\boldsymbol{A}\|_{\infty,p}$ to denote the maximum $\ell_p$-norm among all column vectors of $\boldsymbol{A}$. The Laplace and Gaussian mechanisms for linear queries are stated next.

▶ **Lemma 13** (Laplace Mechanism, [16, 34]). *For the $\boldsymbol{W}$-linear query problem, the algorithm that outputs $\boldsymbol{W}\boldsymbol{x} + \boldsymbol{z}$ is $\varepsilon$-DP, where each entry of $\boldsymbol{z}$ is drawn i.i.d. from $\mathsf{Lap}\left(\|\boldsymbol{W}\|_{\infty,1}/\varepsilon\right)$.*

▶ **Lemma 14** (Gaussian Mechanism, [17, 23]). *Assume $\varepsilon, \delta \in (0,1)$. For the $\boldsymbol{W}$-linear query problem, the algorithm that outputs $\boldsymbol{W}\boldsymbol{x} + \boldsymbol{z}$ is $(\varepsilon, \delta)$-DP, where each entry of $\boldsymbol{z}$ is drawn i.i.d. from $\mathsf{N}\left(0, 2\ln(1.25/\delta)\|\boldsymbol{W}\|^2_{\infty,2}/\varepsilon^2\right)$.*

Recall that for a tree $\mathcal{T}$, we let $\boldsymbol{W}^{\mathcal{T}} \in \{0,1\}^{\mathsf{nodes}(\mathcal{T}) \times \mathsf{leaves}(\mathcal{T})}$ be the indicator matrix whether a leaf is a descendant of (or itself) a node. This represents the tree aggregation problem as $\boldsymbol{W}^{\mathcal{T}}$-linear queries. Observe also that $\left\|\boldsymbol{W}^{\mathcal{T}}\right\|_{\infty,1} = d$ and $\left\|\boldsymbol{W}^{\mathcal{T}}\right\|_{\infty,2} = \sqrt{d}$. Therefore, we can apply Lemma 13 and Lemma 14 (together with tail bounds for Laplace and Gaussian distributions) to obtain the following baselines for tree aggregation.

▶ **Corollary 15** (Baseline Algorithms). *For the tree aggregation problem, there exists an $\varepsilon$-DP (resp., $(\varepsilon, \delta)$-DP) algorithm with $\mathsf{mRMSE}$ $O(d/\varepsilon)$ (resp., $O(\sqrt{d \cdot \log(1/\delta)}/\varepsilon)$).*

We will also use the Laplace mechanism with a bounded range. For any $R > 0$, we use $\mathsf{TruncLap}(\sigma, R)$ to denote the *truncated Laplace distribution* with parameter $\sigma$ and range $[-R, R]$, whose density function is proportional to $\exp(-|x|/\sigma)$ for $x \in [-R, R]$ and is 0 if $|x| > R$. Note that $\mathsf{Lap}(\sigma) = \mathsf{TruncLap}(\sigma, +\infty)$.

▶ **Lemma 16** (Truncated Laplace Mechanism, [27]). *The algorithm that, on input $x \in \mathbb{Z}$, outputs $x + z$ is $(\varepsilon, \delta)$-DP, where $z \sim \mathsf{TruncLap}\left(\frac{1}{\varepsilon}, \frac{1}{\varepsilon}\ln\left(1 + \frac{e^{\varepsilon}-1}{2\delta}\right)\right)$.*

Our algorithm will also make use of the celebrated *sparse vector technique* [22]. For convenience, we apply it in a black-box way as the following oracle.

▶ **Lemma 17** (Sparse Vector Technique, [22, 23]). *There exists an $\varepsilon$-DP algorithm $\texttt{Sparse}(\boldsymbol{x}, \{f_i\}; \eta, c, \tau, \varepsilon)$ such that:*

- *Input. A dataset $\boldsymbol{x}$, an adaptively chosen stream $\{f_i\}_{i=1,\ldots,d}$ of sensitivity-1 queries, error probability $\eta > 0$, a cutoff point $c \geq 1$, a threshold $\tau$, and a privacy bound $\varepsilon > 0$.*[5]
- *Output. A stream $\{a_i\}_{i=1,\ldots,d} \in \{\bot, \top\}^*$ of on-the-fly answers.*
- *Accuracy. Let $i^*$ be the index of the $c$th $\top$ in $\{a_i\}_{i\in[d]}$; if there are less than $c$ $\top$'s, let $i^* = d$. Then, for $\Delta = 8c/\varepsilon \cdot \ln(2d/\eta)$, with probability at least $1-\eta$ the following holds for all $i \leq i^*$: If $a_i = \top$, then $f_i(\boldsymbol{x}) \geq \tau - \Delta$; otherwise (i.e., $a_i = \bot$) $f_i(\boldsymbol{x}) < \tau + \Delta$.*

The $\texttt{Sparse}()$ algorithm is in [23, Algorithm 2] with $\delta = 0$, where its privacy is proved in [23, Theorem 3.25]. The accuracy part is also immediate from the algorithm description. For completeness, we give a proof in the full version.

[5] We say a query $f$ is *sensitivity-1* if $\left|f(\boldsymbol{x}) - f(\boldsymbol{x}')\right| \leq 1$ for any two neighboring inputs $\boldsymbol{x}, \boldsymbol{x}'$.

## 3 Threshold-Based Utility

As mentioned in Section 1.2, we consider a probabilistic utility guarantee with a threshold value supplied at each node in the tree. Then we relate it to the original $\alpha$-mRMSE notion.

▶ **Problem 18** (Tree Aggregation with Thresholds). *Consider the tree aggregation problem (i.e., Problem 2), wherein additionally, we have a threshold $\tau_u \geq 0$ corresponding to each node $u \in \mathsf{nodes}(\mathcal{T})$ (both internal nodes and leaves). The desired output for the problem is an estimate $\widetilde{\boldsymbol{w}} \in \mathbb{R}^{\mathsf{nodes}(\mathcal{T})}$ of $\boldsymbol{w}$ as before.*

*For parameters $\eta, \alpha \in [0, 1)$, we say that an algorithm for tree aggregation is* $(\alpha, \eta)$-accurate *(w.r.t. the given thresholds) if its output vector $\widetilde{\boldsymbol{w}} \in \mathbb{R}^{\mathsf{nodes}(\mathcal{T})}$ satisfies the following:*

$$\text{For all } u \in \mathsf{nodes}(\mathcal{T}) \quad : \quad \mathbf{Pr}\left[|\widetilde{w}_u - w_u| \leq \alpha \cdot \max\{w_u, \tau_u\}\right] \geq 1 - \eta. \tag{1}$$

Unless otherwise specified, for the rest of the paper, we assume that the input to the tree aggregation problem includes a threshold at each node in the tree. Let $\tau_{\mathsf{min}} := \min_{u \in \mathsf{nodes}(\mathcal{T})} \tau_u$ and $\tau_{\mathsf{max}} := \max_{u \in \mathsf{nodes}(\mathcal{T})} \tau_u$. Now, we show that any algorithm with low $\alpha$-mRMSE also yields a certain utility guarantee in the sense of (1).

