Differentially Private Release of Sparse and Skewed Data

Christian Janos Lebeda

Advisors: Rasmus Pagh and Martin Aumüller
Submitted: August 2023

IT UNIVERSITY OF COPENHAGEN

Abstract

In this thesis we study *differential privacy*. Differential privacy allows us to give formal privacy guarantees of mechanisms for statistical query release by quantifying privacy loss. However, achieving a good tradeoff between privacy and utility can be challenging. Our focus in this thesis is on settings where data is either sparse or skewed. The main contributions of this thesis are:

- We introduce a differentially private data structure for representing sparse vectors. The difficulty of this problem lies in designing a data structure with low error guarantees, low space complexity, and fast random access to all entries. Previous techniques achieve only two of these three goals simultaneously. Our data structure is the first mechanism with good guarantees in all three metrics.
- We introduce a mechanism for releasing a Misra-Gries sketch under differential privacy. The Misra-Gries sketch is commonly used in the non-private setting to compute approximate histograms and find heavy hitters in a data stream. We present an improved analysis of the structure of the sketch which allows us to add significantly less noise compared to the previous state-of-the-art. The error guarantees of our mechanism are optimal up to small constants for both pure and approximate differential privacy.
- We present a simple technique for splitting up the privacy budget when answering multiple queries with the Gaussian or Laplace mechanisms. We show that this technique outperforms standard approaches for a class of error measures when the sensitivities of the queries are skewed. We use this technique in the design of a mechanism for differentially private mean estimation. We give theoretical error guarantees of our mechanism under a concentration assumption and perform experiments with both synthetic and real-world datasets.

Resumé

I denne afhandling undersøger vi *differentiel privathed*[1]. Differentiel privathed lader os give formelle privathed garantier af mekanismer til offentliggørelse af statistiske data ved at kvantificere privathedstabet. Det kan dog være udfordrende at opnå en god balance mellem privathed og brugbarhed. Vores fokus i denne afhandling er på situationer, hvor data enten er tyndt eller skævt fordelt. De vigtigste bidrag i denne afhandling er:

- Vi introducerer en differentiel privat datastruktur, som kan repræsentere tynde vektorer. Udfordringen i dette problem ligger i at designe en datastruktur med gode fejlgarantier, lav hukommelseskompleksitet og hurtig vilkårlig adgang til indgangene i vektoren. Tidligere teknikker opnår kun to af disse tre mål samtidigt. Vores datastruktur er den første mekanisme med gode garantier for alle tre mål.
- Vi introducerer en mekanisme til at frigive en Misra-Gries skitse under differentiel privathed. Den ikke-private version af Misra-Gries skitsen bruges ofte til at beregne approksimative histogrammer og finde ofte forekommende elementer i en datastrøm. Vi præsenterer en forbedret analyse af skitsens struktur, hvilket vi udnytter til at tilføje betydeligt mindre støj end den tidligere bedste metode. Fejlgarantierne for vores mekanisme er optimale indenfor små konstanter for både ægte og approksimativ differentiel privathed.
- Vi præsenterer en simpel teknik til at opdele privathedsbudgettet, når vi frigiver flere statistikker med Gauss eller Laplace mekanismerne. Vi viser, at denne teknik er bedre end standardmetoder for en klasse af fejl beskrivelser, når følsomheden af statistikkerne er skævt fordelt. Vi bruger denne teknik i designet af en mekanisme til differentiel privat gennemsnit estimering. Vi giver teoretiske fejlgarantier for vores mekanisme under en koncentrationsantagelse og udfører eksperimenter med både syntetiske og virkelige datasæt.

[1]Der findes ikke en udbredt dansk oversættelse af *differential privacy*. Vi har valgt at bruge ordet *privathed* her til at beskrive "tilstanden af at være privat".

Acknowledgements

I have been fortunate to have two excellent advisors throughout my PhD studies in Rasmus Pagh and Martin Aumüller. They are always willing to set aside time for discussing research ideas or providing general guidance. Rasmus and Martin complement each other's strengths as advisors well. I have learned a lot from working with them, and I am grateful to them for their role in my development as a researcher.

I would like to thank my many great colleagues in the Algorithms Group at ITU and the BARC corridors at DIKU for creating a pleasant work environment. It is important to me that I have a workplace where I enjoy spending my time. During my PhD I had the luxury of having two.

During parts of the 2023 Winter and Spring, I visited the University of Waterloo where I was hosted by Gautam Kamath. I would like to thank Gautam, his research group The Salon, and the other PhD students in the algorithms lab for being welcoming and making my stay a memorable experience.

Lastly, I would like to thank my friends and family for supporting me through the ups and downs of my PhD studies. A special mention to my childhood friend, Martin Edvardsen, who first suggested we apply to the software development study program at the IT University of Copenhagen 8 years ago.

Contents

1 Introduction 1
- 1.1 Motivation and background . . . 1
- 1.2 Differential Privacy . . . 4
- 1.3 Contribution and overview . . . 16

2 Representing Sparse Vectors with Differential Privacy, Low Error, Optimal Space, and Fast Access 21
- 2.1 Introduction . . . 22
- 2.2 Preliminaries . . . 25
- 2.3 Related work . . . 29
- 2.4 The ALP mechanism . . . 31
- 2.5 Combined data structure . . . 44
- 2.6 Faster evaluation with multiplicative error . . . 51
- 2.7 Improvements for Sparse Integer-valued Vectors . . . 54
- 2.8 Constant Access Time with Optimal Expected Error . . . 57
- 2.9 Experiments . . . 59
- 2.10 Suggestions to Practitioners . . . 63
- 2.A Closed-form proof of Lemma 2.9 . . . 66

3 Better Differentially Private Approximate Histograms and Heavy Hitters using the Misra-Gries Sketch 67
- 3.1 Introduction . . . 68
- 3.2 Technical overview . . . 70
- 3.3 Preliminaries . . . 73
- 3.4 Related work . . . 74
- 3.5 Differentially Private Misra-Gries . . . 75
- 3.6 Pure Differential Privacy . . . 89
- 3.7 Privatizing merged sketches . . . 90

4 Differentially Private Vector Aggregation when Coordinates have Different Sensitivity 93
4.1 Introduction . . . 94
4.2 Preliminaries . . . 95
4.3 Calibrating elliptical Gaussian noise . . . 99
4.4 The Laplace mechanism . . . 103

5 PLAN: Variance-Aware Differentially Private Mean Estimation 107
5.1 Introduction . . . 108
5.2 Preliminaries . . . 111
5.3 Algorithm . . . 114
5.4 Analysis . . . 116
5.5 Examples of well-concentrated distributions . . . 122
5.6 Generic Bounds in the Absence of Variance estimates . . . 124
5.7 Empirical Evaluation . . . 126
5.8 Related Work . . . 133
5.9 Conclusion and future work . . . 135
5.A Useful statements from probability theory . . . 136
5.B Proof of Theorem 5.3 . . . 136
5.C Algorithms for Variance Estimation . . . 138
5.D Settings for Experiments . . . 142

6 Conclusion and Open Problems 143

Bibliography 145

Chapter 1

Introduction

1.1 Motivation and background

In our modern world, vast amounts of data are collected by many entities. We constantly interact with systems and devices that track our location, search history, health information, etc. The amount of generated data is increasing globally year by year. This data constitutes an extremely valuable resource. Data scientists can extract information such as the statistical properties of the collected data. This information is useful for many purposes. It can for example help improve existing technologies or form the basis for new ones. The recent emergence of popular publicly available generative AI tools is an excellent example of the potential of large-scale datasets.

However, this data collection also poses a threat to private and sensitive information. There are many examples in recent years of misuse. One such example is the Facebook-Cambridge Analytica scandal [The18] in which the personal data of millions of Facebook users was used without their consent with the intent of creating political advertisements. It is natural to raise privacy concerns over such misuse but privacy risks are present even when data is publicly released with well-meaning intentions.

In 2006 AOL made a dataset of 20 million search queries from 650.000 users publicly available [Arr06]. Such a dataset can be useful for researchers in several areas such as search optimization and network analysis. AOL made an effort towards anonymizing the data by replacing usernames with random ID numbers. This approach is called pseudonymization. The problem was that their approach was easily

broken. Search queries often contain personally identifiable information which allows an adversary to link an individual to the ID number after which they know the entire search history. Sometimes the attack is trivial because people often search for their names or the names of friends and families. It took just days for journalists from the NY Times to reveal the true identity of one such user [BZ06]. The dataset was removed from the AOL website, but it was too late. The then chief technology officer (CTO) of AOL resigned as a result of the scandal [Zel06].

Such scenarios set up a trade-off situation. On the one hand, the personal information of individuals should be protected. The simplest approach is of course to never collect any data. On the other hand, the advantages of publicly available datasets are undeniable. Many research fields rely heavily on understanding population data that might contain sensitive information. Such datasets come in many forms such as medical records or search query logs as discussed above. It is impossible to learn anything about a population without some risk to the privacy of the individuals that make up the population. This motivates the development of techniques for extracting statistical information about a dataset while protecting the privacy of the data subjects.

1.1.1 Privacy-preserving data release

In this thesis we consider the problem of answering statistical queries of a dataset while protecting the privacy of the individuals that make up the dataset. Such techniques can be used to safely release data publicly similar to the intent of AOL discussed above. But it can also be used internally by companies to e.g. learn how their users interact with the company's products without violating their privacy.

Throughout this chapter we use the following terms for the entities involved in a data release: The *data subjects* are the individuals whose data make up a private dataset. In the case of the AOL release the data subjects were users who had performed search queries. The *data publisher* is responsible for the privacy-preserving data release. This is the role of AOL. The *data analysts* perform some analysis on the private data release. A data analyst could be anything from a scientist to a policymaker looking to make an informed decision based on available data. In practice, the data publisher and data analyst are sometimes the same entity. In this thesis we consider differentially private mechanisms. These are tools that can be deployed by a data publisher to protect privacy when publishing statistics.

The challenge of privacy-preserving data release has existed for many years and several techniques exist that predate differential privacy. Pseudonymization is a simple technique for protecting the privacy of a dataset by removing personal identifiers from the data records. Unfortunately, it is often possible to re-identify data subjects by using side information. A famous example of this is when Sweeney [Swe15] showed that the medical records of the governor of Massachusetts could be uniquely identified by the combination of his publicly available birth date, gender, and ZIP code. One of the challenges of designing privacy-preserving tools is that privacy can be very unintuitive. Techniques that seem reasonable can sometimes be broken. It is important to remember that a public data release does not exist in a vacuum.

A stronger technique than pseudonymization is k-anonymity [Swe02]. The idea is to use less fine-grained pseudonyms such that there are at least k copies of all pseudonyms. This protects against the attack described above. We could for example remove the birth data such that a record is only identified by gender and ZIP code and we might no longer be able to uniquely identify the governor's record. However, it turns out that k-anonymity is vulnerable to composition attacks. Ganta, Kasiviswanathan, and Smith [GKS08] showed that it is sometimes possible to uniquely identify a record based on two separate k-anonymous datasets. Even in the paper introducing the framework Sweeney [Swe02] noted that attacks exist that can break the guarantees. These types of approaches are steps in the right direction but after a public data release the dataset is effectively available indefinitely. Therefore we want to give privacy guarantees that are not susceptible to these kinds of attacks.

1.1.2 United States Decennial Census

Throughout this chapter we reference the disclosure avoidance system (DAS) deployed by the United States Census Bureau. This refers specifically to the TopDown algorithm used during the 2020 US Decennial Census [AAC+22]. The Census Bureau conducts a census of the American population every 10 years. The decennial census is mandated by the Constitution of the United States and one of the primary purposes is to divide the seats in the House of Representatives among the states [Uni21a].

The Census Bureau has developed in-house techniques to protect the privacy of citizens used for the decennial census for many years. A few years ago they published a report detailing the progression of their

system from the 1970 to the 2010 Census [McK18]. From 1990 to 2010 they relied on *data swapping* for privacy protection. The technique was seen as appropriate at the time, but the Census Bureau has since designed an attack that could be used to re-identify large parts of the 2010 Census dataset [Uni21b]. For this reason the data swapping technique was replaced with a differentially private system for the 2020 Census [Uni23].

1.2 Differential Privacy

Differential Privacy (DP) is a framework that seeks to address some of the challenges discussed in the previous section. It was introduced by Dwork, McSherry, Nissim, and Smith [DMNS06]. Differential Privacy allows us to give formal privacy guarantees by restricting how much the output distribution of a mechanism is affected by any individual's data. Differential privacy is often introduced in a few sentences either as a technical definition or a vague description. But understanding the implications of the definition is non-trivial and the vague descriptions often fail at conveying the guarantees of the framework to end users [CKR21]. Before introducing the exact definition we therefore discuss some core concepts in the framework of differential privacy. In doing so we also argue that these concepts are useful for analyzing privacy-preserving mechanisms.

When discussing the concepts in this section we consider a *dataset*, denoted $x \in \mathcal{U}^{\mathbb{N}}$, to be a collection of records from some universe $\mathcal{U}$. Each record contains the data about one individual and we want to perform some arbitrary query $\mathcal{M} : \mathcal{U}^{\mathbb{N}} \to \mathcal{R}$ on the dataset.

1.2.1 Neighboring datasets

A central concept in differential privacy is the definition of neighboring datasets. Intuitively, one can think of two datasets as neighboring if they are identical except the data from one individual is in one dataset but not the other. We denote a pair of neighboring datasets as $x \sim x'$.

The exact definition for neighboring datasets depends on the setting and the query we want to answer. The two most common variants of the definition are add/remove (sometimes referred to as unbounded DP) and replacement (sometimes referred to as substitution or unbounded DP). Under the add/remove definition, one of x or x' is obtained by removing one record from the other. As such, the sizes of the datasets differ by

1. In contrast, both x and x' have the same size under the replacement definition where the datasets agree on all but one record. That is, under the replacement definition the existence of a record is not considered private information. The private information instead lies in the value of the record. Additional variants of the definition of neighboring datasets exist that are tailored to specific settings. For example, if a user can contribute multiple records to the dataset we often distinguish between event-level and user-level differential privacy (see e.g. [ZWC+22]).

In this thesis, we use both the add/remove and replacement variants of differential privacy. In most of Chapter 2 we use a generalized definition that falls under both categories, in Section 2.7 and Chapter 3 we use the add/remove definition, and in Chapters 4 and 5 we use the replacement definition. The exact definition is introduced in the preliminary section of each chapter as it depends on the input domain.

Whether the add/remove or the replacement variant is preferred depends on the given query. The difference has little relevance for many queries from a theoretical computer science perspective. A mechanism that satisfies differential privacy with one definition typically also does so with the other. However, for some queries the distinction is crucial. We discuss this after Fact 1.5. Next, we argue that the definition of neighboring datasets is a reasonable concept for privacy-preserving mechanisms.

Statistical inference is *not* a privacy violation It is a common misconception that the purpose of a differentially private query is to restrict how much a data analyst learns about any individual in the dataset. But that is not the promise of differential privacy. The definition does not protect against statistical inference! But that is by design.

Consider a dataset x' constructed by removing the data of one individual, say *Mr. S*, from dataset x. From a query $\mathcal{M}(x')$ on the dataset without the data of *Mr. S* we might hope to learn statistical properties of the underlying population. We might in turn be able to infer a lot about *Mr. S*. But the query $\mathcal{M}(x')$ had no access to *Mr. S*'s data and therefore we do not consider this a privacy violation. This is exactly the kind of information we hope to learn from a statistical query.

A canonical example of this phenomenon is the correlation between lifelong smoking and lung cancer. For many years the relationship between smoking and lung cancer was unclear but large-scale studies with clear, statistically-significant evidence [WG50] eventually convinced

the scientific community and later the general public that smoking and lung cancer are highly correlated. If we know that *Mr. S* is a lifelong smoker we can infer that he has an elevated cancer risk. Since we can learn this from $\mathcal{M}(x')$ we do not necessarily consider it a privacy violation if we learn this when his data is included in the query $\mathcal{M}(x)$. However, if the output of $\mathcal{M}(x)$ is a list of everyone in the dataset with cancer we would consider it a privacy violation. In that case, our assessment of *Mr. S*'s medical condition is no longer based on statistical inference and we learn something about him from $\mathcal{M}(x)$ that we could not learn from $\mathcal{M}(x')$.

The distinction is subtle but critical. It is easy to confuse the two situations as was the case in a paper that questioned the effectiveness of the privacy-preserving properties of the system deployed by the US Census Bureau [KKM+21]. This prompted a strongly worded response in the form of a blog post signed by several prominent researchers in the DP community [BDD+21]. The following quote is their conclusion for the example discussed above: *"The statistical inference of elevated cancer risk—made before Mr. S was born—did not violate Mr. S's privacy. To conclude otherwise is to define science to be a privacy attack."*

As such, we should not think of differential privacy as a framework to limit what we learn about individuals. Instead, we can think of differential privacy as a restriction on how much we learn about *Mr. S* from the query $\mathcal{M}(x)$ compared to $\mathcal{M}(x')$. The risk to *Mr. S* should be roughly the same whether or not his data is in the dataset.

1.2.2 Privacy Loss Random Variables

Next, we discuss how to quantify privacy loss in the framework of differential privacy. It is important to clarify that privacy loss in differential privacy is a function of the output and the data release mechanism and not of the output in and of itself. We have to understand the underlying mechanism to quantify privacy loss. This is in contrast to some privacy-preserving frameworks such as k-anonymity [Swe02]. As a motivation for this choice consider the following toy example: Alice's personal information is part of a dataset that was used for a public data release. She is examining the data and notices her social security number. Is this a privacy violation? Intuitively this seems like a clear privacy violation, but we need to understand the underlying mechanism. If the mechanism used for the data release looked up Alice's data to output her social

security number this is a clear privacy violation. In contrast, a mechanism that prints a string of random numbers clearly does not violate the privacy of anyone. If the random string happens to match Alice's social security number that is merely coincidental. Anyone familiar with the mechanism would disregard the number as being meaningless.

Privacy loss is a function of an output, the underlying mechanism, and a pair of neighboring datasets. Throughout this section we assume that the output domains of all mechanisms are discrete. This assumption simplifies the discussion of these concepts since we can define the probability of outputting a specific output, but analogous definitions exist for continuous output domains.

Definition 1.1 (Privay Loss)**.** Let $\mathcal{M}\colon \mathcal{U}^{\mathbb{N}} \to \mathcal{R}$ be any randomized mechanism where $\mathcal{R}$ is discrete. For a pair of neighboring datasets $x \sim x'$ the privacy loss of an output $y \in \mathcal{R}$ is defined as

$$f_{\mathcal{M}(x)||\mathcal{M}(x')}(y) = \begin{cases} \infty, & \text{if } \Pr[\mathcal{M}(x') = y] = 0 \\ -\infty, & \text{if } \Pr[\mathcal{M}(x) = y] = 0 \\ \ln\left(\frac{\Pr[\mathcal{M}(x)=y]}{\Pr[\mathcal{M}(x')=y]}\right), & \text{otherwise.} \end{cases}$$

Privacy loss is undefined for outputs with probability 0 for both datasets.

We can think of privacy loss as an indicator of which of x or x' is most likely to be the input given the observed output. This perspective is related to statistical hypothesis testing. Using the example from the previous section assume that we suspect that *Mr. S* has cancer and the input dataset is x. Our null hypothesis would be that his data was never collected and the input dataset is therefore x'. If the actual dataset was x the privacy loss of an output y indicates if we gain support for the correct hypothesis and therefore learn something about *Mr. S*. If the privacy loss is high it strongly supports the correct hypothesis. But if the privacy loss is negative the output supports the incorrect hypothesis. When the privacy loss is exactly 0 we do not gain support for either hypothesis, and if the privacy loss is infinite we can conclude that the input must be x. From this definition of privacy loss, it follows that all differentially private mechanisms should be randomized. Since for any pair of neighboring datasets and a deterministic mapping between inputs and outputs, we either have (1) the datasets are mapped to the same output and therefore we effectively disregard part of the dataset or (2) the datasets are mapped to different outputs implying that the privacy

loss is infinite. We can define a random variable to describe the privacy loss from running the mechanism.

Definition 1.2 (Privay Loss Random Variable (PRV))**.** The *privacy loss random variable* for a pair of neighboring datasets $x \sim x'$ and $\mathcal{M}(x)$ is given by $f_{\mathcal{M}(x)||\mathcal{M}(x')}(Y)$ for $Y \leftarrow \mathcal{M}(x)$.