▶ **Lemma 19** (Proof in Full Version). *For any $\alpha' > \alpha \geq 0$, $\eta > 0$, any tree aggregation algorithm with $\alpha$-mRMSE at most $(\alpha' - \alpha)\sqrt{\eta} \cdot \tau_{\mathsf{min}}$ is also $(\alpha', \eta)$-accurate.*

In contrast to the above bound, our algorithms will satisfy much stronger exponential tail bounds, which will be clear in the next section. To complement Lemma 19, we show that any algorithm that is $(\alpha, \eta)$-accurate, as per (1), can be made having small $\alpha$-mRMSE.

▶ **Lemma 20** (Proof in Full Version). *If there is an $(\varepsilon/2, \delta/2)$-DP algorithm that is $(\alpha, \eta)$-accurate for tree aggregation, then there is an $(\varepsilon, \delta)$-DP algorithm for tree aggregation with $\alpha$-mRMSE at most $O\left(\alpha \cdot \tau_{\mathsf{max}} + d\sqrt{\eta} \cdot \left(1 + \frac{1}{\varepsilon}\log\left(\frac{1}{\delta}\right)\right)\right)$.*

With the above two lemmas in mind, it essentially suffices for us to consider the accuracy notion in (1), which will be convenient for the rest of the paper.

## 4 Upper Bounds

In this section, we present a new algorithm for tree aggregation in the approximate-DP setting, achieving a significant improvement over the baseline algorithm (i.e., Corollary 15).

▶ **Theorem 21.** *Let $\alpha > 0$ be a parameter. For any $\varepsilon > 0$ and $\delta \in (0, 1)$, there is an $(\varepsilon, \delta)$-DP algorithm for tree aggregation with $\alpha$-mRMSE at most $O\left(\frac{1}{\alpha^3} \cdot \left(\frac{1}{\varepsilon}\log\left(\frac{2d}{\delta}\right) + 1\right)\right)$.*

By Lemma 20, it suffices to design an efficient algorithm for Problem 18 with small thresholds for every node in the tree. This will be the focus of the section. To this end, we will reduce the estimation problem to a classification problem, and design algorithms for the classification task. We first present the classification algorithm in Section 4.1, then describe the reduction in Section 4.2, and finally put them together in Section 4.3.

### 4.1 A Classification Problem

For later reduction, the classification task here needs to have a stronger notion of success: the classification for nodes should be correct *simultaneously* with high probability, whereas Problem 18 only requires the estimation of any fixed node to be correct with high probability.

▶ **Problem 22.** *Let $\mathcal{T}$ be a tree of depth $d$ and arity $k$. Let $\tau \geq 0$ be the (common) threshold for all nodes. Let $M > 0, \eta \in [0, 1/2), \alpha \in [0, 1)$ be parameters.*

*The input to the problem is non-negative integer values $x_v$ for each $v \in \mathsf{leaves}(\mathcal{T})$. For each node $u$, its weight $w_u$ is $w_u = \sum_{v \text{ is a leaf under } u} x_v$.*

*The desired output is a vector $\boldsymbol{w}' \in \{\bot, \top\}^{\mathsf{nodes}(\mathcal{T})}$ that with probability at least $1 - \eta$ satisfies the following: when the weight of the root of $\mathcal{T}$ is at most $M$, for each $u \in \mathsf{nodes}(\mathcal{T})$,*

- *if $w_u \geq (1 + \alpha) \cdot \tau$, then $w'_u = \top$;*
- *if $w_u < (1 - \alpha) \cdot \tau$, then $w'_u = \bot$;*
- *otherwise (i.e., $(1 - \alpha) \cdot \tau \leq w_u < (1 + \alpha) \cdot \tau$), $w'_u$ can be arbitrary.*

We now present our algorithm for this classification problem and its guarantees.

▶ **Lemma 23.** *There is an $(\varepsilon, \delta)$-DP algorithm $\texttt{Classification}(\mathcal{T}; M, \eta, \alpha, \tau, \varepsilon, \delta)$ such that it solves Problem 22 assuming $\tau \geq \sqrt{\frac{2M}{\alpha\varepsilon}} \cdot \max\left\{\sqrt{48 \ln\left(\frac{2d}{\eta}\right)}, \sqrt{6 \ln\left(1 + \frac{e^{\varepsilon/2} - 1}{\delta}\right)}\right\}$.*

**Proof.** Without loss of generality we assume $\alpha \leq 1/2$. Note that if $M < \tau$, then we can simply set $w'_u \leftarrow \bot$ for all $u \in \mathsf{nodes}(\mathcal{T})$. Therefore we assume without loss of generality $M \geq \tau$ from now on. For any node $u$, let $\mathsf{depth}(u)$ denote the number of nodes on the path from the root to $u$. Algorithm 1 contains the formal description.

We first prove the privacy bound. By Lemma 17, Line 7 is $\varepsilon/2$-DP. On the other hand, by Lemma 16 and Fact 12, Lines 11–18 are $(\varepsilon/(2c), \delta/c)$-DP; and since they are executed at most $c$ times, they are $(\varepsilon/2, \delta)$-DP in total. Therefore by Fact 11, Algorithm 1 is $(\varepsilon, \delta)$-DP.

Now we turn to the correctness of Algorithm 1. Define $i^*$ to be the index of the $c$th $\top$ in $a_d, a_{d-1}, \ldots$. If there are less than $c$ $\top$'s, let $i^* \geq 1$ be the index of the last query. Define $\mathcal{E}$ to be the event that the following holds for any $i \geq i^*$:[6] If $a_i = \top$, then $f_i \geq \tau - \Delta$; and if $a_i = \bot$, then $f_i < \tau + \Delta$. By Lemma 17, $\mathbf{Pr}[\mathcal{E}] \geq 1 - \eta$.

We first show, conditioned on $\mathcal{E}$, there are always less than $c$ $\top$'s. Assume towards contradiction that there are $c$ $\top$'s. Then, conditioned on $\mathcal{E}$, for any $a_i = \top$, there exists some $u \in \mathcal{S}_i$ such that $w_u = f_i \geq \tau - \Delta$. Therefore $\widetilde{w}_u \geq w_u - R \geq \tau - \Delta - R$, which implies it will be assigned to $\top$ on Line 14. By design, all these $u$'s satisfying Line 13 form a subset of $\mathcal{T}$ where none of them is an ancestor of another. Therefore the weight of the root is lower bounded by the total weights of these nodes, which is at least $c \cdot (\tau - \Delta)$ but at most $M$. On the other hand, by the assumption on $M$ and assuming $\Delta < \alpha \cdot \tau$, we also have $c > \frac{M}{\tau - \Delta}$, which gives a contradiction.

Then for the correctness part, it suffices to show for any fixed node $v \in \mathsf{nodes}(\mathcal{T})$, we have $w'_v = \top$ if $w_v \geq (1 + \alpha) \cdot \tau$, and $w'_v = \bot$ if $w_v < (1 - \alpha) \cdot \tau$:

- Case $w_v \geq (1 + \alpha) \cdot \tau$. Assume towards contradiction that $w'_v = \bot$. Let $i_v = \mathsf{depth}(v)$. Then it means when $i = i_v$ on Line 4, $v \in \mathcal{S}_i$. Thus $f_i \geq w_v \geq (1 + \alpha) \cdot \tau$. On the other hand since $w'_v = \bot$, we must proceed to Line 9. Conditioned on $\mathcal{E}$, this implies $f_i < \tau + \Delta$, which is a contradiction assuming $\Delta \leq \alpha \cdot \tau$.
- Case $w_v < (1 - \alpha) \cdot \tau$. Assume towards contradiction that $w'_v = \top$. Let $r \in \mathsf{nodes}(\mathcal{T})$ be the deepest node in the subtree below $v$ that is assigned $\top$. Let $i_r = \mathsf{depth}(r)$. Then it means when $i = i_r$ we execute Line 14 for $r$. Thus $w_r + R \geq \widetilde{w}_r \geq \tau - \Delta - R$. Meanwhile, we also have $w_r \leq w_v < (1 - \alpha) \cdot \tau$, which is a contradiction assuming $\Delta + 2R \leq \alpha \cdot \tau$.

[6] Note that $i$ goes from $d$ down to 1 in our algorithm.

**Algorithm 1** Classification.

**Input:** $\mathcal{T}$ and parameters $M, \eta, \alpha, \tau, \varepsilon, \delta$ described in Problem 22
**Output:** $w'_u \in \{\perp, \top\}$ for all $u \in \mathsf{nodes}(\mathcal{T})$

**if** $M < \tau$ **then** set $w'_u \leftarrow \perp$ for all $u \in \mathsf{nodes}(\mathcal{T})$ and **return** $\boldsymbol{w}'$
Set $c \leftarrow \frac{M}{(1-\alpha)\tau}$ and

$$\Delta \leftarrow \frac{16c}{\varepsilon} \ln\left(\frac{2d}{\eta}\right), \quad R \leftarrow \frac{2c}{\varepsilon} \ln\left(1 + \frac{c \cdot (e^{\varepsilon/(2c)} - 1)}{\delta}\right)$$

Define $\mathcal{S}_i \leftarrow \{u \in \mathsf{nodes}(\mathcal{T}) \mid \mathsf{depth}(u) = i\}$ for each $i \in [d]$
**foreach** $i = d$ **to** $1$ **do**
  **if** $\mathcal{S}_i = \emptyset$ **then continue**
  Define query $f_i \leftarrow \max_{u \in \mathcal{S}_i} w_u$
  Get $a_i \leftarrow \mathtt{Sparse}(\boldsymbol{x}, f_i; \eta, c, \tau, \varepsilon/2)$ `/* x is the values of leaves(T) */`
  **if** $a_i = \perp$ **then**
    Set $w'_u \leftarrow \perp$ for all $u \in \mathcal{S}_i$ and update $\mathcal{S}_i \leftarrow \emptyset$
  **else** `/*` $a_i = \top$ `*/`
    **foreach** $u \in \mathcal{S}_i$ **do**
      Compute $\widetilde{w}_u \leftarrow w_u + \mathsf{TruncLap}(2c/\varepsilon, R)$
      **if** $\widetilde{w}_u \geq \tau - \Delta - R$ **then**
        Set $w'_v \leftarrow \top$ and remove $v$ from $\mathcal{S}_{i_v}$ for each $u$'s ancestor $v$ (including $u$ itself) where $i_v = \mathsf{depth}(v)$.
      **else** `/*` $\widetilde{w}_u < \tau - \Delta - R$ `*/`
        Set $w'_u \leftarrow \perp$ and remove $u$ from $\mathcal{S}_i$
      **end**
    **end**
  **end**
**end**
**return** $\boldsymbol{w}'$

Thus it suffices to make sure $\Delta, R \leq \alpha \cdot \tau/3$. Since $M \geq \tau$, we have $c \geq 1$ and $c \cdot \left(e^{\varepsilon/(2c)} - 1\right) \leq e^{\varepsilon/2} - 1$, which gives the assumption in the statement by rearranging terms and noticing $1 - \alpha \geq 1/2$. ◀

Eventually, we will use $\mathtt{Classification}(\mathcal{F}; M, \eta, \alpha, \tau, \varepsilon, \delta)$ algorithm on a forest $\mathcal{F}$ of disjoint trees with the same set of parameters. There we do not need all nodes in $\mathcal{F}$ to be classified correctly. Instead, it suffices to have all nodes in any $\mathcal{T} \in \mathcal{F}$ classified correctly.

▶ **Problem 24.** *Let $\mathcal{F} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots\}$ be a forest of disjoint trees of depth $d$ and arity $k$. Let $\tau \geq 0$ be the threshold for every node. Let $M > 0, \eta \in [0, 1/2), \alpha \in [0, 1)$ be parameters.*

*The input to the problem is non-negative integer values $x_v$ for each leaf $v$ in $\mathcal{F}$. For each node $u$, its weight $w_u$ is $w_u = \sum_{v \text{ is a leaf under } u} x_v$.*

*The desired output is a vector $\boldsymbol{w}' \in \{\perp, \top\}^{\mathsf{nodes}(\mathcal{F})}$ that, for any $\mathcal{T} \in \mathcal{F}$ with probability at least $1 - \eta$, satisfies the following: when the root of $\mathcal{T}$ has weight at most $M$, for each $u \in \mathsf{nodes}(\mathcal{T})$,*

- *if $w_u \geq (1 + \alpha) \cdot \tau$, then $w'_u = \top$;*
- *if $w_u < (1 - \alpha) \cdot \tau$, then $w'_u = \perp$;*
- *otherwise (i.e., $(1 - \alpha) \cdot \tau \leq w_u < (1 + \alpha) \cdot \tau$), $w'_u$ can be arbitrary.*

The algorithm for Problem 24 is simply running $\texttt{Classification}(\mathcal{T}; M, \eta, \alpha, \tau, \varepsilon, \delta)$ for each $\mathcal{T} \in \mathcal{F}$. Since the trees are disjoint, the privacy bound follows from Fact 12. Therefore we omit the proof and summarize the following.