The distribution of the PRV defined above describes the privacy loss from running $\mathcal{M}(x)$ in regards to the pair $x \sim x'$. Informally, a mechanism provides good privacy guarantees if the privacy loss random variable is small with high probability. We are now ready to define differential privacy. The goal of differential privacy is to give good privacy guarantees for any input. As such the definition of differential privacy is a restriction over all possible pairs of neighboring datasets and their privacy loss random variables. Note that since the neighborhood relation between datasets is symmetric each pair has two PRV.

Definition 1.3 ((ε, δ)-differential privacy [DR14])**.** A randomized mechanism $\mathcal{M} \colon \mathcal{U}^{\mathbb{N}} \to \mathcal{R}$ satisfies (ε, δ)-differential privacy if and only if for all subsets of outputs $Z \subseteq \mathcal{R}$ and all pairs of neighboring datasets $x \sim x'$ it holds that

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

We refer to the case of $\delta = 0$ as pure differential privacy and $\delta > 0$ as approximate differential privacy. Pure differential privacy is also denoted simply as ε-differential privacy. We discuss other variants of differential privacy later in this section. We generally think of δ as negligible in the size of the dataset (typically $\delta < |x|^{-\omega(1)}$).

Early work on differential privacy was primarily on pure differential privacy. This definition is arguably the most natural and simplest to understand. The definition states that the privacy loss of any ε-DP mechanism is never above ε. This strong guarantee makes pure differential privacy desirable and several of the most popular mechanisms satisfy the definition (e.g. [DMNS06, War65, MT07]).

However, the strict requirement of pure differential privacy comes at a high utility cost for some queries. The definition does not differentiate between a mechanism that has privacy loss ε with high probability from one with privacy loss ε with negligible probability. Since events with infinite privacy loss can never occur the output space must also be the same for all inputs. The relaxation to approximate differential privacy

allows us to design mechanisms with much better utility for certain queries.

One technique for designing an approximate differential privacy algorithm is to bound the probability that the privacy loss exceeds ε by δ for all inputs. We use this approach in Chapters 2 and 3. This condition is sufficient for (ε, δ)-DP but not necessary. Fact 1.1 gives the necessary condition for satisfying (ε, δ)-DP. In some cases using this exact condition leads to better utility through an improved analysis (see e.g. [BW18]).

Fact 1.1. Let $\mathcal{M}\colon \mathcal{U}^{\mathbb{N}} \to \mathcal{R}$ be a (ε, δ)-differentially private mechanism and let $E := \{y \in \mathcal{R} | f_{\mathcal{M}(x)||\mathcal{M}(x')}(y) > \varepsilon\}$ be the set of all outputs with privacy loss greater than ε for a pair of neighboring datasets $x \sim x'$. Then

$$\Pr[\mathcal{M}(x) \in E] - e^{\varepsilon} \Pr[\mathcal{M}(x') \in E] \leq \delta \ .$$

The strength of differential privacy comes from quantifying privacy loss. By providing formal quantitative privacy guarantees we can directly compare the privacy loss of mechanisms with completely different behavior. The quantitative privacy parameters also give us the flexibility to adjust our privacy requirement to a query. For example, we can require strong privacy guarantees for mechanisms working with highly confidential data such as medical history, but we might accept worse privacy guarantees for a query on less confidential data such as whether or not we clicked on a specific advertisement.

Sensitivity A common technique in differential privacy is to first compute some deterministic function over the dataset and then perturb the output by adding random noise scaled to the sensitivity of the function. In fact, we use the technique at least once in each chapter of this thesis. The sensitivity of a function is a restriction on the difference between outputs of the function with a pair of neighboring datasets as inputs. The intuition behind this technique is that if the output of a function does not change much between neighboring datasets we do not need to add much noise to achieve privacy. In contrast, it requires much more noise to privatize a function whose output can change drastically by adding the data of one individual.

Two commonly used variants of sensitivity are ℓ_1-sensitivity and ℓ_2-sensitivity. Both are special cases of ℓ_p-sensitivity as defined below. In some papers either of these two variants is simply referred to as *the sensitivity* when only one is used. In this thesis the 'sensitivity' of a

deterministic function refers to any restriction on the change to output that holds over all pairs of neighboring datasets. In Chapter 3 we present an analysis of the sensitivity of the Misra-Gries sketch that does not fall under ℓ_p-sensitivity.

Definition 1.4 (ℓ_p-sensitivity)**.** Let $f\colon \mathcal{U}^{\mathbb{N}} \to \mathbb{R}^d$ be a deterministic function. For any $p \geq 1$ the ℓ_p-sensitivity of f is defined as

$$\Delta_p := \max_{x \sim x'} \|f(x) - f(x')\|_p \ ,$$

where $\|f(x) - f(x')\|_p := \left(\sum_{i=1}^{d} |f(x) - f(x')|^p\right)^{1/p}$ is the ℓ_p-distance.

The Laplace and Gaussian mechanisms add independently sampled random noise to each coordinate of a query. The noise is scaled to the ℓ_1- and ℓ_2-sensitivity, respectively. We define the mechanisms below and reintroduce them in the preliminary sections in relevant chapters.

Definition 1.5 (The Laplace Mechanism [DMNS06])**.** The Laplace Mechanism outputs $f(x) + \eta$ for a vector $\eta \in \mathbb{R}^d$ where each entry η_i is sampled independently from Laplace (Δ_1/ε). The distribution Laplace (Δ_1/ε) has probability density $\frac{\varepsilon}{2\Delta_1} e^{-|y|\varepsilon/\Delta_1}$ at $y \in \mathbb{R}$. The Laplace Mechanism satisfies ε-differential privacy.

Definition 1.6 (The Gaussian Mechanism [DR14, BW18])**.** The Gaussian Mechanism outputs $f(x) + \eta$ for a vector $\eta \in \mathbb{R}^d$ where each entry η_i is sampled independently from $\mathcal{N}(0, \sigma^2)$. The Gaussian Mechanism satisfies (ε, δ)-differential privacy if and only if $\Phi\left(\frac{\Delta_2}{2\sigma} - \frac{\varepsilon\sigma}{\Delta_2}\right) - e^{\varepsilon}\Phi\left(-\frac{\Delta_2}{2\sigma} - \frac{\varepsilon\sigma}{\Delta_2}\right) \leq \delta$.

1.2.3 Properties of differential privacy

Next, we discuss some useful properties of differential privacy. In later chapters we again introduce some of the properties. There we only introduce properties if we use them in the proofs of our results. Here we also discuss why these properties are desirable for any privacy-preserving mechanisms. We discuss both the technical benefits when designing mechanisms and the benefits for data subjects.

The first property is immunity to post-processing. This property states that applying any function to the output of a differentially private mechanism does not break the privacy guarantee.

Fact 1.2. (Post-processing [DR14, Proposition 2.1]) Let $\mathcal{M}\colon \mathcal{U}^{\mathbb{N}} \to \mathcal{R}$ be any (ε, δ)-differentially private mechanism and let $g\colon \mathcal{R} \to \mathcal{R}'$ be any (randomized) function. Then the composed mechanism defined as $g \circ \mathcal{M}\colon \mathcal{U}^{\mathbb{N}} \to \mathcal{R}'$ satisfies (ε, δ)-differential privacy.

Immunity to post-processing is useful in the design of mechanisms since we can apply any data analysis if we only access the private dataset through differentially private mechanisms. The best example in this thesis is our mechanism introduced in Chapter 2. We design a data structure that compactly encodes the entries of a vector and prove that releasing a noisy version of the data structure satisfies differential privacy. We also design an algorithm for estimating the value of an entry from its noisy encoding. Since accessing the data structure is merely post-processing the privacy guarantees follow directly from Fact 1.2. Furthermore, if we were to change our access algorithm it would not negatively affect the privacy guarantees. We could also further analyze the noisy encodings. This can be useful if we for example decide to compute confidential intervals of an estimate after the data release.

The most important benefit of this property is protection against unforeseen adversaries. This is one of the strongest arguments for using differential privacy. In Section 1.1.1 we discussed how pseudonymization can sometimes be easily broken using side information. But more complicated mechanisms are also vulnerable to attacks. The US Census Bureau considered data swapping sufficient for more than 20 years before they broke their technique. Any technique designed to protect against specific attacks could be vulnerable to this kind of future attack. It is impossible to predict all kinds of techniques will exist in the future. But Fact 1.2 ensures that no attack can break the privacy guarantees of differential privacy.

Next we discuss the composition of differentially private mechanisms. On a high level, this property ensures that a mechanism that performs multiple differentially private accesses to a dataset still satisfies differential privacy. This allows us to construct complicated mechanisms by combining simpler mechanisms. We make use of this in both Chapters 2 and 5. However, the privacy guarantees deteriorate with each access. This is unfortunately inevitable for any privacy-preserving framework that allows us to accurately answer arbitrary statistical queries. We know this due to Dinur and Nissim [DN03] who showed that answering sufficiently many subset sums accurately allows us to reconstruct most of the underlying dataset.

Fact 1.3. (Basic Composition [DR14]) Let $\mathcal{M}_1 \colon \mathcal{U}^{\mathbb{N}} \to \mathcal{R}_1$ and $\mathcal{M}_2 \colon \mathcal{U}^{\mathbb{N}} \to \mathcal{R}_2$ denote randomized mechanisms satisfying $(\varepsilon_1, \delta_1)$-DP and $(\varepsilon_2, \delta_2)$-DP, respectively. Then the mechanism defined as $\mathcal{M}(x) := (\mathcal{M}_1(x), \mathcal{M}_2(x))$ satisfies $(\varepsilon_1 + \varepsilon_2, \delta_1 + \delta_2)$-DP.

The basic composition theorem presented in Fact 1.3 gives us an easy way of upper bounding the privacy parameters of a composed mechanism. We simply add up privacy parameters for each part of the mechanism. However, the basic composition theorem is sometimes far from optimal if we run many queries. Here we could instead use the advanced composition theorem presented in a simplified setting in Fact 1.4. Understanding composition is important because it allows us to give better privacy parameters for a mechanism which in turn improves the privacy-utility trade-off. There is much more to discuss regarding the technical details of composition but it is not required for understanding this thesis. A great resource for more details is the book chapter by Steinke [Ste22]. He presents some key results for both composition and privacy amplification. Privacy amplification is another important technical tool but we do not discuss it since we do not use amplification for any of the mechanisms in this thesis.

Fact 1.4. (Advanced Composition (simplified) [KOV15, Ste22]) Let $\mathcal{M}(x) := (\mathcal{M}_1(x), \ldots, \mathcal{M}_k(x))$ where $\mathcal{M}_i$ satisfies (ε, δ)-DP for all $i \in [k]$. Then $\mathcal{M}$ satisfies (ε', δ')-DP for any $\delta' \geq k\delta$ where

$$\varepsilon' = k\varepsilon^2/2 + \sqrt{2k\varepsilon^2 \ln(1/\delta')} \ .$$

Composition is a crucial property of any privacy-preserving framework used to publish statistics. In the real world personal information is not contained in a single dataset. Several entities store part of your sensitive information and therefore it can be included in multiple data releases. Composition ensures that we can still give formal privacy guarantees for all of these data releases if each of the independent data releases is performed with differential privacy. This is in contrast to the example with k-anonymity discussed in Section 1.1.1 where two k-anonymized datasets were combined to break privacy [GKS08]. The combination of composition and immunity to post-processing ensures that we can release statistics under differential privacy knowing the privacy guarantees cannot be broken.

The final property we discuss is group privacy. Definition 1.3 guarantees that the output distribution of a mechanism does not change

significantly between neighboring datasets. Under the definition of neighboring datasets we discussed earlier in the section this protects the data of any individual in the dataset. However, sometimes we want to give guarantees to a group of people. That is for example the case if all members of a family are part of a dataset. Their data could be highly correlated so we want guarantees similar to Definition 1.3. In such a situation we can use group privacy.

Fact 1.5. (Group privacy) Let $\mathcal{M}: \mathcal{U}^{\mathbb{N}} \to \mathcal{R}$ be an (ε, δ)-DP mechanism for groups of size 1. Then $\mathcal{M}$ is $(k\varepsilon, e^{(k-1)\varepsilon} k\delta)$-DP for groups of size k.

We can prove this by constructing intermediate datasets. Consider the case of $k = 2$. That is, two datasets x and x' differ only in the data of two individuals. Then there must exist at least one dataset x'' such $x \sim x''$ and $x'' \sim x'$. It follows from Definition 1.3 that

$$\Pr[\mathcal{M}(x) \in Z] \leq e^{\varepsilon} \Pr[\mathcal{M}(x'') \in Z] + \delta \leq e^{2\varepsilon} \Pr[\mathcal{M}(x') \in Z] + (1 + e^{\varepsilon})\delta$$

Fact 1.5 follows from applying this approach inductively. We can use the same idea to show that (ε, δ)-DP under add/remove neighborhood implies $(2\varepsilon, (1 + e^{\varepsilon})\delta)$-DP under replacement neighborhood. We can replace an entry of a dataset by first removing it and then adding an entry with its new value.

Most differentially private mechanisms analyzed under replacement neighborhood also satisfy differential privacy under add/remove neighborhood. Often the privacy parameters are even slightly better than the ones under replacement. However, differential privacy under replacement does not imply differential privacy under add/remove neighborhood. Consider a toy example where the dataset is an ordered list of n bits. If we want to estimate the sum of all entries with an even index replacing an entry only changes the output by 1. But if we add or remove an entry the sum can change by up to $n/2$. This example is of course extreme by construction. But the takeaway is that we should carefully choose our definition.

As a final note about the properties discussed in this section, it is worth mentioning that applying any of them does not always give us a tight analysis. For example, post-processing can sometimes improve privacy, some mechanisms compose better than the advanced composition theorem, and analyzing datasets differing in k entries directly sometimes gives better guarantees than group privacy. But the strength of these properties is their versatility and black-box nature. We do not need to

understand the details of a specific mechanism to give these types of guarantees. The properties hold for any differentially private mechanism.

1.2.4 Alternative definitions

In recent years several alternatives to Definition 1.3 have been proposed. Most relevant to this thesis is zero-Concentrated Differential Privacy [BS16]. We use this variant of DP in Chapters 4 and 5.

Definition 1.7 (ρ-zCDP [BS16])**.** A randomized mechanism $\mathcal{M}\colon \mathcal{U}^{\mathbb{N}} \to \mathcal{R}$ satisfies ρ-zCDP if and only if for all $\alpha > 1$ and all pairs of neighboring datasets $x \sim x'$ it holds that

$$D_\alpha\left(\mathcal{M}(x)||\mathcal{M}(x')\right) \leq \rho\alpha \ ,$$

where $D_\alpha(P||Q) := \frac{1}{\alpha-1}\ln\left(\mathbb{E}_{x\sim P}\left[\left(P(x)/Q(x)\right)^{\alpha-1}\right]\right)$ is the α-Rényi divergence ([Rén61, Equation (3.3)]) between two distributions P and Q.

Various definitions have been introduced to improve the analysis of specific mechanisms. Each of these definitions has an advantage over Definition 1.3 in certain settings. Here we list some noteworthy definitions along with one such advantage for each of them: zero-concentrated differential privacy [DR16, BS16] behaves particularly well under composition. Rényi differential privacy [Mir17, MTZ19, WBK18] and the moments accountant [ACG+16] are useful for analyzing the Gaussian mechanism under subsampling. Gaussian differential privacy [DRS19] captures the privacy loss random variable of the Gaussian mechanism exactly. The notion of f-DP [DRS19] gives a detailed description of the privacy guarantees in the hypothesis testing formulation of privacy. Lastly, a line of recent work [KJH20, GLW21, ZDW22] approximates the privacy curve numerically which gives tighter privacy guarantees than the analytical bounds in some cases.

These alternative definitions allow us to better analyze the privacy loss distributions of some mechanisms. This improved analysis is important because it allows us to give better privacy guarantees which lead to an improved privacy-utility trade-off.

A common property of all the definitions discussed above is that they describe a property that holds for all pairs of neighboring datasets. This means that a mechanism must consider worst-case datasets even if such datasets are unlikely to occur. Some attempts have been made to

aggressively relax this property with the intend of improving utility by only considering a small subset of datasets. Unfortunately, some of these definitions are vulnerable to reconstruction attacks (see [PDD+22]).

1.2.5 Transparency

Another benefit of differential privacy is the potential for increased transparency. The details of a differentially private mechanism can be revealed without affecting the privacy guarantees. This is in contrast to some ad hoc approaches where obscuring the release mechanism is necessary. This was the case for the data swapping approach previously deployed for the US Decennial Census: *"Its parameters and the details of the swapping algorithm cannot be published without compromising the privacy guarantee" [AAC+22]*. Since adopting differential privacy the Census Bureau have shared details about their disclosure avoidance system. They even publicly released their source code [Uni21c].

The possibility for transparency of course does not guarantee that a data release will be transparent. Fortunately, all parties have incentives to advocate for transparency. The data publisher wants to gain the trust of the data subjects. By publishing the details of their technique they show that they have done their due diligence to ensure that privacy is protected. They can also seek feedback on concrete design decisions for the mechanism. This can lead to closer collaboration with data analysts which can benefit both parties.

The biggest advantage for the data analyst is that they can take noise from the mechanism into account in their analysis. If they for example discover a strong correlation in the published data they can estimate whether it is present in the underlying dataset or a consequence of the injected noise. They would have no way of distinguishing these two cases from each other if the details of the mechanism is kept secret. It is worth noting here that the US Census Bureau has been criticized for the fact that they are injecting noise in the first place since the data is used by policymakers for important decisions. They were even subject to a lawsuit [Bre21] which has since been dismissed. But it is impossible to give any privacy guarantees without perturbing the dataset. It is also important to remember that all data collection is noisy. As an example, the Census Bureau themselves estimates that they undercounted the population of Texas by more than 500.000 people [Uni22]. This noise from the data collection is much more significant than the noise injected by the TopDown algorithm and it is much more difficult to account for.

The data subjects benefit from transparency because they want the best protection of their data. When the details of a mechanism are kept secret only a few people can be involved in the design process. If the mechanism is public privacy experts everywhere can scrutinize the details. This increases the probability that any mistakes are found.

A long-term benefit of transparency is shared knowledge between institutions. Designing a mechanism such as the TopDown algorithm is a challenging task that requires significant resources. The US Census Bureau has the necessary size to undertake such a process but other institutions, such as census bureaus from smaller countries, might want to conduct similar surveys but lack the resources to design a privacy-preserving system from the ground up. They can instead learn from the experiences of the US Census Bureau and perhaps even use parts of their source code. As such they can deploy a better privacy-preserving mechanism. This benefits everyone because our personal information is known to several institutions of varying sizes. Many technologies greatly benefit from the ability to reuse and build upon the work of others. Privacy concerns affect everyone and as such privacy-preserving mechanisms should be one such technology.

1.3 Contribution and overview

In the previous section we discussed differential privacy and its use for quantifying privacy loss. The definition allows us to design privacy-preserving mechanisms with strong formal privacy guarantees. However, it is crucial that the mechanisms also provide sufficient utility. In this thesis we present novel mechanisms with improved utility over previous state-of-the-art techniques. The thesis is based on the following results

- Chapter 2: Martin Aumüller, Christian Janos Lebeda, and Rasmus Pagh. Representing Sparse Vectors with Differential Privacy, Low Error, Optimal Space, and Fast Access [ALP22]. CCS 2021 & JPC 2022.
- Chapter 3: Christian Janos Lebeda and Jakub Tětek. Better Differentially Private Approximate Histograms and Heavy Hitters using the Misra-Gries Sketch [LT23]. PODS 2023 (Distinguished Paper).

- Chapter 4: Christian Janos Lebeda and Rasmus Pagh. Differentially Private Vector Aggregation when Coordinates have Different Sensitivity. Poster presented at TPDP 2022.
- Chapter 5: Martin Aumüller, Christian Janos Lebeda, Boel Nelson, and Rasmus Pagh. PLAN: Variance-Aware Differentially Private Mean Estimation [ALNP23]. Unpublished.
- Not included. James Hsin-yu Chiang, Bernardo David, Mariana Gama, and Christian Janos Lebeda. Correlated-Output-Differential-Privacy and Applications to Dark Pools [yCDGL23]. AFT 2023.