▶ **Corollary 25.** *There is an $(\varepsilon, \delta)$-DP algorithm $\texttt{Classification}(\mathcal{F}; M, \eta, \alpha, \tau, \varepsilon, \delta)$ such that it solves Problem 24 assuming $\tau \geq \sqrt{\frac{2M}{\alpha\varepsilon}} \cdot \max\left\{\sqrt{48\ln\left(\frac{2d}{\eta}\right)}, \sqrt{6\ln\left(1+\frac{e^{\varepsilon/2}-1}{\delta}\right)}\right\}$.*

## 4.2 A Reduction from Estimation to Classification

Now we present the reduction algorithm from the estimation problem (i.e., Problem 18) to the classification problem (i.e., Problem 24). The reduction here is given with large flexibility for choosing parameters. Later we will design geometric convergent sequences for simplicity of calculation and derive the final bounds.

▶ **Lemma 26.** *Let $\ell \geq 1$ be an integer. Let $M, M_0$, and $(M_i, \eta_i, \alpha_i, \tau_i, \varepsilon_i, \delta_i)_{i\in[\ell]}$ be a sequence of parameters. There is an $(\varepsilon, \delta)$-DP algorithm $\texttt{Reduction}(\mathcal{T}; \ell, (M_i, \eta_i, \alpha_i, \tau_i, \varepsilon_i, \delta_i)_{i\in[\ell]})$, where $(\varepsilon, \delta) = \left(\sum_{i=1}^{\ell}\varepsilon_i, \sum_{i=1}^{\ell}\delta_i\right)$ such that it solves Problem 18 by carefully combining results from $\texttt{Classification}(\cdot; M_i, \eta_i, \alpha_i, \tau_i, \varepsilon_i, \delta_i)$'s and assuming the weight of the root of $\mathcal{T}$ is at most $M$ and*

$$\tau_i \;\geq\; \sqrt{\tfrac{2M_i}{\alpha_i\varepsilon_i}} \cdot \max\left\{\sqrt{48\ln\left(\tfrac{2d}{\eta_i}\right)}, \sqrt{6\ln\left(1+\tfrac{e^{\varepsilon_i/2}-1}{\delta_i}\right)}\right\} \quad \forall i \in [\ell], \tag{2}$$

$$\eta \;\geq\; \sum_{i=1}^{\ell}\eta_i, \tag{3}$$

$$M_i \;\geq\; (1+\alpha_{i+1})\cdot\tau_{i+1} \quad \forall i = 0, 1, \ldots, \ell-1 \quad \textit{and} \quad M_\ell \;\geq\; M, \tag{4}$$

$$(1-\alpha_i)\cdot\tau_i \;\leq\; M_i \;\leq\; (1+\alpha)(1-\alpha_i)\cdot\tau_i \quad \forall i \in [\ell], \tag{5}$$

$$0 \;\leq\; M_0 \;\leq\; \alpha\cdot\tau_{\mathit{min}}. \tag{6}$$

The reduction algorithm is formalized in Algorithm 2 and analyzed in the full version.

**Algorithm 2** Reduction.

**Input:** $\mathcal{T}$ and parameters $\ell, M_0, (M_i, \eta_i, \alpha_i, \tau_i, \varepsilon_i, \delta_i)_{i\in[\ell]}$ described above
**Output:** $\widetilde{w}_u \in \mathbb{R}$ for all node $u \in \mathsf{nodes}(\mathcal{T})$

Initialize $\mathcal{F}_\ell \leftarrow \{\mathcal{T}\}$
**foreach** $i = \ell$ **to** 1 **do**
  Initialize $\mathcal{F}_{i-1} \leftarrow \emptyset$
  Compute $\boldsymbol{w}' \leftarrow \texttt{Classification}(\mathcal{F}_i; M_i, \eta_i, \alpha_i, \tau_i, \varepsilon_i, \delta_i)$
  For each node $u$ in $\mathcal{F}_i$, let $\mathcal{T}_u$ be the subtree of $u$ in $\mathcal{F}_i$
  **foreach** *node $u$ satisfying $w'_u = \top$ and $w'_v = \bot$ for all $v \in \mathsf{nodes}(\mathcal{T}_u)\setminus\{u\}$* **do**
    Set $\widetilde{w}_v \leftarrow M_i$ for each $u$'s ancestor $v$ (including $u$ itself) in $\mathcal{T}$
    Update $\mathcal{F}_{i-1} \leftarrow \mathcal{F}_{i-1} \cup \{\mathcal{T}_1, \mathcal{T}_2, \ldots\}$ where $\mathcal{T}_1, \mathcal{T}_2, \ldots$ are the disjoint trees of $\mathcal{T}_u \setminus \{u\}$
  **end**
**end**
Set $\widetilde{w}_v \leftarrow M_0$ for each node $v$ in $\mathcal{F}_0$

## 4.3 Putting Everything Together

Now we give the algorithm for Problem 18. To this end, we carefully choose parameters and apply Lemma 26, where the required upper bound $M$ is privately estimated with Lemma 16.

▶ **Corollary 27** (Proof in Full Version). *There is an $(\varepsilon, \delta)$-DP algorithm* `Estimation`$(\mathcal{T}, \alpha, \varepsilon, \delta, \eta)$ *such that it solves Problem 18 assuming*

$$\tau_{min} \geq \frac{324 \cdot (1+\alpha)^2}{\alpha^4 \cdot \varepsilon} \cdot \max\left\{8 \ln\left(\frac{4d}{\eta}\right), \ln\left(1 + \frac{2 \cdot \left(e^{\varepsilon/4} - 1\right)}{\delta}\right)\right\}.$$

Now we complete the proof of Theorem 21 using Lemma 20 and Corollary 27.

**Proof of Theorem 21.** We first note that if $\alpha \geq 1$ then the $\alpha$-mRMSE is trivially zero by outputting the all-zeros vector. Therefore we assume without loss of generality $\alpha \in (0, 1)$.

Let $C > 0$ be a constant to be optimized later. Fix $\eta = d^{-2}$ and $\tau = \frac{C}{\alpha^4} \cdot \left(\frac{1}{\varepsilon} \log\left(\frac{2d}{\delta}\right) + 1\right)$. Let $\varepsilon' = \varepsilon/2$ and $\delta' = \delta/2$. Since $\alpha \in (0, 1)$, $d \geq 1$, and $\delta \in (0, 1]$, we have

$$\tau^* := \frac{324 \cdot (1+\alpha)^2}{\alpha^4 \cdot \varepsilon'} \cdot \max\left\{8 \ln\left(\frac{4d}{\eta}\right), \ln\left(1 + \frac{2 \cdot \left(e^{\varepsilon'/4} - 1\right)}{\delta'}\right)\right\} \leq O\left(\frac{1}{\alpha^4} \cdot \left(\frac{1}{\varepsilon} \log\left(\frac{2d}{\delta}\right) + 1\right)\right).$$

We set $C$ large enough such that $\tau \geq \tau^*$. By Corollary 27, there is an $(\varepsilon', \delta') = (\varepsilon/2, \delta/2)$-DP algorithm for Problem 18 when $\tau_u \equiv \tau$ for all $u \in \mathsf{nodes}(\mathcal{T})$.

The desired $\alpha$-mRMSE bound now follows from Lemma 20 and the parameters above. ◀

# 5 Lower Bounds

In this section we prove lower bounds for DP tree aggregation algorithms. In particular, Theorem 28 proves pure-DP lower bounds for all $\alpha$-mRMSE whenever $\alpha \in [0, 1)$; and Theorem 29 proves approximate-DP lower bounds for additive-only error, i.e., $(\alpha = 0)$-mRMSE.

▶ **Theorem 28** (Pure-DP Lower Bound). *Let $\alpha \in [0, 1)$ be a parameter. For any $\varepsilon > 0$, any $\varepsilon$-DP algorithm for tree aggregation on the complete depth-$d$ binary tree must incur $\alpha$-mRMSE at least $\Omega\left((1-\alpha)^2 \cdot d/\varepsilon\right)$.*

▶ **Theorem 29** (Approximate-DP Lower Bound for $\alpha = 0$). *For any $\varepsilon > 0$ and any $\delta > 0$ sufficiently small depending on $\varepsilon$, there is a constant $C_{\varepsilon,\delta} > 0$ that any $(\varepsilon, \delta)$-DP algorithm for tree aggregation on the complete depth-$d$ binary tree must incur mRMSE at least $C_{\varepsilon,\delta} \cdot \sqrt{d}$.*

Theorem 28 is proved in Section 5.1. The proof of Theorem 29 relies on results from [24] and the factorization norm of the binary tree matrix, which we defer to the full version.

## 5.1 Pure-DP Lower Bound

To prove Theorem 28, by Lemma 19 it suffices to rule out DP algorithms for Problem 18 with small thresholds. Since Problem 18 is interesting on its own, we will present its lower bound in the approximate-DP setting for full generality.

▶ **Lemma 30.** *Assume $\mathcal{T}$ in Problem 18 is a complete binary tree of depth $d$. Let $D = 2 \cdot \lceil \tau_{max}/(1-\alpha) \rceil$. If $\mathcal{A}$ is an $(\varepsilon, \delta)$-DP algorithm for Problem 18 and suppose $\eta \leq 1/8$ and*

$$\delta \cdot \frac{e^{\varepsilon \cdot D} - 1}{e^{\varepsilon} - 1} \leq \frac{1}{8} \cdot 2^{-(d-1) \cdot \mathcal{H}(4\eta)}, \tag{7}$$

*then*

$$\tau_{max} = \Omega\left((1-\alpha)\cdot(d-3-(d-1)\cdot\mathcal{H}(4\eta))/\varepsilon\right),$$

*where* $\mathcal{H}(x) = x\log(1/x) + (1-x)\log(1/(1-x))$ *is the binary entropy function.*

**Proof.** We define input datasets $\boldsymbol{x}^1, \ldots, \boldsymbol{x}^{2^d}$ where $\boldsymbol{x}^i$ assigns $D/2$ to the $i$th leaf and 0 to the remaining leaves. Let $\mathcal{P}_i$ be the path from root to the $i$th leaf. We define a randomized decoding algorithm $\mathsf{Dec}$ as follows:

- $\mathsf{Dec}$ takes the output $\widetilde{\boldsymbol{w}}$ of $\mathcal{A}$ as input and starts from the root of $\mathcal{T}$.
- Assume $\mathsf{Dec}$ is at node $u \in \mathsf{nodes}(\mathcal{T})$.
  - If $u$ is a leaf, then output the index of $u$ among all the leaves.
  - Otherwise let $u_0, u_1$ be the children of $u$ and we divide into the following cases.
    - If $\widetilde{w}_{u_0} \geq \tau_{\mathsf{max}}$ and $\widetilde{w}_{u_1} \geq \tau_{\mathsf{max}}$, then we move to $u_0$ or $u_1$ with equal probability.
    - If $\widetilde{w}_{u_0} < \tau_{\mathsf{max}}$ and $\widetilde{w}_{u_1} < \tau_{\mathsf{max}}$, then we move to $u_0$ or $u_1$ with equal probability.
    - Otherwise let $p \in \{0,1\}$ be such that $\widetilde{w}_{u_p} \geq \tau_{\mathsf{max}}$ and $\widetilde{w}_{u_{1-p}} < \tau_{\mathsf{max}}$, then we move to $u_p$ with probability $\kappa$ and to $u_{1-p}$ with probability $1-\kappa$, where $\kappa = 1-4\eta \in [1/2, 1]$.

Now we fix an index $i \in [2^d]$. Let $\mathcal{P}_i$ be $u_1, \ldots, u_d$. Then for each $j \in [d-1]$, let $u_j^0, u_j^1$ be the children of $u_j$ and assume without loss of generality $u_j^0 = u_{j+1}$; then we define the following indicators:

- $a_j = \mathbb{I}[(\widetilde{w}_{u_j^0} \geq \tau_{\mathsf{max}} \text{ and } \widetilde{w}_{u_j^1} \geq \tau_{\mathsf{max}}) \text{ or } (\widetilde{w}_{u_j^0} < \tau_{\mathsf{max}} \text{ and } \widetilde{w}_{u_j^1} < \tau_{\mathsf{max}})]$.
- $b_j = \mathbb{I}[\widetilde{w}_{u_j^0} \geq \tau_{\mathsf{max}} \text{ and } \widetilde{w}_{u_j^1} < \tau_{\mathsf{max}}]$.
- $c_j = \mathbb{I}[\widetilde{w}_{u_j^0} < \tau_{\mathsf{max}} \text{ and } \widetilde{w}_{u_j^1} \geq \tau_{\mathsf{max}}]$.