Common for all projects included in the thesis is that the goal is to estimate some value in $\mathbb{R}^d$. The exact problem setup and utility measure are presented in the preliminary section of each chapter. In Chapters 2 and 3 we consider problems with sparse data where the challenge is to satisfy memory constraints without sacrificing utility. In Chapters 4 and 5 we present mechanisms for settings where the magnitude of entries is skewed. We tailor the noise to this skew to improve utility. During my PhD studies I co-authored a paper on applying differential privacy to market mechanisms [yCDGL23]. The paper is not included in the thesis as the topic is not on release of statistics. Next, we give a short overview of each chapter.

Chapter 2 - Representing Sparse Vectors with Differential Privacy, Low Error, Optimal Space, and Fast Access

We consider the problem of releasing a sparse vector under either pure or approximate differential privacy. A sparse vector is a d-dimensional vector of real values where most entries are zero. The Laplace mechanism (Definition 1.5) is often the preferred solution for releasing dense vectors. However, the Laplace mechanism adds noise to all entries of the vector. As such, the noisy vector is not sparse and we must store all entries explicitly. But doing so is infeasible when the dimensionality d is huge. Therefore we want to design a mechanism with low memory usage without sacrificing utility. The primary challenge was to design a compact representation that has low sensitivity and is also robust to noise. Previous work either requires much more memory than the non-private representation [DMNS06], results in large error for worst-case inputs [CPST12, KKMN09], or has slow access time to entries [BV19].

We introduce the first data structure that performs well in all three metrics. We also present other techniques along with variants of our data structure such as a version designed specifically for sparse histograms as input. A histogram is a special case of a vector where all entries are integers. During the Summer of 2021 I was an OpenDP Visiting Fellow. My contribution was an open-source implementation of a variant of our mechanism for the OpenDP Library.

Chapter 3 - Better Differentially Private Approximate Histograms and Heavy Hitters using the Misra-Gries Sketch

The data structure presented in Chapter 2 allows us to release a differentially private sparse histogram (or vector) with memory requirements within a constant factor of a non-private representation. However, sometimes we cannot store a histogram even when privacy is not a concern. In the non-private streaming setting the Misra-Gries sketch [MG82] allows us to approximate a histogram using k counters. Chan, Li, Shi, and Xu [CLSX12] presented a technique for releasing a Misra-Gries sketch under ε-differential privacy. However, their technique adds noise that scales linearly in the size of the sketch. We present an improved analysis of the sensitivity of the Misra-Gries sketch. We utilize the structure of the sensitivity and present mechanisms that adds noise with at most twice the magnitude of the non-streaming setting. That is a significant improvement over the previous result and our error guarantees are optimal within small constants for both pure and approximate differential privacy.

Chapter 4 - Differentially Private Vector Aggregation when Coordinates have Different Sensitivity

In this chapter we consider the problem of releasing d real-valued queries using either the Laplace or Gaussian mechanisms. We explore a setting where each query has its own sensitivity. We are mostly interested in cases where some queries have significantly higher sensitivity than others. The two most natural approaches in this setting are to either (1) add noise with the same magnitude to each query or (2) add noise to each query with magnitude linearly proportional to the sensitivity. However, it turns out that neither of those approaches achieves the best utility for our error metric. Our goal is to minimize the p'th moment of the noise

where $p > 0$ is a parameter. The first approach adds a lot of noise to queries with low sensitivity while the latter adds a lot of noise to queries with high sensitivity. We give closed-form expressions for the optimal way to split the noise between queries for any choice of p. In the case of mean squared error ($p = 2$) and the Gaussian mechanism, the noise is scaled proportional to the root of the sensitivity. As such, we add less noise to low-sensitivity queries than approach (1) and less noise to high-sensitivity queries than approach (2). This reduces the mean squared error by a factor between 1 and d depending on the skew of the sensitivities.

This work was presented as a poster at the TPDP 2022 workshop but never resulted in a published paper. Instead, we used the technique in the design of our mechanism for mean estimation described in Chapter 5. The project is included as a stand-alone chapter because the idea is cleaner in this setting and the author list differs between the two projects.

Chapter 5 - PLAN: Variance-Aware Differentially Private Mean Estimation

We present a novel mechanism for estimating the mean of a d-dimensional distribution. Our mechanism consists of the following (simplified) steps: (1) we privately compute a rough estimate of the mean (2) the data is translated such that the rough estimate from the previous step is the new origin and each coordinate is scaled based on an estimate of the variance (3) each data sample is clipped to a maximum distance from the new origin (4) we add Gaussian noise to the mean and reverse the translation and scaling from step (2) as post-processing.

The rescaling in step (2) follows the approach from Chapter 4 with the same purpose. By spending more privacy budget on coordinates with high variance we reduce the error but we still add less noise to coordinates with low variance. Each step of our mechanism is complemented with theoretical error guarantees under a concentration assumption of the distribution. We also present and discuss experiments with both synthetic and real-world datasets.

Chapter 2

Representing Sparse Vectors with Differential Privacy, Low Error, Optimal Space, and Fast Access

Originally published in: ACM Conference on Computer and Communications Security (CCS 2021) & Journal of Privacy and Confidentiality (JPC 2022)

Joint work with: Martin Aumüller and Rasmus Pagh

Representing a sparse histogram, or more generally a sparse vector, is a fundamental task in differential privacy. An ideal solution would use space close to information-theoretical lower bounds, have an error distribution that depends optimally on the desired privacy level, and allow fast random access to entries in the vector. However, existing approaches have only achieved two of these three goals.

In this chapter we introduce the Approximate Laplace Projection (ALP) mechanism for approximating k-sparse vectors. This mechanism is shown to simultaneously have information-theoretically optimal space (up to constant factors), fast access to vector entries, and error of the same magnitude as the Laplace-mechanism applied to dense vectors. A key new technique is a *unary* representation of small integers, which we show to be robust against "randomized response" noise. This representation is combined with hashing, in the spirit of Bloom filters, to obtain a space-efficient, differentially private representation.

Our theoretical performance bounds are complemented by simulations which show that the constant factors on the main performance parameters are quite small, suggesting practicality of the technique.

2.1 Introduction

One of the fundamental results in differential privacy is that a histogram can be made differentially private by adding noise from the Laplace distribution to each entry of the histogram before it is released [DMNS06]. The expected magnitude of the noise on each histogram entry is $O(1/\varepsilon)$, where ε is the privacy parameter, and this is known to be optimal [HT10]. In fact, there is a sense in which the Laplace mechanism is optimal [KHP15]. However, some histograms of interest are extremely sparse, and cannot be represented in explicit form. Consider, for example, a histogram of the number of HTTP requests to various servers. Already the IPv4 address space has over 4 billion addresses, and the number of unique, valid URLs have long exceeded 10^{12}, so it is clearly not feasible to create a histogram with a (noisy) counter for each possible value.

Korolova, Kenthapadi, Mishra, and Ntoulas [KKMN09] showed that it is possible to achieve *approximate* differential privacy with space that depends only on the number of non-zero entries in the histogram. However, for (ε, δ)-differential privacy the upper bound on the expected per-entry error becomes $O\left(\frac{\log(1/\delta)}{\varepsilon}\right)$, which is significantly worse than the Laplace mechanism for small δ. Cormode, Procopiuc, Srivastava, and Tran [CPST12] showed how to achieve pure ε-differential privacy with expected per-entry error bounded by $O\left(\frac{\log(d)}{\varepsilon}\right)$, where d is the dimension of the histogram, i.e., the number of entries including zero entries. While both these methods sacrifice accuracy they are very fast, allowing access to entries of the private histogram in constant time. If access time is not of concern, it is possible to combine small space with small per-entry error, as shown by Balcer and Vadhan [BV19]. They achieve an error distribution that is comparable to the Laplace mechanism (up to constant factors) and space proportional to the sum n of all histogram entries — but the time to access a single entry is $\tilde{O}(n/\varepsilon)$, which is excessive for large datasets.

2.1.1 Our results

Our contribution is a mechanism that achieves optimal error and space (up to constant factors) with only a small increase in access time. The mechanism works for either approximate or pure differential privacy,

	Section 2.5	Section 2.6	Section 2.7	Section 2.8
Evaluation time	$O(\log d)$	$O(\log(\log(d)/\varepsilon))$	$O(\log d)$	$O(1)$
Error guarantee	Additive	Multiplicative	Additive	Additive
Input domain	Reals	Reals	Integers	Reals
Strong tail bounds	Yes	No	Yes	No

Table 2.1: Informal overview of our main results for pure differential privacy. Each mechanism also has an approximate differentially private version where $1/\delta$ replaces d in evaluation time.

with the former providing faster access time. Our main results are summarized in Theorem 2.1.

Theorem 2.1 (Informal Version of Theorems 2.3 and 2.4)**.** *Let x be a histogram with d entries each bounded by some value u where at most k entries have non-zero values. Given privacy parameters $\varepsilon > 0$ and $\delta \geq 0$, there exists an (ε, δ)-differentially private algorithm to represent x with per-entry error matching the Laplace mechanism up to constant factors and the following properties:*

- *If $\delta = 0$, the representation uses $O(k \log(d + u))$ bits and the access time is $O(\log d)$.*
- *If $\delta > 0$, it uses $O(k \log(d + u) + k \log(1/\delta))$ bits and the access time is $O(\log(1/\delta))$.*

For simplicity, the memory requirement does not include the space needed for storing hash functions. Asymptotically, this additional cost shows up for $k = o(\log d)$ as an additional $O(\log^2 d)$ or $O(\log(1/\delta) \log d)$ term. See the theorem statements in Section 2.5 for more details.

We present variations on this central result in Sections 2.6–2.8. These results provide a tighter error analysis for large ε values, and a way to improve the access time. These improvements in running time will sacrifice the property that the per-entry error matches the error of the Laplace mechanism. Table 2.1 shows an informal overview of notable properties for each variation.

2.1.2 Techniques

On a high level, we treat "small" and "large" values of the histogram differently. Large values are handled by the thresholding technique developed in [KKMN09, CPST12]. For small entries, we represent them using a *unary encoding* as fixed-length bit strings. From [KKMN09, CPST12] we know that their length is logarithmic in either d (for ε-DP) or $1/\delta$ (for (ε, δ)-DP). Privacy is achieved by perturbing each bit using randomized response [War65]. The value of an entry is estimated by finding the unary encoding that is closest in Hamming distance to the noisy bit string. As it turns out, the unary encoding is redundant enough to allow accurate estimation even when the probability of flipping each bit is a constant bounded away from $1/2$. In order to pack all unary representations into small space, we use hashing to randomize the position of each bit in the unary representation of a given entry. The access time is linear in the length of the bit representation, given constant time evaluation of the hash function. Interestingly, although hash collisions can lead to overestimates, they do not influence the error asymptotically.

We remark here that a direct application of randomized response does not give the desired $O(1/\varepsilon)$ error dependency, but we solve this issue with an initial scaling step that gives ε-differential privacy when combined with randomized response. Though the discussion above has been phrased in terms of histograms, which makes the comparison to earlier work easier, our techniques apply more generally to representing sparse real vectors, with privacy for neighboring datasets with bounded ℓ_1-distance. We also discuss a variant of our mechanism for the special case of histograms. Here it is possible to get $O\left(1/e^{\varepsilon}\right)$ error dependency, which is preferred in the low privacy setting.

2.1.3 Overview

In Section 2.2 we define differential privacy for vectors, discuss the Laplace mechanism, and provide probabilistic tools necessary for the analysis. In Section 2.3 we discuss related work on differentially private sparse histograms. In Section 2.4 we introduce the Approximate Laplace Projection (ALP) mechanism and analyze its theoretical guarantees. In Section 2.5 we improve space and access time using techniques from earlier work [KKMN09, CPST12]. Section 2.6 discusses using the ALP mechanism on bit length-encoded coordinates and shows that this improves the running time while incurring a multiplicative error. Section 2.7

discusses a tighter utility analysis for the special case of histograms. Section 2.8 discusses a data structure that achieves constant access time with expected error $O(1/\varepsilon)$, but with weak tail bounds. In Section 2.9 we evaluate the performance of the ALP mechanism based on simulations. We conclude the chapter with suggestions for practical applications in Section 2.10.

2.2 Preliminaries

Problem Setup.

In this work, we consider d-dimensional k-sparse vectors of non-negative real values. We say that a vector $x \in \mathbb{R}_+^d$ is k-sparse if it contains at most k non-zero entries. All entries are bounded from above by a value $u \in \mathbb{R}$, i.e., $\max_{i\in[d]} x_i =: \|x\|_\infty \leq u$. Here $[d]$ is the set of integers $\{1, \ldots, d\}$. We consider the problem of constructing an algorithm $\mathcal{M}$ for releasing a differentially private representation of x, i.e., $\tilde{x} := \mathcal{M}(x)$. Note that $\tilde{x}$ does not itself need to be k-sparse. In fact, Balcer and Vadhan [BV19] provided a lower bound for the error of any differentially private mechanism that always outputs a sparse vector. We discuss this further in Section 2.3.

Utility Measures.

We use two measures for the utility of an algorithm $\mathcal{M}$. We define the *per-entry error* as $|x_i - \tilde{x}_i|$ for any $i \in [d]$. We define the *maximum error* as $\max_{i\in[d]} |x_i - \tilde{x}_i| = \|x - \tilde{x}\|_\infty$. We compare the utility of algorithms using the expected per-entry and maximum error and compare the tail probabilities of the per-entry error of our algorithm with the Laplace mechanism introduced below.

Differential Privacy.

Differential privacy is a constraint to limit privacy loss, introduced by Dwork, McSherry, Nissim, and Smith [DMNS06]. We use definitions and results as presented by Dwork and Roth [DR14]. Intuitively, a differentially private algorithm ensures that a slight change in the input does not significantly impact the probability of seeing any particular output. We measure the distance between inputs using their ℓ_1-distance. In this work,

two vectors are neighbors iff their ℓ_1-distance is at most 1. That is for all neighboring vectors $x, x' \in \mathbb{R}_+^d$ we have $\|x - x'\|_1 := \sum_{i \in [d]} |x_i - x'_i| \leq 1$. We can now define differential privacy for neighboring vectors.

Definition 2.1 (Differential privacy [DR14, Def 2.4])**.** Given $\varepsilon > 0$ and $\delta \geq 0$, a randomized algorithm $\mathcal{M} \colon \mathbb{R}_+^d \to \mathcal{R}$ is (ε, δ)-differentially private if for all subsets of outputs $S \subseteq \mathcal{R}$ and pairs of k-sparse input vectors $x, x' \in \mathbb{R}_+^d$ such that $\|x - x'\|_1 \leq 1$ it holds that:

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

$\mathcal{M}$ satisfies *approximate differential privacy* when $\delta > 0$ and *pure differential privacy* when $\delta = 0$. In particular, a pure differentially private algorithm satisfies *ε-differential privacy*. The following properties of differential privacy are useful in this chapter.

Lemma 2.1 (Post-processing [DR14, Proposition 2.1])**.** *Let $\mathcal{M} \colon \mathbb{R}_+^d \to \mathcal{R}$ be an (ε, δ)-differentially private algorithm and let $f \colon \mathcal{R} \to \mathcal{R}'$ be any randomized mapping. Then $f \circ \mathcal{M} \colon \mathbb{R}_+^d \to \mathcal{R}'$ is (ε, δ)-differentially private.*

Lemma 2.2 (Composition [DR14, Theorem 3.16])**.** *Let $\mathcal{M}_1 \colon \mathbb{R}_+^d \to \mathcal{R}_1$ and $\mathcal{M}_2 \colon \mathbb{R}_+^d \to \mathcal{R}_2$ be randomized algorithms such that M_1 is $(\varepsilon_1, \delta_1)$-differentially private and $\mathcal{M}_2$ is $(\varepsilon_2, \delta_2)$-differentially private. Then the algorithm $\mathcal{M}$ where $\mathcal{M}(x) = (\mathcal{M}_1(x), \mathcal{M}_2(x))$ is $(\varepsilon_1 + \varepsilon_2, \delta_1 + \delta_2)$-differentially private.*

Throughout this chapter, we clamp the output of all algorithms to the interval $[0, u]$. An estimate outside this interval is due to noise and clamping outputs cannot increase the error. It follows from Lemma 2.1 that clamping the output does not affect privacy. We clamp the output implicitly to simplify presentation.

Probabilistic Tools.

The Laplace Mechanism introduced by Dwork, McSherry, Nissim, and Smith [DMNS06] satisfies pure differential privacy by adding noise calibrated to the ℓ_1-distance to each entry. For completeness, Algorithm 1 provides a formulation of the Laplace mechanism in the context of releasing an ε-differentially private representation of a sparse vector.

Here Laplace$(1/\varepsilon)$ is the Laplace distribution with scale parameter $1/\varepsilon$. The PDF and CDF of the distribution are presented in Definition 2.2

Algorithm 1: The Laplace Mechanism

Parameters: $\varepsilon > 0$.
Input : k-sparse vector $x \in \mathbb{R}_+^d$.
Output : ε-differentially private approximation of x.

(1) Let $\tilde{x}_i = x_i + \eta_i$ for all $i \in [d]$, where $\eta_i \sim \text{Laplace}(1/\varepsilon)$ is sampled independently.
(2) Release $\tilde{x}$.

and the expected error and tail bound of the mechanism are shown in Proposition 2.1. The Laplace mechanism works well for vectors with low dimensionality and serves as a baseline for our work. However, it is impractical or even infeasible in the setting of k-sparse vectors. The output vector is dense, and as such the space requirement scales linearly in the input dimensionality d.

Definition 2.2. The probability density and cumulative distribution functions of the Laplace distribution centered around 0 with scale parameter $1/\varepsilon$ are

$$f(\tau) = \frac{\varepsilon}{2} e^{-|\tau|\varepsilon} \ .$$

$$\Pr[\text{Laplace}(1/\varepsilon) \leq \tau] = \begin{cases} \frac{1}{2} e^{\tau\varepsilon}, & \text{if } \tau < 0 \\ 1 - \frac{1}{2} e^{-\tau\varepsilon}, & \text{if } \tau \geq 0 \end{cases}$$

Proposition 2.1 (Expected Error and Tail Bound [DR14, Theorem 3.8])**.** The expected per-entry error and the maximum error of the Laplace mechanism are $\mathbb{E}[|x_i - \tilde{x}_i|] = O(1/\varepsilon)$ and $\mathbb{E}[\|x - \tilde{x}\|_\infty] = O\left(\frac{\log(d)}{\varepsilon}\right)$, respectively. With probability at least $1 - \beta$ we have:

$$|\text{Laplace}(1/\varepsilon)| \leq \frac{1}{\varepsilon} \ln \frac{1}{\beta} \ .$$

Random rounding (also known as stochastic rounding) is used for rounding a real value probabilistically based on its fractional part. We define random rounding for any real $r \in \mathbb{R}$ as follows:

$$\text{RandRound}\,(r) = \begin{cases} \lceil r \rceil & \text{with probability } r - \lfloor r \rfloor \\ \lfloor r \rfloor & \text{with probability } 1 - (r - \lfloor r \rfloor) \end{cases}$$

Lemma 2.3. *The expected error of random rounding is maximized when* $r - \lfloor r \rfloor = 0.5$*. For any* r *we have:*

$$\mathbb{E}[|r - \text{RandRound}\,(r)\,|] \leq \frac{1}{2} \ .$$

Randomized response was first introduced by Warner [War65]. The purpose of the mechanism is to achieve plausible deniability by changing one's answer to some question with probability p and answer truthfully with probability $1 - p$. We define randomized response for a boolean value $b \in \{0, 1\}$ as follows:

$$\text{RandResponse}(b, p) = \begin{cases} 1 - b & \text{with probability } p \\ b & \text{with probability } 1 - p \end{cases}$$

Universal Hashing.

A *hash family* is a collection of functions $\mathcal{H}$ mapping keys from a universe $\mathcal{U}$ to a range R. A family $\mathcal{H}$ is called *universal*, if each pair of different keys collides with probability at most $1/|R|$, where the randomness is taken over the random choice of $h \in \mathcal{H}$. A particularly efficient construction that uses $O(\log |\mathcal{U}|)$ bits and constant evaluation time is presented in [DHKP97].

Model of Computation.

We use the w-bit word RAM model defined by Hagerup [Hag98] where $w = \Theta(\log(d) + \log(u))$. This model allows constant time memory access and basic operations on w-bit words. As such, we can store a k-sparse vector using $O(k \log(d + u))$ bits with constant lookup time using a hash table. We assume that the privacy parameters ε and δ can be represented in a single word.

Negative Values.

In this chapter, we consider vectors with non-negative real values, but the mechanism can be generalized for negative values using the following reduction. Let $v \in \mathbb{R}^d$ be a real-valued k-sparse vector. Construct $x, y \in \mathbb{R}^d_+$ from v such that $x_i = \max(v_i, 0)$ and $y_i = -\min(v_i, 0)$. By construction both x and y are k-sparse and the ℓ_1-distance between vectors is preserved. We can access elements in v as $v_i = x_i - y_i$. As such, any differentially private representation of x and y can be used as a differentially private representation of v with at most twice the error. We can avoid the increased error by instead increasing the access time. We discuss this variant of our mechanism in Section 2.10.

Algorithm	Space (bits)	Access time	Per-entry error	Maximum error
Dwork et al. [DMNS06]	$O(d\log(u))$	$O(1)$	$O\left(\frac{1}{\varepsilon}\right)$	$O\left(\frac{\log(d)}{\varepsilon}\right)$
Cormode et al. [CPST12]	$O(k\log(d+u))$	$O(1)$	$O\left(\frac{\log(d)}{\varepsilon}\right)$	$O\left(\frac{\log(d)}{\varepsilon}\right)$
Balcer & Vadhan [BV19]	$\tilde{O}\left(\frac{n}{\varepsilon}\log(d)\right)$	$\tilde{O}\left(\frac{n}{\varepsilon}\right)$	$O\left(\frac{1}{\varepsilon}\right)$	$O\left(\frac{\log(d)}{\varepsilon}\right)$
Theorem 2.3 (this work)	$O(k\log(d+u))$	$O(\log(d))$	$O\left(\frac{1}{\varepsilon}\right)$	$O\left(\frac{\log(d)}{\varepsilon}\right)$
Korolova et al. [KKMN09]	$O(k\log(d+u))$	$O(1)$	$O\left(\frac{\log(1/\delta)}{\varepsilon}\right)$	$O\left(\frac{\log(1/\delta)}{\varepsilon}\right)$
Theorem 2.4 (this work)	$O(k(\log(d+u)+\log(1/\delta)))$	$O(\log(1/\delta))$	$O\left(\frac{1}{\varepsilon}\right)$	$O\left(\frac{\log(1/\delta)}{\varepsilon}\right)$

Table 2.2: Comparison with previous work. The performance is stated for worst-case input, and all bounds hold in expectation. Space for storing hash functions is not considered. The first four rows are results on ε-differential privacy, and the last two are on (ε,δ)-differential privacy. The $\tilde{O}$-notation suppresses logarithmic factors.

2.3 Related work

Previous work on releasing differentially private sparse vectors primarily focused on the special case of discrete vectors in the context of releasing the histogram of a dataset.

Korolova, Kenthapadi, Mishra, and Ntoulas [KKMN09] first introduced an approximately differentially private mechanism for the release of a sparse histogram. A similar mechanism was later introduced independently by Bun, Nissim, and Stemmer [BNS19b] in another context. The mechanism adds noise to non-zero entries and removes those with a noisy value below a threshold $t = O\left(\frac{\log(1/\delta)}{\varepsilon}\right)$. The threshold is chosen such that the probability of releasing an entry with true value 1 is at most δ. The expected maximum error is $O\left(\frac{\log(\max(k,1/\delta))}{\varepsilon}\right)$. Since δ is usually chosen to be negligible in the input size, we assume that $\delta \leq 1/k$. As such, the expected maximum error is $O\left(\frac{\log(1/\delta)}{\varepsilon}\right)$. We discuss the per-entry error below. Their mechanism is designed to satisfy differential privacy for discrete data. We extend their technique to real-valued data as part of Section 2.5, where we combine it with our mechanism.

Cormode, Procopiuc, Srivastava, and Tran [CPST12] introduced a differentially private mechanism in their work on range queries for sparse data. The mechanism adds noise to all entries and removes those with a noisy value below a threshold $t = O\left(\frac{\log(d)}{\varepsilon}\right)$. Here the threshold is used to reduce the expected output size. The number of noisy entries above t is $O(k)$ with high probability. The construction time of a naive implementation of their technique scales linearly in d. They improve on

this by sampling from a binomial distribution to determine the number of zero entries to store. They show that their approach produces the same output distribution as a naive implementation that adds noise to every entry. Their mechanism works for real-valued data in a straightforward way.

Since the expected number of non-zero entries in the output is $O(k)$ for both mechanisms above, their memory requirement is $O(k \log(d+u))$ bits using a hash table. An entry is accessed in constant time. The expected per-entry error depends on the true value of the entry. If the noisy value is above the threshold with sufficiently high probability, the expected error is $O(1/\varepsilon)$. However, this does not hold for entries that are likely removed. Consider for example an entry with a true value exactly at the threshold t. This entry is removed for any negative noise added. As such the expected per-entry error is $O(t)$ for worst-case input, which is $O\left(\frac{\log(1/\delta)}{\varepsilon}\right)$ and $O\left(\frac{\log(d)}{\varepsilon}\right)$ for the two mechanisms, respectively.

In their work on differential privacy on finite computers, Balcer and Vadhan [BV19] introduced several algorithms including some with similar utility as the mechanisms described above. Moreover, they provided a lower bound of $\Omega\left(\frac{\min\{\log(d), \log(\varepsilon/\delta), n\}}{\varepsilon}\right)$ for the expected per-entry error of any algorithm that always outputs a sparse histogram. (See [BV19, Theorem 7.2] for the precise technical statement.) Here n is the number of rows in the dataset, i.e., the sum of all entries of the histogram. This lower bound means that an algorithm that always outputs a $O(k)$-sparse histogram cannot achieve $O(1/\varepsilon)$ expected per-entry error for all input. They bypass this bound by producing a compact representation of a dense histogram. Their representation has expected per-entry and maximum error of $O(1/\varepsilon)$ and $O\left(\frac{\log(d)}{\varepsilon}\right)$, respectively. It requires $\tilde{O}\left(\frac{n}{\varepsilon}\log(d)\right)$ bits and an entry is accessed in time $\tilde{O}\left(\frac{n}{\varepsilon}\right)$. Note that their problem setup differs from ours in that each entry is bounded only by n such that $\|x\|_\infty \leq n$. That is, n serves a similar purpose as u does in our setup. We do not know how to extend their approach to our setup with real-valued input.

In light of the results achieved in previous work, our motivation is to design a mechanism that achieves three properties simultaneously: $O(1/\varepsilon)$ expected per-entry error for arbitrary input, fast access, and (asymptotically) optimal space. Previous approaches only achieved at most two of these properties simultaneously. Moreover, we want the per-entry error to match the tail bounds of the Laplace mechanism

up to constant factors. We construct a compact representation of a dense vector to bypass the lower bound for sparse vectors by Balcer and Vadhan [BV19]. The access time of our mechanism is $O(\log(d))$ and $O(\log(1/\delta))$ for pure and approximate differential privacy, respectively. Table 2.2 summarizes the results of previous work and our approach.

2.4 The ALP mechanism

In this section, we introduce the Approximate Laplace Projection (ALP) mechanism[1] and give an upper bound on the expected per-entry error. The ALP mechanism consists of two algorithms. The first algorithm constructs a differentially private representation of a k-sparse vector and the second estimates the value of an entry based on its representation.

2.4.1 A 1-differentially private algorithm

We start by considering the special case of $\varepsilon = 1$ and later generalize to all values of $\varepsilon > 0$. Moreover, the mechanism works well only for entries bounded by a parameter ψ. In general, this would mean that we had to set $\psi = u$ if we only were to use the ALP mechanism. However, in Section 2.5 we will discuss how to set ψ smaller and still perform well for all entries.

In the first step of the projection algorithm, we scale every non-zero entry by a parameter of the algorithm and use random rounding to map each such entry to an integer. We then store the unary representation of these integers in a two-dimensional bit-array using a sequence of universal hash functions [CW79]. We call this bit-array the *embedding*. Lastly, we apply randomized response on the embedding to achieve privacy. The pseudocode of the algorithm is given in Algorithm 2 and we discuss it next.

Figure 2.1 shows an example of an embedding before applying randomized response. The input is a vector x where the ith entry x_i is the only non-zero value. The result of evaluating i for each hash function is shown in the table at the bottom and the $m = 8$ bits representing the ith entry in the bit-array are highlighted. In Step (1) of the algorithm, x_i is

[1]The name is chosen to indicate that the error distribution is approximately like the Laplace distribution, and that we *project* the sparse vector to a much lower-dimensional representation. It also celebrates the mountains, whose silhouette plays a role in a certain random walk considered in the analysis of the ALP mechanism.

Algorithm 2: ALP1-Projection

Parameters: $\alpha, \psi > 0$, and $s \in \mathbb{N}$.
Input : k-sparse vector $x \in \mathbb{R}_+^d$ where $s > 2k$. Sequence of universal hash functions from domain $[d]$ to $[s]$, $h = (h_1, \ldots, h_m)$, where $m = \left\lceil \frac{\psi}{\alpha} \right\rceil$.
Output : 1-differentially private representation of x.

(1) Apply random rounding to a scaled version of each non-zero entry of x such that $y_i = \text{RandRound}\left(\frac{x_i}{\alpha}\right)$.

(2) Construct $z \in \{0,1\}^{s \times m}$ by hashing the unary representations of y such that:

$$z_{a,b} = \begin{cases} 1, & \exists i : \ b \leq y_i \text{ and } h_b(i) = a \\ 0, & \text{otherwise} \end{cases}$$

(3) Apply randomized response to each bit of z such that $\tilde{z}_{a,b} = \text{RandResponse}\left(z_{a,b}, \frac{1}{\alpha+2}\right)$.

(4) Release h and $\tilde{z}$.

scaled by $1/\alpha$ and randomized rounding is applied to the scaled value. This results in $y_i = 5$. Using the hash functions, we represent this value in unary encoding by setting the first five bits to 1 in Step (2), where the jth bit is selected by evaluating the hash function h_j on i. The final three bits are unaffected by the entry. Finally, we apply randomized response in each cell of the bit-array. The bit-array after applying randomized response is not shown here, but we present it later in Figure 2.2. Both the bit-array and the hash functions are the differentially private representation of the input vector x. We use this construction when estimating the value of x_i later.

The algorithm takes three parameters α, ψ, and s. The parameters α and s are adjustable and affect constant factors for space usage, error, and access time. By increasing s, we reduce the probability of hash collisions while increasing the size of the representation. The parameter α is used to balance two sources of error: by lowering α, we reduce the error in the encoding in Step (1) but increase the noise in Step (3). This parameter also affects the size because it is used to set m. We further discuss these parameters later as part of the error analysis. In Section 2.9 we discuss how to select values for α and s. Throughout the chapter we sometimes

	$z_{\_,1}$	$z_{\_,2}$	$z_{\_,3}$	$z_{\_,4}$	$z_{\_,5}$	$z_{\_,6}$	$z_{\_,7}$	$z_{\_,8}$
$z_{1,\_}$	0	0	0	0	0	0	0	0
$z_{2,\_}$	1	0	0	0	0	0	0	0
$z_{3,\_}$	0	1	0	1	0	0	0	0
$z_{4,\_}$	0	0	0	0	1	0	0	0
$z_{5,\_}$	0	0	1	0	0	0	0	0

$h_1(i)$	$h_2(i)$	$h_3(i)$	$h_4(i)$	$h_5(i)$	$h_6(i)$	$h_7(i)$	$h_8(i)$
2	3	5	3	4	4	1	2

Figure 2.1: Embedding with $\psi/\alpha = m = 8$, $s = 5$ and $y_i = 5$. The ith entry is the only non-zero entry.

assume that α is a constant and s is a constant multiple of k, that is $\alpha = \Theta(1)$ and $s = \Theta(k)$. The parameter ψ bounds the values stored in the embedding. We discuss ψ as part of the error analysis as well.

Lemma 2.4. *Algorithm 2 satisfies 1-differential privacy.*

Proof. Let $x, x' \in \mathbb{R}_+^d$ denote two neighboring vectors. We prove the lemma in several steps. First, the vectors differ only in their ith entry. In this case, we start by assuming that only a single bit of z is affected by changing x to x' and that there are no hash collisions. We then allow z to differ in several bits and include hash collisions. Finally, we generalize to the case where x and x' differ in more than one entry.

Assume that z differs only in a single bit for x and x'. Let Y and Y' denote the events that the affected bit is set to one after running the algorithm with input x and x', respectively. Let $p = \frac{1}{\alpha+2}$ be the parameter of the randomized response step. Then we have $\Pr[Y] = (1-r)\cdot p + r\cdot(1-p)$, where $r = \frac{x_i}{\alpha} - \left\lfloor \frac{\min(x_i, x_i')}{\alpha} \right\rfloor$ denotes the probability of the bit being one before the randomized response step. Similarly for x' we define $r' = \frac{x_i'}{\alpha} - \left\lfloor \frac{\min(x_i, x_i')}{\alpha} \right\rfloor$. The minimum term is needed when $\max(x_i, x_i')$ is a multiple of α such that $\max(r, r') = 1$. We find the

difference in the probability of Y and Y' occurring as:

$$\begin{aligned}\Pr[Y] - \Pr[Y'] &= (p + r - 2rp) - (p + r' - 2r'p) \\ &= (r - r') \cdot (1 - 2p) \\ &= \frac{x_i - x'_i}{\alpha + 2} \ .\end{aligned}$$

By symmetry, the absolute difference in probability for setting the bit to either zero or one is $\frac{|x_i - x'_i|}{\alpha+2}$. Let Z be an arbitrary output of Algorithm 2. Since x and x' agree on all but the ith entry, the change in probability of outputting Z depends only on the affected bit. Now let Y and Y' denote the events that the bit agrees with output Z for input x and x'. Then we find the ratio of probabilities of outputting Z as:

$$\begin{aligned}\frac{\Pr[\text{ALP1-Projection}(x') = Z]}{\Pr[\text{ALP1-Projection}(x) = Z]} = \frac{\Pr[Y']}{\Pr[Y]} &\leq \frac{\Pr[Y] + \frac{|x_i - x'_i|}{\alpha+2}}{\Pr[Y]} \\ &\leq \frac{p + \frac{|x_i - x'_i|}{\alpha+2}}{p} = 1 + |x_i - x'_i| \\ &\leq e^{|x_i - x'_i|} \ .\end{aligned}$$

Here the second inequality follows from $p \leq \Pr[Y] \leq 1 - p$. Next, we take hash collisions into account as follows: Let p' denote the probability that the bit agrees with Z for input x after setting the ith entry to zero. That is, we have $p \leq p' \leq 1 - p$ and $\Pr[Y] = (1 - r) \cdot p' + r \cdot (1 - p)$. The absolute difference in probability is still bounded such that $\Pr[Y] - \Pr[Y'] \leq \frac{|x_i - x'_i|}{\alpha+2}$. As such it still holds that:

$$\frac{\Pr[\text{ALP1-Projection}(x') = Z]}{\Pr[\text{ALP1-Projection}(x) = Z]} \leq e^{|x_i - x'_i|} \ .$$

Finally, we remove the assumption that only a single bit of z is affected by composing probabilities. We provide the following inductive construction. Let $x, x' \in \mathbb{R}^d_+$ be vectors that differ in the ith entry such that exactly two bits of z are affected. We consider the case of $x_i < x'_i$ and fix a vector $x'' \in \mathbb{R}^d_+$ with $x_i < x''_i < x'_i$ such that the differences affects exactly one bit each. Again, let Z be an arbitrary output of Algorithm 2. Applying the upper bound from above twice, we may bound the change in probabilities by:

$$\begin{aligned}\frac{\Pr[\text{ALP1-Projection}(x') = Z]}{\Pr[\text{ALP1-Projection}(x) = Z]} &= \frac{\Pr[\text{ALP1-Projection}(x'') = Z]}{\Pr[\text{ALP1-Projection}(x) = Z]} \\ &\quad \cdot \frac{\Pr[\text{ALP1-Projection}(x') = Z]}{\Pr[\text{ALP1-Projection}(x'') = Z]} \\ &\leq e^{|x_i - x_i''|} \cdot e^{|x_i'' - x_i'|} \\ &= e^{|x_i - x_i'|} \ ,\end{aligned}$$

which can be applied inductively if changing an entry affects more than two bits of z.

We are now ready to generalize to any vectors $x, x' \in \mathbb{R}_+^d$, i.e., where vectors may differ in more than a single position. Using the bound from above, we can bound the ratio of probabilities by:

$$\begin{aligned}\frac{\Pr[\text{ALP1-Projection}(x') = Z]}{\Pr[\text{ALP1-Projection}(x) = Z]} &\leq \prod_{i \in [d]} e^{|x_i - x_i'|} \\ &= e^{\sum_{i \in [d]} |x_i - x_i'|} \\ &= e^{\|x - x'\|_1} \ .\end{aligned}$$

The privacy loss is thus bounded by the ℓ_1-distance of the vectors for any output. Recall that the ℓ_1-distance is upper bounded by 1 for two neighboring vectors. As such the algorithm is 1-differentially private as for any pair of neighboring vectors x and x' and any subset of outputs S we have:

$$\begin{aligned}\Pr[\text{ALP1-Projection}(x) \in S] &\leq e^{\|x - x'\|_1} \Pr[\text{ALP1-Projection}(x') \in S] \\ &\leq e \cdot \Pr[\text{ALP1-Projection}(x') \in S] \ .\end{aligned}$$

□

The following lemma summarizes the space complexity of storing the bit-array and the collection of hash functions.

Lemma 2.5. *The number of bits required to store h and $\tilde{z}$ is*

$$O\left(\frac{(s + \log d) \cdot \psi}{\alpha}\right) \ .$$

Proof. By definition $m = O\left(\frac{\psi}{\alpha}\right)$ and as such $s \cdot m = O\left(\frac{s\psi}{\alpha}\right)$ bits are used to store $\tilde{z}$. Each hash function uses $O(\log(d))$ bits for a total of $O\left(\frac{\log(d)\psi}{\alpha}\right)$ bits to store h. □

Algorithm 3: ALP1-Estimator

Parameters: $\alpha > 0$.
Input : Embedding $\tilde{z} \in \{0,1\}^{s \times m}$. Sequence of universal hash functions $h = (h_1, \ldots, h_m)$. Index $i \in [d]$.
Output : Estimate of x_i.

(1) Define the function $f\colon \{0, \ldots, m\} \to \mathbb{Z}$ as:

$$f(n) = \sum_{a=1}^{n} 2\tilde{z}_{h_a(i),a} - 1$$

(2) Let P be the set of arguments maximizing f. That is,

$$P = \{n \in \{0, \ldots, m\} : f(a) \leq f(n) \text{ for all } a \in \{0, \ldots, m\}\}$$

(3) Let $\tilde{y}_i = \text{average}(P)$
(4) Return $\tilde{y}_i \cdot \alpha$.

2.4.2 Estimating an entry

We now introduce the algorithm to estimate an entry based on the embedding from Algorithm 2. When accessing the ith entry, we estimate the value of y_i and multiply by α to reverse the initial scaling of x_i. The estimate of y_i is chosen to maximize a partial sum. If multiple values maximize the sum we use their average.

Intuition.

The first y_i bits representing the ith entry are set to one before applying noise in Algorithm 2, cf. Figure 2.1. The last $m - y_i$ bits are zero, except if there are hash collisions. Some bits might be flipped due to randomized response, but we expect the majority of the first y_i bits to be ones and the majority of the remaining $m - y_i$ bits to be zeros. As such the estimate of y_i is based on prefixes maximizing the difference between ones and zeros. The pseudocode for the algorithm is given as Algorithm 3.

Figure 2.2 shows an example of Algorithm 3. The example is based on the embedding from Figure 2.1 after adding noise. The plot shows the value of f for all candidate estimates. This sum is maximized at positions 3 and 5. This is visualized as the global *peaks* in the plot. The estimate is the average of those positions.