Let $A = \sum_j a_j$, $B = \sum_j b_j$, and $C = \sum_j c_j$. Then it is easy to see $A + B + C = d - 1$ and

$$\mathbf{Pr}\left[\mathsf{Dec}(\mathcal{A}(\boldsymbol{x}^i)) = i\right] = \mathbb{E}\left[2^{-A}\kappa^B(1-\kappa)^C\right] = \kappa^{d-1}\,\mathbb{E}\left[1/(2\kappa)^{d-1-B}(2-2\kappa)^C\right]. \tag{8}$$

Now we further fix the input to be $\boldsymbol{x}^i$ defined above. Then $w_u = D/2$ for $u \in \mathcal{P}_i$ and $w_u = 0$ if otherwise. For $p \in \{0,1\}$, consider the event $\mathcal{E}_j^p$: $\left|\widetilde{w}_{u_j^p} - w_{u_j^p}\right| \leq \alpha \cdot \max\left\{w_{u_j^p}, \tau_{u_j^p}\right\}$. Hence when $\mathcal{E}_j^0$ happens, we have

$$\widetilde{w}_{u_j^0} \geq w_{u_j^0} - \alpha \cdot \max\left\{w_{u_j^0}, \tau_{u_j^0}\right\} \geq D/2 - \alpha \cdot \max\{D/2, \tau_{\mathsf{max}}\} = (1-\alpha)\cdot D/2 \geq \tau_{\mathsf{max}}.$$

Similarly when $\mathcal{E}_j^1$ happens, we have $\widetilde{w}_{u_j^1} \leq w_{u_j^1} + \alpha \cdot \max\left\{w_{u_j^1}, \tau_{u_j^1}\right\} \leq \alpha \cdot \tau_{\mathsf{max}} < \tau_{\mathsf{max}}$. Meanwhile by the definition of Problem 18, we know $\mathbf{Pr}\left[\mathcal{E}_j^p\right] \geq 1 - \eta$. Thus

$$\mathbf{Pr}\left[b_j = 1\right] \geq \mathbf{Pr}\left[\mathcal{E}_j^0 \wedge \mathcal{E}_j^1\right] \geq 1 - 2\eta, \quad \text{and} \quad \mathbf{Pr}\left[c_j = 1\right] \leq \mathbf{Pr}\left[\neg\mathcal{E}_j^0 \wedge \neg\mathcal{E}_j^1\right] \leq \mathbf{Pr}\left[\neg\mathcal{E}_j^0\right] \leq \eta,$$

which implies $\mathbb{E}\left[d - 1 - B\right] \leq 2\eta \cdot (d-1)$ and $\mathbb{E}\left[C\right] \leq \eta \cdot (d-1)$. Note that $d - 1 - B \geq 0$. Define the event $\mathcal{E}$: $d - 1 - B \leq 4\eta \cdot (d-1)$ and $C \leq 4\eta \cdot (d-1)$. Then by Markov's inequality and a union bound, we have $\mathbf{Pr}\left[\mathcal{E}\right] \geq 1 - 1/2 - 1/4 = 1/4$. Plugging into (8), we have

$$\begin{aligned}
\mathbf{Pr}\left[\mathsf{Dec}(\mathcal{A}(\boldsymbol{x}^i)) = i\right] &\geq \kappa^{d-1}/4 \cdot \mathbb{E}\left[1/(2\kappa)^{d-1-B}(2-2\kappa)^C \;\middle|\; \mathcal{E}\right] \\
&\geq \kappa^{d-1}/4 \cdot 1/(2\kappa)^{4\eta\cdot(d-1)} \cdot (2-2\kappa)^{4\eta\cdot(d-1)} && \text{(since } \kappa \in [1/2,1]\text{)} \\
&= \frac{1}{4} \cdot 2^{-(d-1)\cdot\mathcal{H}(4\eta)}. && \text{(setting } \kappa = 1-4\eta \in [1/2,1]\text{)}
\end{aligned}$$

Since $\sum_{i'} \mathbf{Pr}\left[\mathsf{Dec}(\mathcal{A}(\boldsymbol{x}^i)) = i'\right] = 1$, by an averaging argument there exists an $i^*$ such that $\mathbf{Pr}\left[\mathsf{Dec}(\mathcal{A}(\boldsymbol{x}^i)) = i^*\right] \leq 2^{-d}$. Since $\left\|\boldsymbol{x}^i - \boldsymbol{x}^{i^*}\right\|_1 \in \{0, D\}$, by Fact 10 we have

$$\mathbf{Pr}\left[\mathsf{Dec}(\mathcal{A}(\boldsymbol{x}^{i^*})) = i^*\right] \leq \mathbf{Pr}\left[\mathsf{Dec}(\mathcal{A}(\boldsymbol{x}^i)) = i^*\right] \cdot e^{\varepsilon\cdot D} + \delta \cdot \tfrac{e^{\varepsilon\cdot D}-1}{e^\varepsilon - 1} \leq 2^{-d} \cdot e^{\varepsilon\cdot D} + \delta \cdot \tfrac{e^{\varepsilon\cdot D}-1}{e^\varepsilon - 1}.$$

In all, we have

$$\frac{1}{4} \cdot 2^{-(d-1)\cdot\mathcal{H}(4\eta)} \leq \mathbf{Pr}\left[\mathsf{Dec}(\mathcal{A}(\boldsymbol{x}^{i^*})) = i^*\right] \leq 2^{-d} \cdot e^{\varepsilon \cdot D} + \delta \cdot \frac{e^{\varepsilon \cdot D} - 1}{e^{\varepsilon} - 1},$$

which proves the bound after plugging in Assumption (7) and rearranging the terms. ◀

To deal with general $\eta$ from Problem 18, we simply run independent copies of the algorithm to decrease the error probability.

▶ **Corollary 31** (Proof in Full Version). *Assume $\mathcal{T}$ in Problem 18 is a complete binary tree of depth $d$. Define $D = 2 \cdot \left\lceil \frac{\tau_{max}}{1-\alpha} \right\rceil$ and $s = \left\lceil \frac{\ln(4/\kappa)}{2\cdot(1/2-\eta)^2} \right\rceil$ for any parameter $\kappa \in (0, 1/2]$. If $\mathcal{A}$ is an $(\varepsilon, \delta)$-DP algorithm for Problem 18 and suppose $s \cdot \delta \cdot \frac{e^{s\cdot\varepsilon\cdot D}-1}{e^{s\cdot\varepsilon}-1} \leq \frac{1}{8} \cdot 2^{-(d-1)\cdot\mathcal{H}(\kappa)}$, then*

$$\tau_{max} = \Omega\left(\frac{(1-\alpha)\cdot(1/2-\eta)^2\cdot(d-3-(d-1)\mathcal{H}(\kappa))}{\varepsilon\cdot\ln(4/\kappa)}\right).$$

▶ Remark 32 (General Tree Structures). The proof above works almost identically for binary tree $\mathcal{T}$ that is not necessarily complete, where the bound is simply replacing $d-3$ with $\log(|\mathsf{leaves}(\mathcal{T})|/8)$. To deal with general arity $k$, we can embed a binary tree $\mathcal{T}'$ into $\mathcal{T}$ where we say $\mathcal{T}$ *embeds* a tree $\mathcal{T}'$ if we can obtain $\mathcal{T}'$ from $\mathcal{T}$ by deleting nodes and edges. Then we can ignore nodes in $\mathsf{nodes}(\mathcal{T}) \setminus \mathsf{nodes}(\mathcal{T}')$ and obtain lower bounds for $\mathcal{T}'$.