	$\tilde{z}_{\_,1}$	$\tilde{z}_{\_,2}$	$\tilde{z}_{\_,3}$	$\tilde{z}_{\_,4}$	$\tilde{z}_{\_,5}$	$\tilde{z}_{\_,6}$	$\tilde{z}_{\_,7}$	$\tilde{z}_{\_,8}$
$\tilde{z}_{1,\_}$	0	0	0	1	0	0	0	0
$\tilde{z}_{2,\_}$	1	1	0	0	0	1	0	1
$\tilde{z}_{3,\_}$	0	1	0	0	0	0	0	0
$\tilde{z}_{4,\_}$	0	1	0	0	1	0	0	1
$\tilde{z}_{5,\_}$	0	0	1	1	0	0	0	0

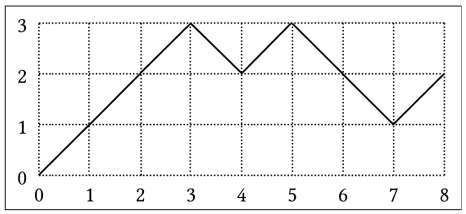


Figure 2.2: Estimation of ith entry from Figure 2.1.
The partial sum is maximized at indices 3 and 5.
The estimate is 4, while the true value was 5.

Lemma 2.6. *The evaluation time of Algorithm 3 is* $O\left(\frac{\psi}{\alpha}\right)$.

Proof. We can compute all partial sums by evaluating each bit $(\tilde{z}_{h_1(i),1}, \ldots, \tilde{z}_{h_m(i),m})$ once using dynamic programming. As such the evaluation time is $O(m)$ with $m = \left\lceil \frac{\psi}{\alpha} \right\rceil$. □

We now analyze the per-entry error of Algorithm 3. We first analyze the expected error based on the parameters of the algorithm. The results are presented in Lemma 2.11. In Lemmas 2.12 and 2.13 we bound the tail distribution of the per-entry error of the algorithm.

Lemma 2.7. *The expected per-entry error of Algorithm 3 is bounded by* $(\frac{1}{2} + \mathbb{E}[|y_i - \tilde{y}_i|]) \cdot \alpha$ *for entries with a value of at most* ψ.

Proof. It is clear that the error of the ith entry is α times the difference between $\tilde{y}_i$ and $\frac{x_i}{\alpha}$. The expected difference is bounded by:

$$\mathbb{E}\left[\left|\frac{x_i}{\alpha} - \tilde{y}_i\right|\right] \leq \mathbb{E}\left[\left|\frac{x_i}{\alpha} - y_i\right|\right] + \mathbb{E}[|y_i - \tilde{y}_i|] \leq \frac{1}{2} + \mathbb{E}[|y_i - \tilde{y}_i|] \ .$$

The last inequality follows from Lemma 2.3. □

We find an upper bound on $\mathbb{E}[|y_i - \tilde{y}_i|]$ by analyzing simple random walks. A simple random walk is a stochastic process such that $S_0 = 0$ and $S_n = \sum_{\ell=1}^{n} X_\ell$, where X are independent and identically distributed random variables with $\Pr[X_\ell = 1] = p$ and $\Pr[X_\ell = -1] = 1 - p$.

Lemma 2.8. *Let* S *be a simple random walk with* $p < 1/2$. *At any step* n *the probability that there exists a later step* $\ell > n$ *such that* $S_\ell > S_n$ *is* $\frac{p}{1-p}$.

Proof. Alm [Alm02, Theorem 1] showed that for any $p < 1/2$ there exists a step $\ell > n$ such that $S_\ell = S_n + k$ with probability $\left(\frac{p}{1-p}\right)^k$. The Lemma follows from the case of $k = 1$. □

For our analysis, we are concerned with the maximum n such that $S_n \geq 0$. For an infinite random walk where $p < 1/2$ such an n exists with probability 1.

Lemma 2.9. *Let S be a simple random walk with $p < 1/2$. The expected last non-negative step of S is:* $\mathbb{E}[\max_n : S_n \geq 0] = \frac{4(p-p^2)}{(1-2p)^2}$.

Proof. We use Lemma 2.8 to find the probability that S_n is the unique maximum in $\{S_n, \ldots, S_\infty\}$ as follows:

$$\begin{aligned}\Pr[S_n > \max(\{S_{n+1}, \ldots, S_\infty\})] \\ &= \Pr[X_{n+1} = -1] \cdot \Pr[S_{n+1} = \max(\{S_{n+1}, \ldots, S_\infty\})] \\ &= (1-p) \cdot \left(1 - \frac{p}{1-p}\right) = 1 - 2p \ .\end{aligned}$$

The last non-negative step must have value exactly zero and as such must be at an even numbered step. The probability that step $2i$ is the last non-negative is:

$$\begin{aligned}\Pr[(\max_n : S_n \geq 0) = 2i] &= \Pr[S_{2i} = 0] \cdot \Pr[S_{2i} > \max(\{S_{2i+1}, \ldots, S_\infty\})] \\ &= \binom{2i}{i} p^i (1-p)^i (1-2p) \\ &= \binom{2i}{i} (p-p^2)^i (1-2p) \ .\end{aligned}$$

We are now ready to find the expected last non-negative step of an infinite simple random walk as:

$$\begin{aligned}\mathbb{E}[\max_n : S_n \geq 0] &= \sum_{i=0}^{\infty} 2i \cdot \Pr[(\max_n : S_n \geq 0) = 2i] \\ &= \sum_{i=0}^{\infty} 2i \binom{2i}{i} (p-p^2)^i (1-2p) \\ &= 2(1-2p) \sum_{i=0}^{\infty} i \binom{2i}{i} (p-p^2)^i \\ &= \frac{4(p-p^2)}{(1-2p)^2} \ .\end{aligned}$$

The last equality follows from the identity $\sum_{i=0}^{\infty} i\binom{2i}{i}(p-p^2)^i = \frac{2(p-p^2)}{(1-2p)^3}$. See Appendix 2.A for a proof of this identity. □

We are now ready to bound $\mathbb{E}[|y_i - \tilde{y}_i|]$. We consider entries with value at most ψ, i.e., $y_i \leq m$.

Lemma 2.10. *Let $y_i \leq m$ and $\gamma = \frac{\alpha+2}{1+\frac{\alpha k}{s}} - 2$. Then the expected value of $|y_i - \tilde{y}_i|$ is bounded such that*

$$\mathbb{E}[|y_i - \tilde{y}_i|] \leq \frac{4\alpha+4}{\alpha^2} + \frac{4\gamma+4}{\gamma^2} \ .$$

Proof. Recall the definition of P from Algorithm 3. Let $\bar{y}_i \in P$ denote an element furthest from y_i that is $|y_i - a| \leq |y_i - \bar{y}_i|$ for all $a \in P$. It is clearly sufficient to consider $\bar{y}_i$ for the proof since $|y_i - \tilde{y}_i| \leq |y_i - \bar{y}_i|$. We first consider the case of $\bar{y}_i \leq y_i$. It follows from the definition of $\bar{y}_i$ as a maximum that $\sum_{j=\bar{y}_i+1}^{y_i} \tilde{z}_{h_j(i),j} \leq 0$. As such at least half the bits $(\tilde{z}_{h_{\bar{y}_i+1}(i),\bar{y}_i+1}, \ldots, \tilde{z}_{h_{y_i}(i),y_i})$ must be zero, that is they were flipped by randomized response in Step (3) of Algorithm 2. As such the length of the longest interval ending at bit $\tilde{z}_{h_{y_i}(i),y_i}$ where at least half the bits were flipped is an upper bound on the value of $y_i - \bar{y}_i$. The expected size of said interval is bounded by the expected last non-negative step of a simple random walk with $p = \frac{1}{\alpha+2}$. It follows from Lemma 2.9 that:

$$\mathbb{E}[y_i - \bar{y}_i \mid \ \bar{y}_i \leq y_i] \leq \frac{4(p-p^2)}{(1-2p)^2} = \frac{\frac{4\alpha+4}{(\alpha+2)^2}}{\frac{\alpha^2}{(\alpha+2)^2}} = \frac{4\alpha+4}{\alpha^2} \ .$$

We can use a similar argument when $y_i \geq \bar{y}_i$ to show that at least half the bits in $(\tilde{z}_{h_{y_i+1}(i),y_i+1}, \ldots, \tilde{z}_{h_{\bar{y}_i}(i),\bar{y}_i})$ must be 1 since $\bar{y}_i$ is a maximum. In this case we have to consider the possibility of hash collisions. Each hash function maps to $[s]$ and at most k entries result in a hash collision. The probability of a hash collision is at most $\frac{k}{s}$ using a union bound. As such for $j > y_i$ we have $\Pr[\tilde{z}_{h_j(i),j} = 1] \leq (1 - \frac{k}{s}) \cdot p + \frac{k}{s} \cdot (1-p) = \frac{1+\frac{\alpha k}{s}}{\alpha+2}$. We let $\frac{1+\frac{\alpha k}{s}}{\alpha+2} = \frac{1}{\gamma+2}$ such that $\mathbb{E}[\bar{y}_i - y_i \mid \bar{y}_i \geq y_i] \leq \frac{4\gamma+4}{\gamma^2}$ by Lemma 2.9 and the calculation above. We isolate γ to find:

$$\begin{aligned} \frac{1}{\gamma+2} &= \frac{1+\frac{\alpha k}{s}}{\alpha+2} \\ (\Leftrightarrow) \qquad \gamma &= \frac{\alpha+2}{1+\frac{\alpha k}{s}} - 2 \ . \end{aligned}$$

Note that $\gamma > 0$ holds due to the requirement $s > 2k$ of Algorithm 2. By conditional expectation, we may upper bound the total expected error by

$$\begin{aligned}\mathbb{E}[|y_i - \tilde{y}_i|] &\leq \mathbb{E}[|y_i - \bar{y}_i|] \\ &\leq \mathbb{E}[y_i - \bar{y}_i \mid \bar{y}_i \leq y_i] + \mathbb{E}[\bar{y}_i - y_i \mid \ \bar{y}_i \geq y_i] \\ &\leq \frac{4\alpha + 4}{\alpha^2} + \frac{4\gamma + 4}{\gamma^2} \ .\end{aligned} \tag{2.1}$$

□

As such we can bound the expected per-entry error for entries with a true value of at most ψ by a function of the parameters α and s. In Section 2.9 we discuss the choice of these parameters based on the upper bond and experiments. For any fixed values of α and $\frac{k}{s}$ we have:

Lemma 2.11. *Let $\alpha = \Theta(1)$ and $s = \Theta(k)$. Then the expected per-entry error of Algorithm 3 is $\mathbb{E}[|x_i - \tilde{x}_i|] \leq \max(0, x_i - \psi) + O(1)$.*

Proof. It follows from Lemmas 2.7 and 2.10 that the expected error for any entry bounded by ψ satisfies:

$$x_i \leq \psi \implies \mathbb{E}[|x_i - \tilde{x}_i|] \leq \left(\frac{1}{2} + \frac{4\alpha + 4}{\alpha^2} + \frac{4\gamma + 4}{\gamma^2}\right) \cdot \alpha \ ,$$

where $\gamma = \frac{\alpha+2}{1+\frac{\alpha k}{s}} - 2$. Entries above ψ have an additional error of up to $x_i - \psi$, since $y_i = m$ and $y_i > m$ are represented identically in the embedding by Algorithm 2. Since α and $\frac{k}{s}$ are constants we have:

$$\mathbb{E}[|x_i - \tilde{x}_i|] \leq \max(0, x_i - \psi) + O(1) \ .$$

□

Next, we bound the tail probabilities for the per-entry error of the mechanism. We bound the error of the estimate $\tilde{y}_i$, which implies bounds on the error of the mechanism.

Lemma 2.12. *Let $\gamma = \frac{\alpha+2}{1+\frac{\alpha k}{s}} - 2$ and $\tau > 0$. Let $p = \frac{1}{\gamma+2}$. For any fixed index $i \in [d]$ when using Algorithm 3 we have:*

$$\Pr[|y_i - \tilde{y}_i| \geq \tau] \leq \frac{2 \cdot (4(p - p^2))^{\tau/2}}{\sqrt{\pi}(1 - 2p)} \ ,$$

Proof. Let S be a simple random walk. We find an upper bound on the probability that the position of the last non-negative step in S is at least τ:

$$\begin{aligned}\Pr[(\max_n : S_n \geq 0) \geq \tau] &= \sum_{j=\lceil \tau/2 \rceil}^{\infty} \binom{2j}{j} (p-p^2)^j (1-2p) \\ &\leq \frac{1-2p}{\sqrt{\pi}} \sum_{j=\lceil \tau/2 \rceil}^{\infty} (4(p-p^2))^j \\ &= \frac{1-2p}{\sqrt{\pi}} \frac{(4(p-p^2))^{\lceil \tau/2 \rceil}}{1-4(p-p^2)} \\ &\leq \frac{(4(p-p^2))^{\tau/2}}{\sqrt{\pi}(1-2p)} \ ,\end{aligned}$$

where the first inequality follows from $\binom{2j}{j} \leq \frac{4^j}{\sqrt{\pi j}}$ when $j \geq 1$ [Elk13]. The last inequality simply follows from $1-4(p-p^2) = (1-2p)^2$ and $4(p-p^2) < 1$. As discussed in the proof of Lemma 2.10, the expectation of $|y_i - \tilde{y}_i|$ can be bounded by two random walks each with p at most $\frac{1}{\gamma+2}$. □

Lemma 2.13. *Let $\gamma = \frac{\alpha+2}{1+\frac{\alpha k}{s}} - 2$ and $p = \frac{1}{\gamma+2}$. For any fixed index $i \in [d]$ with probability at least $1-\beta$ for Algorithm 3 we have:*

$$|y_i - \tilde{y}_i| \leq \frac{2\log\left(\frac{2}{\beta\sqrt{\pi}(1-2p)}\right)}{\log(1/(4p-4p^2))} \ .$$

Proof. We set $\beta = \frac{2\cdot(4(p-p^2))^{\tau/2}}{\sqrt{\pi}(1-2p)}$ and isolate τ such that:

$$\tau = \frac{2\log\left(\frac{2}{\beta\sqrt{\pi}(1-2p)}\right)}{\log(1/(4p-4p^2))} \ .$$

By Lemma 2.12 we have: $\Pr[|y_i - \tilde{y}_i| \leq \tau] \geq 1-\beta$. □

Up to constant factors, the tail probabilities of our mechanism are similar to the properties of the Laplace mechanism summarized in Proposition 2.1. The probabilities depend on the parameters of the mechanism. In Section 2.9 we fix the parameters and evaluate the error in practice. We summarize the tail probabilities for $|x_i - \tilde{x}_i|$ in Lemma 2.14.

Lemma 2.14. *Let* $\gamma = \frac{\alpha+2}{1+\frac{\alpha k}{s}} - 2$, $p = \frac{1}{\gamma+2}$, $x_i \leq \psi$, *and* $\tau \geq \alpha$. *For any fixed index* $i \in [d]$ *when using Algorithm 3 we have:*

$$\Pr[|x_i - \tilde{x}_i| \geq \tau] < \frac{2 \cdot (4(p-p^2))^{(\tau/2\alpha)-1/2}}{\sqrt{\pi}(1-2p)} \ ,$$

With probability at least $1-\beta$ *we have:*

$$|x_i - \tilde{x}_i| < \left(1 + \frac{2\log\left(\frac{2}{\beta\sqrt{\pi}(1-2p)}\right)}{\log(1/(4p-4p^2))}\right) \cdot \alpha \ .$$

Proof. It is easy to see that $|x_i - x'_i| < (1 + |y_i - \tilde{y}_i|) \cdot \alpha$ holds, as the error of random rounding is strictly less than 1. The bounds follow from Lemmas 2.12 and 2.13. □

2.4.3 Generalization to ε-differential privacy

We now generalize the ALP mechanism from 1-differential privacy to satisfying ε-differential privacy. A natural approach is to use a function of ε as the parameter for randomized response in Algorithm 2. The projection algorithm is ε-differentially private if we remove the scaling step and set $p = \frac{1}{\varepsilon+2}$. However, the expected per-entry error would be bounded by $\frac{8\varepsilon+8}{\varepsilon^2}$ by Equation 2.1 (without considering hash collisions), which is as large as $O(1/\varepsilon^2)$ for small values of ε. Other approaches modifying the value of p have a similar expectation. In Section 2.7 we discuss a special case where such an approach is useful for large ε.

In the following, we use a simple pre-processing and post-processing step to achieve optimal error. The idea is to scale the input vector as well as the parameter ψ by ε before running Algorithm 2. We scale back the estimates from Algorithm 3 by $1/\varepsilon$. These generalizations are given as Algorithm 4 and Algorithm 5, respectively.

Lemma 2.15. *Algorithm 4 satisfies* ε*-differential privacy.*

Proof. It follows from the proof of Lemma 2.4 that for any subset of outputs S we have $\frac{\Pr[\text{ALP1-Projection}(\hat{x}')\in S]}{\Pr[\text{ALP1-Projection}(\hat{x})\in S]} \leq e^{\|\hat{x}-\hat{x}'\|_1}$. As such for any pair of neighboring vectors x and x' we have:

$$\begin{aligned}\frac{\Pr[\text{ALP-Projection}(x') \in S]}{\Pr[\text{ALP-Projection}(x) \in S]} &= \frac{\Pr[\text{ALP1-Projection}(\hat{x}') \in S]}{\Pr[\text{ALP1-Projection}(\hat{x}) \in S]} \\ &\leq e^{\|\hat{x}-\hat{x}'\|_1} = e^{\varepsilon\cdot\|x-x'\|_1} \leq e^{\varepsilon} \ .\end{aligned}$$

Algorithm 4: ALP-Projection

Parameters: $\alpha, \psi, \varepsilon > 0$, and $s \in \mathbb{N}$.
Input : k-sparse vector $x \in \mathbb{R}_+^d$, where $s > 2k$. Sequence of universal hash functions from domain $[d]$ to $[s]$, $h = (h_1, \ldots, h_m)$, where $m = \left\lceil \frac{\psi\varepsilon}{\alpha} \right\rceil$.
Output : ε-differentially private representation of x.

(1) Scale the entries of x such that $\hat{x}_i = x_i \cdot \varepsilon$.
(2) Let $h, \tilde{z} = \text{ALP1-Projection}_{\alpha,\psi\cdot\varepsilon,s}(\hat{x}, h)$.
(3) Release h and $\tilde{z}$.

Algorithm 5: ALP-Estimator

Parameters: $\alpha, \varepsilon > 0$.
Input : Embedding $\tilde{z} \in \{0,1\}^{s\times m}$. Sequence of universal hash functions $h = (h_1, \ldots, h_m)$. Index $i \in [d]$.
Output : Estimate of x_i.

(1) Let $\tilde{x}_i = \text{ALP1-Estimator}_\alpha(\tilde{z}, h, i)$.
(2) Return $\frac{\tilde{x}_i}{\varepsilon}$.

□

Lemma 2.16. *Let $\alpha = \Theta(1)$ and $s = \Theta(k)$. The output of Algorithm 4 can be stored using $O((k + \log d)\psi\varepsilon)$ bits.*

Proof. It follows directly from Lemma 2.5. □

Lemma 2.17. *Let $\alpha = \Theta(1)$ and $s = \Theta(k)$. Then the expected per-entry error of Algorithm 5 is $\mathbb{E}[|x_i - \tilde{x}_i|] \leq \max(0, x_i - \psi) + O(1/\varepsilon)$ and the evaluation time is $O(\psi\varepsilon)$.*

Proof. It is clear that the error of Algorithm 5 is $\frac{1}{\varepsilon}$ times the error of Algorithm 3 for entries at most ψ. As such the expected per-entry error follows from Lemma 2.11. The evaluation time follows directly from Lemma 2.6. □

Lemma 2.18. *Let $\gamma = \frac{\alpha+2}{1+\frac{\alpha k}{s}} - 2$, $p = \frac{1}{\gamma+2}$, $x_i \leq \psi$, and $\tau \geq \frac{\alpha}{\varepsilon}$. For any fixed index $i \in [d]$ when using Algorithm 5 we have:*

$$\Pr[|x_i - \tilde{x}_i| \geq \tau] < \frac{2 \cdot (4(p - p^2))^{(\tau\varepsilon/2\alpha)-1/2}}{\sqrt{\pi}(1-2p)} \ ,$$

With probability at least $1 - \beta$ *we have:*

$$|x_i - \tilde{x}_i| < \left(1 + \frac{2\log\left(\frac{2}{\beta\sqrt{\pi}(1-2p)}\right)}{\log(1/(4p-p^2))}\right) \cdot \frac{\alpha}{\varepsilon} \ .$$

Proof. It follows directly from Lemma 2.14. □

We are now ready to state the following theorem which summarizes the properties of the ALP mechanism.

Theorem 2.2. *Let* $\alpha = \Theta(1)$ *and* $s = \Theta(k)$*. Then there exists an algorithm where the expected per-entry error is* $O(1/\varepsilon)$ *for all entries, the access time is* $O(u\varepsilon)$*, and the space usage is* $O((k + \log d)u\varepsilon)$ *bits.*

Proof. By setting $\psi = u$, it follows directly from Lemmas 2.16 and 2.17. □

The space usage and access time of the mechanism both scale linearly with the parameter ψ. As such the mechanism performs well only for small values of u. However, in many contexts u scales with the input size. One example is a histogram, where u is the number of rows in the underlying dataset. Next, we show how to handle such cases.

2.5 Combined data structure

In this section, we combine the ALP mechanism with techniques from previous work to improve space requirements and access time. As shown in Theorem 2.2 the ALP mechanism performs well when all entries are bounded by a small value. The per-entry error is low only for entries bounded by ψ but the space requirements and access time scale linearly with ψ. Some of the algorithms from previous work perform well for large entries but have large per-entry error for small values. The idea of this section is to combine the ALP mechanism with such an algorithm to construct a composite data structure that performs well for both small and large entries.

To handle large values, we use the thresholding technique from Cormode et al. [CPST12]. It adds noise to each entry, but only stores entries above a threshold. The pseudocode of the algorithm is given as Algorithm 6.

Algorithm 6: Threshold [CPST12]

Parameters: $\varepsilon, t > 0$.
Input : k-sparse vector $x \in \mathbb{R}_+^d$.
Output : ε-differentially private representation of x.

(1) Let $v_i = x_i + \eta_i$ for all $i \in [d]$, where $\eta_i \sim \text{Laplace}(1/\varepsilon)$.
(2) Truncate entries below t:

$$\tilde{v}_i = \begin{cases} v_i, & \text{if } y_i \geq t \\ 0, & \text{otherwise} \end{cases}$$

(3) Return $\tilde{v}$.

Lemma 2.19. *Algorithm 6 satisfies ε-differential privacy.*

Proof. The algorithm is equivalent to the Laplace mechanism followed by post-processing. The Laplace mechanism satisfies ε-differential privacy, and privacy is preserved under post-processing as stated by Lemma 2.1. □

Lemma 2.20. *Let $t = \frac{2\ln(d)}{\varepsilon}$. Then the output of Algorithm 6 is k-sparse with probability at least $1 - \frac{1}{2d}$.*

Proof. Using Definition 2.2 we find that the probability of storing a zero entry of x is:

$$\Pr[\text{Laplace}(1/\varepsilon) \geq t] = \Pr[\text{Laplace}(1/\varepsilon) \leq -t] = \frac{1}{2}e^{-t\varepsilon} = \frac{1}{2d^2} \ .$$

If the output is not k-sparse there must exists at least one coordinate i such that $\tilde{v}_i \neq 0$ and $x_i = 0$. By a union bound such a coordinate exists with probability at most $1/(2d)$. □

As discussed in Section 2.3, the expected per-entry error of Algorithm 6 is $O\left(\frac{\log(d)}{\varepsilon}\right)$ for worst-case input. We combine the algorithm with the ALP mechanism from the previous section to achieve $O(1/\varepsilon)$ expected per-entry error for any input. We use the threshold parameter t as value for parameter ψ in Algorithm 4. The algorithm is presented in Algorithm 7. We use a separate privacy parameter for each part of the algorithm. Throughout this section we assume that the ratio between them is fixed such that $\varepsilon_1 = \Theta(\varepsilon_2)$. We discuss what happens if this ratio is not fixed after Lemma 2.23.

Algorithm 7: Threshold ALP-Projection

Parameters: $\alpha, \varepsilon_1, \varepsilon_2 > 0$, and $s \in \mathbb{N}$.
Input : k-sparse vector $x \in \mathbb{R}_+^d$, where $s > 2k$. Sequence of universal hash functions from domain $[d]$ to $[s]$, $h = (h_1, \ldots, h_m)$, where $m = \left\lceil \frac{t\varepsilon_2}{\alpha} \right\rceil$.
Output : $(\varepsilon_1 + \varepsilon_2)$-differentially private representation of x.

(1) Let $t = \frac{2\ln(d)}{\varepsilon_1}$.
(2) Let $\tilde{v} = \text{Threshold}_{\varepsilon_1,t}(x)$.
(3) Let $h, \tilde{z} = \text{ALP-Projection}_{\alpha,\varepsilon_2,t,s}(x, h)$
(4) Return $\tilde{v}$, h and $\tilde{z}$.

Lemma 2.21. *Algorithm 7 satisfies $(\varepsilon_1 + \varepsilon_2)$-differential privacy.*

Proof. The two parts of the algorithm are independent as there is no shared randomness. The first part of the algorithm satisfies ε_1-differential privacy by Lemma 2.19 and the second part satisfies ε_2-differential privacy by Lemma 2.15. As such it follows directly from composition (Lemma 2.2) that Algorithm 7 satisfies $(\varepsilon_1 + \varepsilon_2)$-differential privacy. □

Lemma 2.22. *Let $\alpha = \Theta(1)$, $s = \Theta(k)$, $\varepsilon_1 = \Theta(\varepsilon_2)$. Then the output of Algorithm 7 is stored using $O(k\log(d+u) + \log^2(d))$ bits with probability at least $1 - \frac{1}{2d}$.*

Proof. It follows from Lemma 2.20 that we can store $\tilde{v}$ using $O(k\log(d+u))$ bits with probability at least $1 - \frac{1}{2d}$. Since $\psi = t$ it follows from Lemma 2.16 that we can store h and $\tilde{z}$ using $O((k + \log(d))t\varepsilon_2) = O(k\log(d) + \log^2(d))$ bits. □

To estimate an entry, we access $\tilde{v}$ when a value is stored for the entry and the ALP embedding otherwise. This algorithm is presented in Algorithm 8.

Lemma 2.23. *Let $\alpha = \Theta(1)$, $s = \Theta(k)$, and $\varepsilon = \varepsilon_1 + \varepsilon_2$ with $\varepsilon_1 = \Theta(\varepsilon_2)$. Let $\tilde{v}$, h, and $\tilde{z}$ be the output of Algorithm 7 given these parameters. Then the evaluation time of Algorithm 8 is $O(\log(d))$. The expected per-entry error is $O(1/\varepsilon)$ and the expected maximum error is $O\left(\frac{\log(d)}{\varepsilon}\right)$.*

Proof. The evaluation time follows from Lemma 2.17. That is, the evaluation time is $O(\psi\varepsilon) = O(t\varepsilon) = O(\log(d))$.

Algorithm 8: Threshold ALP-Estimator

Parameters: $\alpha, \varepsilon_2 > 0$.
Input : Vector $\tilde{v} \in \mathbb{R}_+^d$. Embedding $\tilde{z} \in \{0,1\}^{s \times m}$. Sequence of universal hash functions $h = (h_1, \ldots, h_m)$. Index $i \in [d]$.
Output : Estimate of x_i.

(1) Estimate the entry using either the vector or the embedding such that:

$$\tilde{x}_i = \begin{cases} \tilde{v}_i, & \text{if } \tilde{v}_i \neq 0 \\ \text{ALP-Estimator}_{\varepsilon_2,\alpha}(\tilde{z}, h, i), & \text{otherwise} \end{cases}$$

(2) Return $\tilde{x}_i$.

The error depends on both parts of the algorithm. The expected per-entry error for the ith entry is $\max(0, x_i - \psi) + O(1/\varepsilon_2)$ when $\tilde{v}_i = 0$ by Lemma 2.17. That is, when η_i is less than $\psi - x_i$ in Algorithm 6. When $\tilde{v}_i \neq 0$ the error is the absolute value of η_i. That is, we can analyze it using conditional probability and the probability density function of the Laplace distribution from Definition 2.2.

$$\begin{aligned}
\mathbb{E}[|x_i - \tilde{x}_i|] &= \mathbb{E}[|x_i - \tilde{x}_i| \mid \tilde{v}_i = 0] \cdot \Pr[\tilde{v}_i = 0] \\
&\quad + \mathbb{E}[|x_i - \tilde{x}_i| \mid \tilde{v}_i \neq 0] \cdot \Pr[\tilde{v}_i \neq 0] \\
&\leq (\max(0, x_i - \psi) + O(1/\varepsilon_2)) \cdot \Pr[\text{Laplace}(1/\varepsilon_1) < \psi - x_i] \\
&\quad + \int_{\psi - x_i}^{\infty} |v - x_i| \cdot \frac{\varepsilon_1}{2} e^{-|v - x_i|\varepsilon_1} \, dv \\
&< \int_{-\infty}^{\psi - x_i} (|v - x_i| + O(1/\varepsilon_2)) \cdot \frac{\varepsilon_1}{2} e^{-|v - x_i|\varepsilon_1} \, dv \\
&\quad + \int_{\psi - x_i}^{\infty} |v - x_i| \cdot \frac{\varepsilon_1}{2} e^{-|v - x_i|\varepsilon_1} \, dv \\
&< \int_{-\infty}^{\infty} |v - x_i| \cdot \frac{\varepsilon_1}{2} e^{-|v - x_i|\varepsilon_1} \, dv + O(1/\varepsilon_2) \\
&= O(1/\varepsilon_1) + O(1/\varepsilon_2) = O(1/\varepsilon) \ .
\end{aligned}$$

The expected maximum error of Algorithm 6 is $O\left(\frac{\log(d)}{\varepsilon}\right)$ and the output of the Algorithm 5 is at most ψ. Since $\psi = O\left(\frac{\log(d)}{\varepsilon}\right)$ the expected maximum error is $O\left(\frac{\log(d)}{\varepsilon}\right)$. □

The asymptotic properties of the algorithms hold for any fixed ratio between ε_1 and ε_2. A natural choice is to set $\varepsilon_1 = \varepsilon_2$. However, the ratio

affects constant factors, and it might not be the best choice in practice. The value of the parameter m and in turn the space consumption and access time of the projection scales with $\varepsilon_2/\varepsilon_1$. But the parameters also affect the expected error for each part of the algorithm. Furthermore, the constant factor for the error of the Laplace mechanism is lower than that of the projection. Therefore one could set ε_2 higher than ε_1 to balance out those constant factors. We explore the constant factors of Algorithm 3 further in Section 2.9.

2.5.1 Removing the dependency on dimension

To make access time independent of the dimension d, we can turn to approximate differential privacy. This allows us to use a smaller threshold in the initial thresholding approach, which in turn results in smaller values for ψ in the ALP mechanism.

The following algorithm is similar to that introduced by Korolova et al. [KKMN09], which we discussed in Section 2.3. It adds noise to non-zero entries only, and uses a threshold to satisfy approximate differential privacy. Our algorithm differs from the work of Korolova et al. by using a random rounding step. This step is not needed in a discrete setting, where at most a single zero-valued entry is changed to a non-zero entry for neighboring vectors. However, in the real-valued context, several zero entries can change.

Lemma 2.24. *Algorithm 9 satisfies (ε, δ)-differential privacy.*

Proof. Let x and x' be neighboring vectors. We consider two additional vectors $\hat{x}$ and $\hat{x}'$ such that:

$$\hat{x}_i = \begin{cases} \min(1, x'_i), & \text{if } x_i \leq 1 \\ x_i, & \text{otherwise;} \end{cases}$$

$$\hat{x}'_i = \begin{cases} 1, & \text{if } x'_i < 1 \text{ and } 1 < x_i \\ x'_i, & \text{otherwise.} \end{cases}$$

The vectors are constructed such that x and $\hat{x}$ can only differ for entries at most 1 in both vectors. The same holds for x' and $\hat{x}'$. Additionally, the ℓ_1-distance is still at most 1 between any pair of vectors.

Algorithm 9: Threshold2 (Following technique by [KKMN09])

Parameters: $\varepsilon, \delta > 0$.
Input : k-sparse vector $x \in \mathbb{R}_+^d$.
Output : (ε, δ)-differentially private approximation of x.

(1) Apply random rounding to non-zero entries below 1 such that:

$$y_i = \begin{cases} \text{RandRound}\,(x_i)\,, & \text{if } 0 < x_i < 1 \\ x_i, & \text{otherwise} \end{cases}$$

(2) Let $v_i = y_i + \eta_i$ for all non-zero entries, where $\eta_i \sim \text{Laplace}(1/\varepsilon)$.
(3) Let $t = \frac{\ln(1/\delta)}{\varepsilon} + 2$.
(4) Truncate entries below t:

$$\tilde{v}_i = \begin{cases} v_i, & \text{if } y_i \neq 0 \text{ and } v_i \geq t \\ 0, & \text{otherwise} \end{cases}$$

(5) Return $\tilde{v}$.

We find the probability of outputting anything for an entry less than or equal to 1 as:

$$\begin{aligned} x_i \leq 1 \implies \Pr[\tilde{v}_i \neq 0] &= \Pr[y_i = 1] \cdot \Pr[\text{Laplace}(1/\varepsilon) \geq t - 1] \\ &= x_i \cdot \Pr[\text{Laplace}(1/\varepsilon) \leq -(t-1)] \\ &= x_i \cdot \frac{1}{2} e^{-(t-1)\varepsilon} = x_i \cdot \frac{1}{2} e^{-\ln(1/\delta)-\varepsilon} = \frac{x_i \delta}{2 \cdot e^{\varepsilon}} \ . \end{aligned}$$

Since x and $\hat{x}$ only differ for entries less than or equal to 1 we have for any subset of outputs S:

$$\begin{aligned} \Pr[\text{Threshold2}(x) \in S] &\leq \Pr[\text{Threshold2}(\hat{x}) \in S] + \sum_{i \in [d]} |\hat{x}_i - x_i| \frac{\delta}{2 \cdot e^{\varepsilon}} \\ &\leq \Pr[\text{Threshold2}(\hat{x}) \in S] + \frac{\delta}{2 \cdot e^{\varepsilon}} \ . \end{aligned}$$

The inequality holds in both directions and for the pair of x' and $\hat{x}'$ as well.

By definition $\hat{x}$ and $\hat{x}'$ only differ for entries of at least 1. As such we can ignore the random rounding step and we have:

$$\begin{aligned} \Pr[\text{Threshold2}(\hat{x}) \in S] &\leq e^{\|\hat{x} - \hat{x}'\|_1 \varepsilon} \Pr[\text{Threshold2}(\hat{x}') \in S] \\ &\leq e^{\varepsilon} \cdot \Pr[\text{Threshold2}(x) \in S] \ . \end{aligned}$$

Using the inequalities above we have:

$$\begin{aligned}\Pr[\text{Threshold2}(x) \in S] &\leq \Pr[\text{Threshold2}(\hat{x}) \in S] + \frac{\delta}{2e^{\varepsilon}} \\ &\leq e^{\varepsilon} \cdot \Pr[\text{Threshold2}(\hat{x}') \in S] + \frac{\delta}{2e^{\varepsilon}} \\ &\leq e^{\varepsilon} \cdot \left(\Pr[\text{Threshold2}(x') \in S] + \frac{\delta}{2e^{\varepsilon}}\right) + \frac{\delta}{2e^{\varepsilon}} \\ &\leq e^{\varepsilon} \cdot \Pr[\text{Threshold2}(x') \in S] + \delta \ .\end{aligned}$$

□

Lemma 2.25. *Let $\delta = O\left(\frac{1}{k}\right)$. Then the expected maximum error of Algorithm 9 is $O\left(\frac{\log(1/\delta)}{\varepsilon}\right)$.*

Proof. The expected maximum error added by the Laplace noise is $O\left(\frac{\log(k)}{\varepsilon}\right)$, since we add noise to at most k entries. By removing entries we add error of up to $O\left(\frac{\log(1/\delta)}{\varepsilon}\right)$. As such the expected maximum error for worst-case input is:

$$\mathbb{E}[\|x - \tilde{v}\|_{\infty}] \leq O\left(\frac{\log(k)}{\varepsilon}\right) + O\left(\frac{\log(1/\delta)}{\varepsilon}\right) = O\left(\frac{\log(1/\delta)}{\varepsilon}\right) \ .$$

□

In the following, we use Algorithm 9 instead of Algorithm 6 in Step (2) of Algorithm 7.

Lemma 2.26. *Let $\alpha = \Theta(1)$, $s = \Theta(k)$, $\varepsilon = \varepsilon_1 + \varepsilon_2$ with $\varepsilon_1 = \Theta(\varepsilon_2)$, and $\delta > 0$ with $\delta = O(1/k)$. By using Algorithm 9 in Algorithm 7 the access time is $O(\log(1/\delta))$. The expected per-entry error is $O(1/\varepsilon)$ and the expected maximum error is $O\left(\frac{\log(1/\delta)}{\varepsilon}\right)$. The combined mechanism satisfies (ε, δ)-differential privacy.*

Proof. The proof is the same as the proofs of Lemmas 2.21 and 2.23. □

Lemma 2.27. *Let $\alpha = \Theta(1)$, $s = \Theta(k)$, and $\varepsilon_1 = \Theta(\varepsilon_2)$. Then the memory requirement of combining Algorithm 9 and the ALP mechanism is $O(k \log(d + u) + k \log(1/\delta) + \log(d) \log(1/\delta))$.*

Proof. The output of Algorithm 9 is always k-sparse and we represent it using $O(k\log(d+u))$ bits. We set $\psi = \frac{\ln(1/\delta)}{\varepsilon_2} + 2$ and therefore h and $\tilde{z}$ are represented using $O(k\log(1/\delta) + \log(d)\log(1/\delta))$ bits by Lemma 2.16. □

We are now ready to summarize our results for both pure and approximate differential privacy.

Theorem 2.3. *Let $\alpha = \Theta(1)$, $s = \Theta(k)$, and $\varepsilon > 0$. Then there exists an ε-differentially private algorithm with $O(1/\varepsilon)$ expected per-entry error, $O\left(\frac{\log(d)}{\varepsilon}\right)$ expected maximum error, access time of $O(\log(d))$, and space usage of $O(k\log(d+u) + \log^2(d))$ with probability at least $1 - \frac{1}{2d}$.*

Proof. It follows directly from Lemmas 2.21, 2.22 and 2.23. □

Theorem 2.4. *Let $\alpha = \Theta(1)$, $s = \Theta(k)$, and $\varepsilon, \delta > 0$. Then there exists an (ε, δ)-differentially private algorithm with $O(1/\varepsilon)$ expected per-entry error, $O\left(\frac{\log(1/\delta)}{\varepsilon}\right)$ expected maximum error, access time of $O(\log(1/\delta))$, and space usage of $O(k\log(d+u) + k\log(1/\delta) + \log(d)\log(1/\delta))$.*

Proof. It follows directly from Lemmas 2.24, 2.26 and 2.27. □

2.6 Faster evaluation with multiplicative error

From the previous sections, we know how to achieve evaluation time $O(\log d)$ and $O(\log(1/\delta))$, respectively. In this section, we improve the evaluation time to $O(\log(\log(d)/\varepsilon))$ and $O(\log(\log(1/\delta)/\varepsilon))$ at the cost of a *multiplicative error* $O(1)$. That is, this technique can be used if it sufficient to estimate the *order of magnitude* of an entry. We first describe the data structure and the estimation algorithm, and then state and analyze its properties.

Projection.

Let $\alpha, \varepsilon > 0, B > 1, \psi \geq \max\{1, 1/\ln(B)\}$, and $s \in \mathbb{N}$. Given a k-sparse vector $x \in \mathbb{R}_+^d$ with $x_i \leq \psi$, define $\dot{x} \in \mathbb{R}_+^d$ by $\dot{x}_i = \max\{\log_B(x_i \ln B), 0\}$ for all $x_i > 0$. Run Algorithm 4 with input $\dot{x}$ and a sequence of universal hash functions from domain $[d]$ to $[s]$, $h = (h_1, \ldots, h_m)$, where $m = \left\lceil \frac{\log_B(\psi \ln B)\varepsilon}{\alpha} \right\rceil$. Let $\tilde{z}$ be the output of Algorithm 4.

Estimation.

Given $\tilde{z}$, h, and an index $i \in [d]$, let $\ddot{x}_i$ be the output of Algorithm 5 with parameters α and ε. Return the value $\tilde{x}_i = B^{\ddot{x}_i} / \ln B$ as an estimate for x_i.