Now we are ready to establish Theorem 28 using Lemma 19 and Corollary 31.

**Proof of Theorem 28.** Let $\mathcal{T}$ be the complete binary tree of depth $d$. Let $C, \tau, \eta$ be parameters to be optimized later. We consider Problem 18 where $\tau_u \equiv \tau$ for all $u \in \mathsf{nodes}(\mathcal{T})$.

Assume towards contradiction that there is an $\varepsilon$-DP tree aggregation algorithm with $\alpha$-mRMSE at most $C \cdot (1-\alpha)^2 d/\varepsilon$. Then by Lemma 19, the algorithm is also $(\alpha', \eta)$-accurate for Problem 18 if $\alpha' > \alpha$ and $(\alpha' - \alpha)\sqrt{\eta} \cdot \tau \leq C \cdot (1-\alpha)^2 d/\varepsilon$.

Now we set $\eta = 1/4$ and apply Corollary 31 with $\delta = 0, \kappa = 1/4$. This gives a lower bound $\tau = \Omega\left((1-\alpha') \cdot d/\varepsilon\right)$, which means $\frac{C\cdot(1-\alpha)^2\cdot d}{\varepsilon} \geq \Omega\left(\frac{(\alpha'-\alpha)(1-\alpha')\cdot d}{\varepsilon}\right)$. Then we set $\alpha' = (1+\alpha)/2 > \alpha$ and $C = O(1)$ small enough to derive a contradiction. ◀

▶ Remark 33 ($\log(1/\delta)$ Factor in the Approximate-DP Algorithm). As mentioned in Section 1.1, our improved $(\varepsilon, \delta)$-DP algorithm (See Theorem 8) has a $\log(1/\delta)$ factor, which is worse than the $\sqrt{\log(1/\delta)}$ factor in the Gaussian mechanism (see Corollary 15). One may wonder if we can further improve the dependency on $\delta$ to, say, $\sqrt{\log(1/\delta)}$, without influencing the other parameters. Combining Corollary 31, we show this is in some sense impossible.

Consider the complete binary tree of depth $d$. Let $\delta = 2^{-\Omega(d)}$. We consider the case where $\alpha, \eta$ are constants and all the $\tau_u$'s are equal to $\tau$. In the approximate-DP setting, we naturally seek bounds better than the ones in the pure-DP setting (recall we can obtain $\tau = O(d/\varepsilon)$ from Corollary 15). Thus we assume $\tau = O(d/\varepsilon)$ in advance.

Then for a suitable choice of $\kappa = \Theta(1)$, the condition in Corollary 31 holds, which gives a lower bound $\tau = \Omega(d/\varepsilon)$ for Problem 18. Then by Lemma 19, this rules out the possibility of improving the dependency on $\delta$ in Theorem 8 without worsening the dependency on $d$.

## 6 Conclusions

We study the problem of privately estimating counts in hierarchical data, and give several algorithms and lower bounds. We propose a new error measure that takes the multiplicative error into account. The commonly used $\ell_2^2$-error measure in evaluating utilities of DP

mechanisms allows some queries to have huge error. On the other hand, the standard measure $\ell_\infty$-error has a "union bound issue" on particularly long output vector (which is the case in Census and Ads applications).

To mitigate these weaknesses, we propose $\alpha$-multiplicative root mean squared error ($\alpha$-mRMSE). Then we examine the standard Laplace mechanism for pure-DP and Gaussian mechanism for approximate-DP, and prove their optimality. Informally, we show Laplace mechanism already achieves optimal bounds in the pure-DP setting for all multiplicative factor $\alpha$ and Gaussian mechanism is optimal in the approximate-DP setting when $\alpha = 0$ (i.e., additive-only error).

For the remaining case where we allow $\alpha > 0$ and an approximate-DP algorithm, we design a new algorithm with exponential improvements over Gaussian mechanism. More precisely, Gaussian mechanism incurs $\alpha$-mRMSE of $O_{\alpha,\varepsilon,\delta}(\sqrt{d})$ while our algorithm gives improved bounds of $O_{\alpha,\varepsilon,\delta}(\log(d))$ (and $O\left(\frac{1}{\alpha^3} \cdot \left(\frac{1}{\varepsilon}\log\left(\frac{2d}{\delta}\right)+1\right)\right)$ specifically). It remains an interesting question if the dependency on $d$ or $\alpha$ can be improved further. Indeed, current lower bounds do not preclude bounds of the form $O_{\varepsilon,\delta}\left(\frac{1}{\alpha}\cdot\log^*(d)\right)$ or even $O_{\varepsilon,\delta}(1/\alpha)$.

Throughout this work, we assumed that the entries of the input $\boldsymbol{x}$ are non-negative. Another interesting direction is to extend the study to the case where the entries of $\boldsymbol{x}$ can be negative. Here, the multiplicative error would be with respect to the absolute value of the true answer. Our algorithms do not apply here and it is unclear whether allowing a multiplicative error can help reduce the additive error in this setting.

## References


1 John Abowd, Daniel Kifer, Brett Moran, Robert Ashmead, Philip Leclerc, William Sexton, Simson Garfinkel, and Ashwin Machanavajjhala. Census topdown: Differentially private data, incremental schemas, and consistency with public knowledge, 2019. Available at https://github.com/uscensusbureau/census2020-das-e2e/blob/master/doc/20190711_0945_Consistency_for_Large_Scale_Differentially_Private_Histograms.pdf.

2 John M. Abowd, Robert Ashmead, Ryan Cumings-Menon, Simson L. Garfinkel, Micah Heineck, Christine Heiss, Robert Johns, Daniel Kifer, Philip Leclerc, Ashwin Machanavajjhala, Brett Moran, William Sexton, Matthew Spence, and Pavel Zhuravlev. The 2020 census disclosure avoidance system TopDown algorithm. *Harvard Data Sci. Rev.*, 2022. Special Issue 2.

3 John M Abowd and Ian M Schmutte. An economic analysis of privacy protection and statistical accuracy as social choices. *Amer. Econ. Rev.*, 109(1):171–202, 2019.

4 Apple Differential Privacy Team. Learning with privacy at scale. *Apple ML J.*, 2017.

5 Aditya Bhaskara, Daniel Dadush, Ravishankar Krishnaswamy, and Kunal Talwar. Unconditional differentially private mechanisms for linear queries. In *STOC*, pages 1269–1284, 2012.

6 Jaroslaw Blasiok, Mark Bun, Aleksandar Nikolov, and Thomas Steinke. Towards instance-optimal private query release. In *SODA*, pages 2480–2497, 2019.

7 Jean Bolot, Nadia Fawaz, Shanmugavelayutham Muthukrishnan, Aleksandar Nikolov, and Nina Taft. Private decayed predicate sums on streams. In *ICDT*, pages 284–295, 2013.

8 Mark Bun, Kobbi Nissim, Uri Stemmer, and Salil P. Vadhan. Differentially private release and learning of threshold functions. In *FOCS*, pages 634–649, 2015.

9 Mark Bun, Jonathan R. Ullman, and Salil P. Vadhan. Fingerprinting codes and the price of approximate differential privacy. *SIAM J. Comput.*, 47(5):1888–1938, 2018. doi:10.1137/15M1033587.

10 T.-H. Hubert Chan, Elaine Shi, and Dawn Song. Private and continual release of statistics. *ACM Trans. Inf. Syst. Secur.*, 14(3):26:1–26:24, 2011. doi:10.1145/2043621.2043626.

11 Aloni Cohen, Moon Duchin, J. N. Matthews, and Bhushan Suwal. Census topdown: The impacts of differential privacy on redistricting. In *FORC*, pages 5:1–5:22, 2021.

12 Aloni Cohen, Moon Duchin, JN Matthews, and Bhushan Suwal. Private Numbers in Public Policy: Census, Differential Privacy, and Redistricting. *Harvard Data Sci. Rev.*, 2022.

13 Graham Cormode, Cecilia M. Procopiuc, Divesh Srivastava, Entong Shen, and Ting Yu. Differentially private spatial decompositions. In *ICDE*, pages 20–31, 2012.

14 Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. Collecting telemetry data privately. In *NeurIPS*, pages 3571–3580, 2017.

15 Irit Dinur and Kobbi Nissim. Revealing information while preserving privacy. In *PODS*, pages 202–210, 2003.

16 Cynthia Dwork. Differential privacy: A survey of results. In *TAMC*, pages 1–19, 2008.

17 Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *EUROCRYPT*, pages 486–503, 2006.

18 Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam D. Smith. Calibrating noise to sensitivity in private data analysis. *J. Priv. Confidentiality*, 7(3):17–51, 2016. doi:10.29012/jpc.v7i3.405.

19 Cynthia Dwork, Frank McSherry, and Kunal Talwar. The price of privacy and the limits of LP decoding. In *STOC*, pages 85–94, 2007.

20 Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N. Rothblum. Differential privacy under continual observation. In *STOC*, pages 715–724, 2010. doi:10.1145/1806689.1806787.

21 Cynthia Dwork, Moni Naor, Omer Reingold, and Guy N. Rothblum. Pure differential privacy for rectangle queries via private partitions. In *ASIACRYPT*, pages 735–751, 2015.

22 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.

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

24 Alexander Edmonds, Aleksandar Nikolov, and Jonathan R. Ullman. The power of factorization mechanisms in local and central differential privacy. In *STOC*, pages 425–438, 2020.

25 Alessandro Epasto, Jieming Mao, Andres Munoz Medina, Vahab Mirrokni, Sergei Vassilvitskii, and Peilin Zhong. Differentially private continual releases of streaming frequency moment estimations. In *ITCS*, pages 48:1–48:24, 2023.

26 Hendrik Fichtenberger, Monika Henzinger, and Wolfgang Ost. Differentially private algorithms for graphs under continual observation. In *ESA*, pages 42:1–42:16, 2021.

27 Quan Geng, Wei Ding, Ruiqi Guo, and Sanjiv Kumar. Tight analysis of privacy and utility tradeoff in approximate differential privacy. In *AISTATS*, pages 89–99, 2020.

28 Moritz Hardt and Kunal Talwar. On the geometry of differential privacy. In *STOC*, pages 705–714, 2010.

29 Monika Henzinger, Jalaj Upadhyay, and Sarvagya Upadhyay. Almost tight error bounds on differentially private continual counting. In *SODA*, pages 5003–5039, 2023.

30 James Honaker. Efficient use of differentially private binary trees. In *TPDP*, 2015.

31 Peter Kairouz, Brendan McMahan, Shuang Song, Om Thakkar, Abhradeep Thakurta, and Zheng Xu. Practical and private (deep) learning without sampling or shuffling. In *ICML*, pages 5213–5225, 2021.

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

33 Troy Lee, Adi Shraibman, and Robert Spalek. A direct product theorem for discrepancy. In *CCC*, pages 71–80, 2008.

34 Chao Li, Gerome Miklau, Michael Hay, Andrew McGregor, and Vibhor Rastogi. The matrix mechanism: optimizing linear counting queries under differential privacy. *VLDB J.*, 24(6):757–781, 2015. doi:10.1007/s00778-015-0398-x.

35 Roy Mathias. The hadamard operator norm of a circulant and applications. *SIAM J. Matr. Anal. Appl.*, 14(4):1152–1167, 1993.

**36** Jiří Matoušek, Aleksandar Nikolov, and Kunal Talwar. Factorization norms and hereditary discrepancy. *Intl. Math. Res. Not.*, 2020(3):751–780, 2018.

**37** Aleksandar Nikolov. An improved private mechanism for small databases. In *ICALP*, pages 1010–1021, 2015.

**38** Aleksandar Nikolov. Private query release via the Johnson–Lindenstrauss transform. In *SODA*, pages 4982–5002, 2023.

**39** Aleksandar Nikolov, Kunal Talwar, and Li Zhang. The geometry of differential privacy: the sparse and approximate cases. In *STOC*, pages 351–360, 2013.

**40** Wahbeh H. Qardaji, Weining Yang, and Ninghui Li. Differentially private grids for geospatial data. In *ICDE*, pages 757–768, 2013.

**41** Salil P. Vadhan. The complexity of differential privacy. In Yehuda Lindell, editor, *Tutorials on the Foundations of Cryptography*, pages 347–450. Springer International Publishing, 2017. doi:10.1007/978-3-319-57048-8_7.

**42** Yonghui Xiao, Li Xiong, Liyue Fan, Slawomir Goryczka, and Haoran Li. DPCube: Differentially private histogram release through multidimensional partitioning. *Trans. Data Priv.*, 7(3):195–222, 2014.

**43** Jun Zhang, Xiaokui Xiao, and Xing Xie. Privtree: A differentially private algorithm for hierarchical decompositions. In *SIGMOD*, pages 155–170, 2016.