Lemma 2.28. *The projection algorithm satisfies ε-differential privacy.*

Proof. By Lemma 2.15, Algorithm 4 satisfies ε-differential privacy for neighboring inputs. We have to show that the mapping $x_i \mapsto \max\{\log_B(x_i \ln B), 0\}$ preserves the neighborhood relation for neighboring x and x'. The function $f(x) = \log_B(x \ln B)$ is Lipschitz continuous in $\mathbb{R}_{>1/\ln B}$ since if $x \geq 1/\ln B$, the absolute value of the derivative $f'(x) = \frac{1}{x \ln B}$ is at most 1, and thus for $x, x' \geq 1/\ln B$:

$$||\log_B(x \ln B) - \log_B(x' \ln B)||_1 \leq 1 \cdot ||x - x'||_1.$$

Since the mapping $x_i \mapsto \max\{\log_B(x_i \ln B), 0\}$ is constant for $x_i \leq 1/\ln B$ we conclude that it is neighborhood-preserving. □

We next consider the properties of this data structure.

Lemma 2.29. *Let $\varepsilon > 0$ and $x \in \mathbb{R}_+^d$ be a k-sparse vector with each coordinate $x_i \leq \psi$. Let $\alpha = \Theta(1)$ and $s = \Theta(k)$. Let $\gamma = \frac{\alpha+2}{1+\frac{\alpha k}{s}} - 2$, $p = \frac{1}{\gamma+2}$, and set $B = (4(p - p^2))^{-\varepsilon/(4\alpha)}$. Then*

- *for all $x_i \geq \frac{1}{\ln B}$ the (multiplicative) expected per-entry error is*

$$\mathbb{E}[\max\{\tilde{x}_i / x_i, x_i / \tilde{x}_i\}] = O(1) \quad .$$

- *the evaluation time is $O(\log(\psi))$, and*
- *the space is $O((k + \log(d)) \log(\psi))$ bits.*

Proof. Let $\dot{x}$ be the transformed input for Algorithm 4 and $\ddot{x}$ the estimate returned by Algorithm 5. The multiplicative estimation errors of $\tilde{x}_i / x_i$ and $x_i / \tilde{x}_i$ are bounded by $B^{|\ddot{x}_i - \dot{x}_i|}$. Let S be a simple random walk with parameter p. We can bound the estimation error by considering the last non-negative step of S similar to the proof of Lemma 2.10. The absolute error between $\dot{x}_i \varepsilon / \alpha$ and the estimate computed in Step (3) of Algorithm 3 is at most one more than the length of longest interval where half the bits are flipped due to rounding. Using the definition of the expectation

and the calculations from Lemma 2.12, we upper bound the expected multiplicative error as follows:

$$\begin{aligned}\mathbb{E}\left[\max\{\tilde{x}_i/x_i, x_i/\tilde{x}_i\}\right] &\leq \sum_{\tau=0}^{\infty} B^{\frac{(2\tau+1)\alpha}{\varepsilon}} \cdot 2\Pr[(\max_n : S_n \geq 0) = 2\tau] \\ &\leq (2-4p)B^{\alpha/\varepsilon} \sum_{\tau=0}^{\infty} \left(B^{2\alpha/\varepsilon} 4\left(p-p^2\right)\right)^{\tau}\end{aligned}$$

We set $B = \left(4(p-p^2)\right)^{-\varepsilon/(4\alpha)}$ such that

$$\begin{aligned}&(2-4p)B^{\alpha/\varepsilon} \sum_{\tau=0}^{\infty} \left(B^{2\alpha/\varepsilon} 4\left(p-p^2\right)\right)^{\tau} \\ &\qquad = \frac{2-4p}{(4p-4p^2)^{1/4}} \sum_{\tau=0}^{\infty} \left(\sqrt{4p-4p^2}\right)^{\tau} \\ &\qquad = \frac{2-4p}{(4p-4p^2)^{1/4} - (4p-4p^2)^{3/4}} = O(1) \ .\end{aligned}$$

With our choice of B, $\log_B(x) = O(\log(x)/\varepsilon)$ and $\log_B(\ln(B)) < 1$ holds for any choice of B. The statements about running time and space usage follow directly from Theorem 2.2 using $u = 1 + O(\log(\psi)/\varepsilon)$. □

We remark that capping $\dot{x}$ to zero means that we treat entries where $x_i \leq 1/\ln B$ as $4\alpha/(\ln(1/(4p-4p^2))\varepsilon) = \Theta(1/\varepsilon)$ for our choice of B, incurring an additive error of $\Theta(1/\varepsilon)$ for small entries. The proof above shows that the expected multiplicative error is bounded by a constant for any fixed α. If α is not fixed we can choose the value to minimize the multiplicative error. As an simplified example, we ignore hash collisions such that $p = \frac{1}{\alpha+2}$. Then the equation for the constant above is minimized with value ≈ 4.83 when $\alpha \approx 20.26$. However, the additive error is minimized for $\alpha \approx 3.07$ where $1/\ln B \approx 26.89/\varepsilon$. As such the choice of α depends on the trade-off between the two kinds of error. It is worth noting that the analysis for the expected multiplicative error is not tight. Simulations similar to those in Section 2.9 can be used to balance the trade-off based on empirical mean error.

2.6.1 Applications

We summarize the properties of the data structure in the settings studied before:

1. If we do not combine the data structure with the thresholding technique as described in Section 2.5, we instantiate the algorithm described above with $\psi = u$. The running time is $O(\log u)$ and the data structure uses $O((k + \log(d)) \log u)$ bits.

2. When combined with the thresholding technique with threshold $O(\log(d)/\varepsilon)$ for pure differential privacy or $O(\log(1/\delta)/\varepsilon)$ for (ε, δ)-differential privacy, we may set ψ to these values, respectively. This results in running time $O(\log(\log(d)/\varepsilon))$ and $O(\log(\log(1/\delta)/\varepsilon))$ with a space usage of $O((k + \log(d)) \log(\log(d)/\varepsilon))$ and $O((k + \log(d)) \log(\log(1/\delta)/\varepsilon))$ bits, respectively, not accounting for the space needed for the threshold data structure.

2.7 Improvements for Sparse Integer-valued Vectors

The algorithms we introduced so far work with real-valued vectors as input. In this section, we discuss a variation of the ALP mechanism if we restrict the input to integers. Recall that two vectors $x, x' \in \mathbb{R}_+^d$ are defined as neighboring iff $\|x - x'\|_1 \leq 1$. Under this definition neighboring vectors might disagree on several entries. As an example, two neighboring k-sparse vectors can differ by $\frac{1}{2k}$ in $2k$ entries. However, for the special case of histograms, that is $x, x' \in \mathbb{N}^d$, neighboring input may only disagree on one entry. We can utilize this to design a version of the ALP mechanism with improved accuracy for some values of ε. Algorithm 10 shows a projection algorithm for histograms. It is similar to Algorithm 4 without the scaling and rounding steps, and with a flip probability in randomized response that depends on ε.

Lemma 2.30. *Algorithm 10 satisfies ε-differential privacy.*

Proof. Neighboring histograms only differ in a single entry. As such at most one bit in z is changed from replacing x with x'. For randomized response with $p = \frac{1}{e^\varepsilon + 1}$ it holds for each $b \in \{0, 1\}$ that

$$\frac{\Pr[\text{RandResponse}\,(b, p) = b]}{\Pr[\text{RandResponse}\,(b, p) = 1 - b]} = \frac{1 - p}{p} = e^\varepsilon \ .$$

By symmetry the mechanism is ε-differentially private.

□

Algorithm 10: Histogram-Projection

Parameters: $\varepsilon > 0$, and $\psi, s \in \mathbb{N}$.
Input : k-sparse histogram $x \in \mathbb{N}^d$ where $s > 2k$. Sequence of universal hash functions from domain $[d]$ to $[s]$, $h = (h_1, \ldots, h_\psi)$.
Output : ε-differentially private representation of x.

(1) Construct $z \in \{0,1\}^{s \times \psi}$ by hashing the unary representations of x such that:

$$z_{a,b} = \begin{cases} 1, & \exists i : \ b \leq x_i \text{ and } h_b(i) = a \\ 0, & \text{otherwise} \end{cases}$$

(2) Apply randomized response to each bit of z such that $\tilde{z}_{a,b} = \text{RandResponse}\left(z_{a,b}, \frac{1}{e^\varepsilon + 1}\right)$.

(3) Release h and $\tilde{z}$.

A key feature of Algorithm 10 is that the parameter for randomized response is a function of ε. For comparison, in Algorithm 4 it depends on the adjustable parameter α, and ε is only used for a linear scaling. For this reason Algorithm 10 is preferred for large values of ε since the noise is significantly reduced as we show next. However, the technique used in Algorithm 4 is still preferred for small ε.

For the error analysis, we first consider the expected error when there are no hash collisions. We later include hash collisions in the analysis. We estimate the value of an entry with Algorithm 3. We assume that the true value is at most ψ. If this is not true there is an additional error as shown in Lemma 2.11.

Lemma 2.31. *The expected per-entry error when using Algorithm 3 on output from Algorithms 10 for entries at most ψ is bounded by $\frac{8 \cdot e^\varepsilon}{(e^\varepsilon - 1)^2}$ if there are no hash collisions.*

Proof. By the argument used in the proof of Lemma 2.10 we can bound the error by examining simple random walks with $p = \frac{1}{e^\varepsilon + 1}$. The expected error is at most twice the expected last non-negative step in the random walk. By Lemma 2.9 this is $\frac{4(p - p^2)}{(1-2p)^2}$. As such we can bound the expected

error by:

$$2 \cdot \frac{4(p-p^2)}{(1-2p)^2} = 8 \cdot \frac{\frac{e^\varepsilon}{(e^\varepsilon+1)^2}}{\frac{(e^\varepsilon-1)^2}{(e^\varepsilon+1)^2}} = \frac{8e^\varepsilon}{(e^\varepsilon-1)^2} \ .$$

□

Recall from Lemma 2.17 that the expected error of Algorithm 5 is $O(1/\varepsilon)$. As such, we expect the above algorithm to perform better for "large values" of ε as the expectation approaches $O(1/e^\varepsilon)$ for $\varepsilon \to \infty$. However, it performs worse for "small values" as the expectation approaches $O(1/\varepsilon^2)$ for $\varepsilon \to 0$. In Section 2.9 we compare the error of the two algorithms for varying values of ε.

Next we consider the effect of hash collisions on the expected error. From previous sections we know that it is sufficient to bound the probability of hash collisions by a constant for the general ALP mechanism. That is however not sufficient for large ε as the probability used for randomized response is very low. That is, a hash collision has a bigger impact on the probability of outputting a one for larger ε.

Lemma 2.32. *The expected per-entry error when using Algorithm 3 on output from Algorithms 10 for entries at most ψ is bounded by* $\frac{4e^\varepsilon}{(e^\varepsilon-1)^2} + \frac{\frac{4e^\varepsilon+4}{1+(e^\varepsilon-1)\frac{k}{s}}-4}{\left(\frac{e^\varepsilon+1}{1+(e^\varepsilon-1)\frac{k}{s}}-2\right)^2}$.

Proof. We know from the proof of Lemma 2.10 that the expected last non-negative step of a simple random walk with $p = \frac{1}{\gamma+2}$ is $\frac{4\gamma+4}{\gamma^2}$. Since we use universal hash functions and store at most k ones in each column the probability of hash collisions is at most $\frac{k}{s}$. As such we can bound the probability of changing a bit from zero to one by $\frac{1+(e^\varepsilon-1)\frac{k}{s}}{e^\varepsilon+1}$. By setting $\frac{1}{\gamma+2} = \frac{1+(e^\varepsilon-1)\frac{k}{s}}{e^\varepsilon+1}$ and isolating γ we get $\gamma = \frac{e^\varepsilon+1}{1+(e^\varepsilon-1)\frac{k}{s}} - 2$. As such we can bound the positive error by

$$\frac{4\gamma+4}{\gamma^2} = \frac{\frac{4e^\varepsilon+4}{1+(e^\varepsilon-1)\frac{k}{s}}-4}{\left(\frac{e^\varepsilon+1}{1+(e^\varepsilon-1)\frac{k}{s}}-2\right)^2} \ .$$

The expected negative error is still bounded by $\frac{4e^\varepsilon}{(e^\varepsilon-1)^2}$ since hash collisions have no impact. □

If we apply the thresholding techniques before running Algorithm 10 the number of bits needed to store $\tilde{z}$ is $O((s \log d)/\varepsilon)$ and $O((s \log(1/\delta))/\varepsilon)$, respectively. By setting $s = \Theta(k)$ we bound the probability of hash collisions by a constant. This works decently for small values of ε, but we need to use more space for large values. When the probability of a hash collision is q, the probability of flipping a bit from zero to one is at least $q/2$. This would put a lower bound on the error for any ε. We can use some extra space to get error $O(1/\varepsilon)$ or $O(1/e^{\varepsilon})$ expected error for large epsilon as summarized below. The corollary follows directly from plugging in different choices of s into Lemma 2.32.

Corollary 2.1. *Assuming that the threshold technique for ε-DP (Algorithm 6) was applied, using Algorithm 3 on output from Algorithms 10 has the following properties:*

1. *Let $s = \Theta(k\varepsilon)$ and $\varepsilon > 1$. Then the expected error for $\varepsilon \to \infty$ is $O(1/\varepsilon)$ and $\tilde{z}$ is stored in $O(k \log d)$ bits.*
2. *Let $s = \Theta(ke^{\varepsilon})$ and $\varepsilon > 1$. Then the expected error for $\varepsilon \to \infty$ is $O(1/e^{\varepsilon})$ and $\tilde{z}$ is stored in $O(e^{\varepsilon} k \log(d)/\varepsilon)$ bits.*

The results extend naturally to the case of (ε, δ)-DP using Algorithm 9.

2.8 Constant Access Time with Optimal Expected Error

In this section, we discuss a data structure that achieves access time $O(1)$ with expected per-entry error $O(1/\varepsilon)$, improving on the mechanisms discussed in previous sections. As a downside, the error bound is only in expectation and the data structure does not have strong tail bounds as compared to the ALP mechanism, cf. Lemma 2.14. The data structure is inspired by the Count-Min sketch [CM05].

2.8.1 The data structure

Algorithm 11 shows the *projection algorithm* that returns a differentially private data structure that can be used for estimation. Given a k-sparse vector $x \in \mathbb{R}_+^d$ and a random hash function h mapping from $[d]$ to $[s]$, the algorithm returns a vector $\tilde{y} \in \mathbb{R}^s$ for a parameter s to be chosen later. The idea is that, before noise, each coordinate y_i stores the maximum entry in x for all coordinates of x that are mapped to i by h. We use the

Algorithm 11: Max-Projection

Parameters: $\varepsilon > 0$, and $s \in \mathbb{N}$.
Input : $x \in \mathbb{R}_+^d$. A universal hash function h from domain $[d]$ to $[s]$.
Output : ε-differentially private representation of x.

(1) Initialize $y \in \mathbb{R}^s$ to the all zeroes vector.
(2) For each $i \in [s]$, set $y_i = \max_{h(j)=i} x_j$.
(3) For each $i \in [s]$, sample $\eta_i \sim \text{Laplace}(1/\varepsilon)$
(4) Release h and $\tilde{y} = y + (\eta_1, \ldots, \eta_s)$.

Laplace mechanism on y to release $\tilde{y}$ as the differentially private version of x. Given a coordinate $i \in [d]$, we *estimate* x_i as $\tilde{y}_{h(i)}$.

Lemma 2.33. *Algorithm 11 satisfies ε-differential privacy.*

Proof. Let x and x' such that $||x - x'||_1 \leq 1$ and define y and y' as in Line (2) of Algorithm 11. Each coordinate i such that x_i and x'_i differ can contribute a change of not more than $|x_i - x'_i|$ to $||y - y'||_1$. Thus, $||y - y'||_1 \leq 1$. Adding Laplace noise with scale $1/\varepsilon$ guarantees ε-differential privacy. □

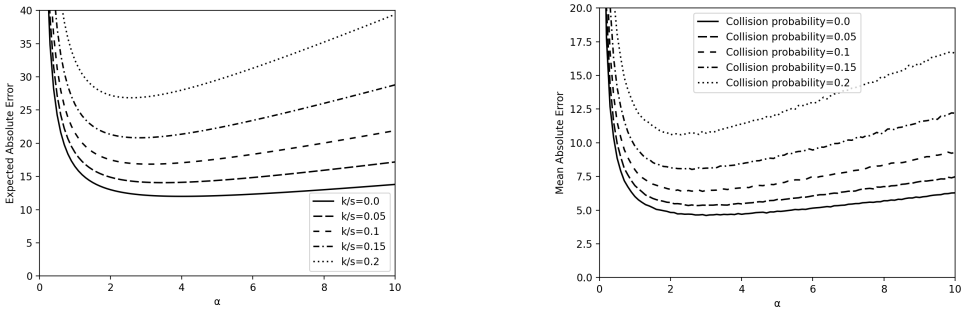
Lemma 2.34. *Given $x \in \mathbb{R}_+^d$ and $\varepsilon > 0$, let $h, \tilde{y}$ be the output of Algorithm 11 with $s = \Omega(\varepsilon||x||_1)$. For each $i \in [d]$, $\mathbb{E}[|x_i - \tilde{y}_{h(i)}|] = O(1/\varepsilon)$. The evaluation of a single coordinate takes time $O(1)$.*

Proof. The running time statement follows because the algorithm evaluates a single hash function value.

Let J be the set of coordinates of the non-zero entries in x. We can use that the hash function is universal to bound the expected difference between x_i and $y_{h(i)}$ by

$$\begin{aligned}\mathbb{E}\left[|x_i - y_{h(i)}|\right] &\leq \sum_{j \in J} \Pr(h(i) = h(j)) \cdot \max(x_j - x_i, 0) \\ &\leq \frac{1}{s} \sum_{j \in J} x_j = \frac{\|x\|_1}{s}.\end{aligned}$$

The expected error from the Laplace noise is $O(1/\varepsilon)$ by Proposition 2.1. For $s = \Omega(\varepsilon||x||_1)$, the expected error is $O(1/\varepsilon)$ for zero entries. Since each non-zero entry x_j with $h(i) = h(j)$ potentially contributes to $y_{h(i)}$, the expected error is only smaller for non-zeroes. □



(a) Upper bound error guarantees

(b) Observed mean per-entry error

Figure 2.3: Theoretical expected per-entry error and experiment results. Note that the y-axes for the plots use different scales.

Corollary 2.2. *Let $\varepsilon > 0$ and $x \in \mathbb{R}_+^d$.*

1. *If $||x||_1 = n$, the data structure described above uses space $s = O(\varepsilon n)$ words, guarantees constant access time, and has expected error $O(1/\varepsilon)$.*

2. *If x is k-sparse, then the data structure uses $O(k \log d)$ words, has constant access time, and has expected error $O(1/\varepsilon)$.*

Proof. If $||x||_1 = n$ is known, we can use $s = \Theta(n\varepsilon)$ as the size of the table. If the vector is k-sparse, we use the thresholding technique (Algorithm 6) with threshold $O(\log(d)/\varepsilon)$ and store the entries below the threshold using Algorithm 11. Restricting on the elements below the threshold, we know that $||x||_1 = O(k \log(d)/\varepsilon)$ and the result follows. □

This approach guarantees constant access time with expected error $O(1/\varepsilon)$, but does not guarantee good tail bounds similar to the ALP mechanism and the Laplace mechanism. Let us focus on the case that we use $s = O(k \log d)$ for a k-sparse vector $x \in \mathbb{R}_+^d$. The probability that one of the $d - k$ zero entries collides with a non-zero is $O(1/\log d)$. All of the non-zero entries can be as large as $O(\log(d)/\varepsilon)$. Thus, we expect $(d-k)/\log d$ zero entries to have error $O(\log(d)/\varepsilon)$.

2.9 Experiments

In this section, we discuss the per-entry error of ALP1-Estimator (Algorithm 3) in practice. Let $\gamma = \frac{\alpha+2}{1+\frac{\alpha k}{s}} - 2$. By Lemma 2.7 and 2.10 the

expected per-entry error of ALP1-Estimator is upper bounded by:

$$\mathbb{E}[|x_i - \tilde{x}_i|] \leq \left(\frac{1}{2} + \frac{4\alpha + 4}{\alpha^2} + \frac{4\gamma + 4}{\gamma^2}\right) \cdot \alpha \ .$$

Figure 2.3a shows the upper bound for varying values of k/s and α. Recall that k/s is a bound on the probability of a hash collision. We see that the effect of hash collisions on the error increases for large values of α, as each bit in the embedding is more significant. We discuss how the upper bound compares to practice next.

Experimental Setup.

We designed experiments to evaluate the effect of the adjustable parameters α and s on the expected per-entry error of ALP1-Estimator. The experiments were performed on artificial data. For our setup, we set parameter $\psi = 5000$ and chose a true value x_i uniformly at random in the interval $[0; \psi]$. We run only on artificial data, as uniform data does not benefit the algorithm, and we can easily simulate worst-case conditions for hash collisions. We simulate running the ALP1-Projection algorithm by computing y_i, simulating hash collisions, and applying randomized response. The probability for hash collisions is fixed in each experiment and the same probability is used for all bits. This simulates worst-case input in which all other non-zero entries have a true value of at least ψ. We increment α by steps of 0.1 in the interval $[0.1, \ldots, 10]$ and the probability of a hash collision by 0.05 in the interval $[0, \ldots, 0.2]$. The probability of 0 serves only as a baseline, as it is not achievable in practice for $k > 1$. The experiment was repeated 10^5 times for every data point.

Figure 2.3b shows plots of the mean absolute error of the experiments. As α is increased, the error drops off at first and slowly climbs. The estimates of y_i are more accurate for large values of α. However, any inaccuracy is more significant, as $\tilde{y}_i$ is scaled back by a larger value. The error from the random rounding step also increases with α. The plots of the upper bound and observed error follow similar trajectories. However, the upper bound is approximately twice as large for most parameters.

Fixed Parameters.

The experiments show how different values of α and s affect the expected per-entry error. However, the parameters also determine constant factors

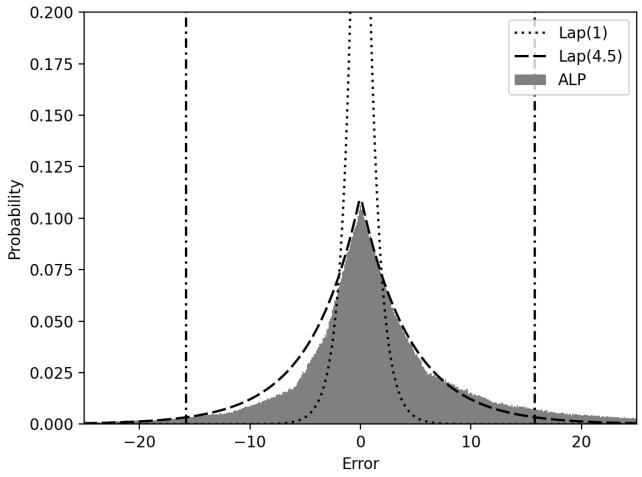


Figure 2.4: Error distribution of Algorithm 3.
($\varepsilon = 1$, $\alpha = 3$, and collision probability $= 0.1$)

for space usage and access time. The space requirements scale linearly in $\frac{s}{\alpha}$ and the access time is inversely proportional to α. As such, the optimal parameter choice depends on the use case due to space, access time, and error trade-offs.

To evaluate the error distribution of the ALP1-Estimator algorithm we fixed the parameters of an experiment. We set $\alpha = 3$ and the hash collision probability to 0.1. We repeated the experiment 10^6 times.

The error distribution is shown in Figure 2.4. The mean absolute error of the experiment is 6.4 and the standard deviation is 11. Plugging in the parameters in Lemma 2.14, with probability at least 0.9 the error is at most

$$|x_i - \tilde{x}_i| < 3 + \frac{6 \log\left(\frac{5}{0.12\sqrt{\pi}}\right)}{\log\left(\frac{25}{19.24}\right)} \approx 75.33 \ .$$

The error of the observed 90th percentile is 15.78, which is shown in Figure 2.4 using vertical lines. Again, this shows that the upper bounds are pessimistic.

For comparison, the plots include the Laplace distribution with scale parameters 1 and 4.5. Note that the Laplace distribution with parameter 1 is optimal for the privacy budget. The standard deviation of the distribution with scale 4.5 is 6.36 and as such the mean absolute error is similar to the ALP mechanism.

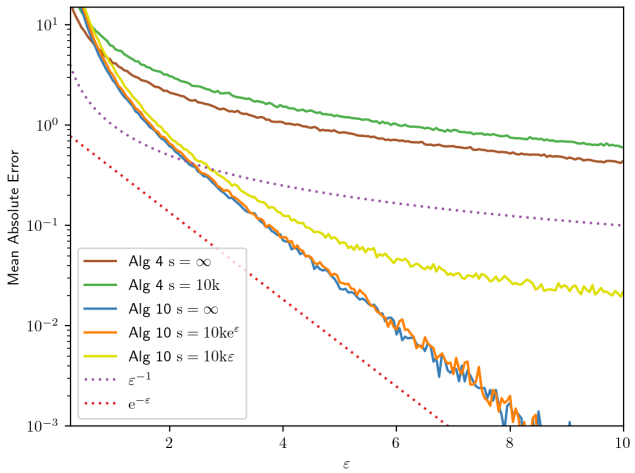


Figure 2.5: Mean per-entry error of Algorithms 4 and 10

The distribution is slightly off-center, and the mean error is 2.33. This is expected due to hash collisions. The effect of hash collisions is also apparent for the largest observed errors. The lowest observed error was -114, while the highest was 274. There is a clear trade-off between space usage and per-entry error. We reran the experiment with hash collision probability 0.01 using the same value for α. The error improved for all the metrics mentioned above. The mean absolute error is 4.8, the standard deviation is 7.8, the mean error is 0.18, the 90th percentile is 11.5, and the largest observed error is 147.

Histograms.

Next we compare the expected per-entry error of ALP-Projection (Algorithm 4) with the variant designed for histograms introduced in Section 2.7. As discussed in the section, Algorithm 10 is more sensitive to ε than Algorithm 4. The experiment in this section compares the effect of changing ε on both algorithms. In particular, the experiments are designed to examine when Algorithm 10 is preferred.

For our setup, we again set $\psi = 5000$. We increment ε by 0.05 in the interval $[0.25, \ldots, 10]$. For algorithm 10 we choose an integer uniformly in $[0, \ldots, \psi]$. We set $\alpha = 3$ based on the previous experiments. Recall that Algorithm 4 introduces error in the scaling step if x_i is not a multiple of

α/ε. For this reason we uniformly select a multiple of α/ε in the interval $[0; \psi]$. The experiment was repeated 10^5 times for each data point.

We ran both simulations with no probability of hash collisions as a baseline. We denote this as $s = \infty$. We ran Algorithm 4 with a probability of 0.1 for hash collisions. We denote this as $s = 10k$ in the legend. For Algorithm 10 we ran the experiment with probabilities of hash collisions of $0.1/e^{\varepsilon}$ and $0.1/\varepsilon$. However, we use 0.1 for $\varepsilon < 1$. The result of the experiment is shown in Figure 2.5. The y-axis has a logarithmic scale and the functions $1/\varepsilon$ and $1/e^{\varepsilon}$ are included for comparison.

The plots show that the preferred algorithm depends on the value of ε as expected. We see that Algorithm 10 is preferred when ε is approximately 0.7 and above. The error of Algorithm 10 is $O(1/e^{\varepsilon})$ and $O(1/\varepsilon)$ for large epsilon as expected if the probabilities of hash collisions is $\Theta(1/e^{\varepsilon})$ and $\Theta(1/\varepsilon)$, respectively. However, it still outperforms Algorithm 4 for large ε by more than an order of magnitude.

Note that Algorithm 4 is much preferred for small ε even though it is not as clearly visible from the Figure. As discussed in Section 2.7, Algorithm 10 uses more space and the expected error scales roughly as $O(1/\varepsilon^2)$ for $\varepsilon < 1$. We ran the experiment with $\varepsilon = 0.05$ and Algorithm 4 outperformed the mean error of Algorithm 10 by an order of magnitude.

2.10 Suggestions to Practitioners

The ALP mechanism introduced in this chapter combines the best of three worlds: It has low error similar to the Laplace mechanism, produces compact representations using asymptotically optimal space, and has an access time that scales only with $O(\log d)$.

In an application that wants to make use of differentially private histograms/vectors, one first has to get an overview of the assumed properties of the data before making a choice on which approach to use. If d is small or the data is assumed to be dense, the Laplace mechanism will offer the best performance. If the data is sparse and the dimension d is large, the analyst must know which error guarantee she wishes to achieve, and which access time is feasible in the setting where the application is deployed. If a larger error is acceptable for "small" entries or access time is crucial, just applying the thresholding technique [KKMN09, CPST12] is the better choice. Otherwise, if small error is paramount or an access time of $O(\log d)$ is sufficient, the ALP mechanism will provide the best solution.

Variants.

We assume in this chapter that k is a known bound on the sparsity of the input data. However, in some applications the value of k itself is private. Here we briefly discuss approaches in such settings. We use the value of k to select the size of the embedding, such that the probability of hash collisions is sufficiently small. When k is not known we can still bound the probability of hash collisions.

If the input is a *histogram* the sparsity differs by at most 1 for neighboring datasets. As such we can use a fraction of the privacy budget to estimate the sparsity. Note that this is not possible for real-valued vectors, as the difference in sparsity can be as large as d for neighboring datasets.

If $\|x\|_1 = n$ is known then we have $\|\hat{x}\|_1 = n\varepsilon$ for the scaled input. We can bound the probability of hash collisions by a constant when the size of the embedding is $\Theta(n\varepsilon)$ bits. We also have $\|x\|_\infty \leq n$ and as such we can set $u = n$ if no better bound is known. If $\|x\|_1$ is unknown we can estimate it using a fraction of the privacy budget. Note that the space differs from the k-sparse setting, and remains $\Theta(n\varepsilon)$ bits when applying the thresholding techniques.

In both cases the estimate affects space and error of our mechanism. We discuss estimating k here but the same principle applies to estimating $\|x\|_1$. Let $\tilde{k}$ be the estimate of k used to set s such that $s = \Theta(\tilde{k})$. It is clear that the space requirement now scales with $\tilde{k}$ instead of k. However, the error guarantees of our mechanism depend on $\frac{k}{s}$. We can bound this by a constant when k is known, but we might introduce large error if $\tilde{k}/k$ is small. This is only likely to happen if the true value of k is small. If a small error is more important than space we can estimate $\tilde{k}$ such that $k \leq \tilde{k}$ with high probability. For example we can set $\tilde{k} = k + \text{Laplace}(1/\varepsilon) + \ln(d/2)/\varepsilon$ such that $k \geq \tilde{k}$ only happens with probability at most $1/d$.

In Section 2.2 we gave a simple reduction from real-valued data to non-negative real-valued data with increased error. It is possible to extend our mechanism to negative values by instead paying an increase in access time. The thresholding techniques in Algorithms 6 and 9 are easily extended to real values by releasing noisy entries with a large absolute value. However, Algorithm 9 only releases entries whose true and noisy values have the same sign to preserve the privacy guarantees. We extend Algorithm 2 by using twice as many columns to store z. The last m columns store non-negative values as before. The first m columns

store the negative values. The bits are set to 1 by default and changed to 0 to encode negative values. Algorithm 3 is unchanged but an offset is used for the returned value.

An implementation of a variant of the ALP mechanism is available as part of the open source project OpenDP (`https://opendp.org/`) in the repository `https://github.com/opendp/opendp`.

2.A Closed-form proof of Lemma 2.9

Here we provide a closed-form expression used in the proof of Lemma 2.9.

In the proof, we will make use of general binomial coefficient([GKP94, Equation 5.1]):

$$\binom{r}{k} = \frac{r(r-1)\dots(r-k+2)(r-k+1)}{k!} \ ,$$

and the binomial theorem ([GKP94, Equation 5.12]):

$$(1+z)^r = \sum_{k=0}^{\infty} \binom{r}{k} (z)^k \ .$$

Starting from an infinite series with $z < 1/4$, we simplify as follows:

$$\begin{aligned}
\sum_{k=0}^{\infty} k\binom{2k}{k}(z)^k &= \sum_{k=1}^{\infty} k\frac{(2k)!}{k!k!}z^k \\
&= \sum_{k=1}^{\infty} k\frac{k(k-\frac{1}{2})(k-1)\dots(\frac{3}{2})1(\frac{1}{2})}{k!k!}2^{2k}z^k \\
&= \sum_{k=1}^{\infty} \frac{(k-\frac{1}{2})(k-\frac{3}{2})\dots(\frac{5}{2})(\frac{3}{2})(\frac{1}{2})}{(k-1)!}(4z)^k \\
&= 2z\sum_{k=1}^{\infty} \frac{(-\frac{3}{2})(-\frac{5}{2})\dots(-k+\frac{3}{2})(-k+\frac{1}{2})}{(k-1)!}(-4z)^{k-1} \\
&= 2z\sum_{k=1}^{\infty} \binom{-\frac{3}{2}}{k-1}(-4z)^{k-1} \\
&= 2z\sum_{k=0}^{\infty} \binom{-\frac{3}{2}}{k}(-4z)^{k} \\
&= \frac{2z}{(1-4z)^{3/2}} \ .
\end{aligned}$$

Finally, let $z = p - p^2$ for $p < 1/2$. This gives us the closed-form expression:

$$\begin{aligned}
\sum_{k=0}^{\infty} k\binom{2k}{k}(p-p^2)^k &= \frac{2(p-p^2)}{(1-4(p-p^2))^{3/2}} \\
&= \frac{2(p-p^2)}{(1-2p)^3} \ .
\end{aligned}$$

Chapter 3

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

Originally published in: Symposium on Principles of Database Systems (PODS 2023) - Chosen for PODS 2023 Distinguished Paper Award

Joint work with: Jakub Tětek

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

3.1 Introduction

Computing the histogram of a dataset is one of the most fundamental tasks in data analysis. This problem has been investigated thoroughly in the differentially private setting [DMNS06, GRS12, GKOV15, CPST12, KKMN09, BV19, ALP22]. These algorithms start by computing the histogram exactly and they then add noise to ensure privacy. However, in practice, the amount of data is often so large that computing the histogram exactly would be impractical. This is, for example, the case when computing the histogram of high-volume streams such as when monitoring computer networks, online users, financial markets, and similar. In that case, we need an efficient streaming algorithm. Since the streaming algorithm would only compute the histogram approximately, the above-mentioned approach that first computes the exact histogram is infeasible. In practice, non-private approximate histograms are often computed using the Misra-Gries (MG) sketch [MG82]. The MG sketch of size k returns at most k items and their approximate frequencies $\hat{f}$ such that $\hat{f}(x) \in [f(x) - n/(k+1), f(x)]$ for all elements x where $f(x)$ is the true frequency and n is the length of the stream. This error is known to be optimal [BKMT03]. In this work, we develop a way of releasing a MG sketch in a differentially private way while adding only a small amount of noise.[1] This allows us to efficiently and accurately compute approximate histograms in the streaming setting while not violating users' privacy. This can then be used to solve the heavy hitters problem in a differentially private way. Our result improves upon the work of Chan, Li, Shi, and Xu [CLSX12] who also show a way of privately releasing the MG sketch, but who need a greater amount of noise; we discuss this below.

In general, the issue with making approximation algorithms differentially private is that although we may be approximating a function with low global sensitivity, the algorithm itself (or rather the function it implements) may have a much larger global sensitivity. This increases the amount of noise required to achieve privacy using standard techniques. We get around this issue by exploiting the structure of the difference between the MG sketches for neighboring inputs. This allows us to prove that the following simple mechanism ensures (ε, δ)-differential privacy: (1) We compute the Misra-Gries sketch, (2) we add to each

[1] See `https://github.com/JakubTetek/Differentially-Private-Misra-Gries` for a sample implementation.

counter independently noise distributed as Laplace$(1/\varepsilon)$, (3) we add to all counters the same value, also distributed as Laplace$(1/\varepsilon)$, (4) we remove all counters smaller than $1 + 2\ln(3/\delta)/\varepsilon$. Specifically, we show that this algorithm satisfies the following guarantees:

THEOREM 3.1 (SIMPLIFIED). *The above algorithm is (ε, δ)-differentially private, uses $2k$ words of space, and returns a frequency oracle $\hat{f}$ with maximum error of $n/(k+1) + O(\log(1/\delta)/\varepsilon)$ with high probability for δ being sufficiently small.*

A construction for a differentially private Misra-Gries sketch has been given before by Chan et al. [CLSX12]. However, the more accurate they want their sketch to be (and the bigger it is), their approach has to add *more* noise. The reason is that they directly rely on the global ℓ_1-sensitivity of the sketch. Specifically, if the sketch has size k (and thus error $n/(k+1)$ on a stream of n elements), its global sensitivity is k, and they thus have to add noise of magnitude k/ε. Their mechanism ends up with an error of $O\left(k\log(d)/\varepsilon\right)$ for ε-differential privacy with d being the universe size. This can be easily improved to $O\left(k\log(1/\delta)/\varepsilon\right)$ for (ε, δ)-differential privacy with a thresholding technique similar to what we do in step (4) of our algorithm above. This also means that they cannot get more accurate than error $\Theta\left(\sqrt{n\log(1/\delta)/\varepsilon}\right)$, no matter what value of k one chooses. We achieve that the biggest error, as compared to the values from the MG sketch, among all elements is $O(\log(1/\delta)/\varepsilon)$ assuming δ is sufficiently small (we show more detailed bounds including the mean squared errors in Theorem 3.1). This is the same as the best private solution that starts with an exact histogram [KKMN09]. In fact, for any mechanism that outputs at most k heavy hitters there exists input with error at least $n/(k+1)$ in the streaming setting [BKMT03] and input with error at least $O(\log(\min(d, 1/\delta))/\varepsilon)$ [BV19] under differential privacy. In Section 3.6 we discuss how to achieve ε-differential privacy with error $n/(k+1) + O(\log(d)/\varepsilon)$. Therefore the error of our mechanisms is asymptotically optimal for approximate and pure differential privacy, respectively. The techniques used in Section 3.6 could also be used to get approximate differential privacy, but the resulting sketch would not have strong competitiveness guarantees with respect to the non-private Misra-Gries sketch, unlike the sketch that we give in Section 3.5.

Chan et al. [CLSX12] use their differentially private Misra-Gries sketch as a subroutine for continual observation and combine sketches with an untrusted aggregator. Those settings are not a focus of our work but our

algorithm can replace theirs as the subroutine, leading to better results also for those settings. However, the error from noise still increases linearly in the number of merges when the aggregator is untrusted. As a side note, we show that in the case of a trusted aggregator, the approach of [CLSX12] can handle merge operations without increasing error. While that approach adds significantly more noise than ours if we do not merge, it can with this improvement perform better when the number of merges is very large (at least proportional to the sketch size).

Another approach that can be used is to use a randomized frequency oracle to recover heavy hitters. However, it seems hard to do this with the optimal error size. In its most basic form [GGK+19, Appendix D], this approach needs noise of magnitude $\Theta(\log(d)/\varepsilon)$, even if we have a sketch with sensitivity 1 (the approach increases the sensitivity to $\log(d)$, necessitating the higher noise magnitude), leading to maximum error at least $\Omega(\log(k)\log(d)/\varepsilon)$. Bassily, Nissim, Stemmer, and Guha Thakurta [BNSGT17] show a more involved approach which reduces the maximum error coming from the noise to $\Theta((\log(k)+\log(d))/\varepsilon)$, but at the cost of increasing the error coming from the sketch by a factor of $\log(d)$. This means that even if we had a sketch with error $\Theta(n/k)$ and sensitivity 1, neither of these two approaches would lead to optimal guarantees, unlike the algorithm we give in this chapter.

Relation to [BK21]. Essentially the same result as Theorem 3.1 has been claimed by Böhler and Kerschbaum [BK21]. However, their approach ignores the discrepancy between the global sensitivity of a function we are approximating and that of the function the algorithm actually computes. Their mechanism adds noise scaled to the sensitivity of the exact histogram which is 1 when a user contributes a single element to the stream. But as shown by Chan et al. [CLSX12] the sensitivity of the Misra-Gries sketch scales linearly with the number of counters in the sketch. The algorithm from [BK21] thus does not achieve the claimed privacy parameters. Moreover, it seems unlikely this could be easily fixed – not without doing something along the lines of what we do in this chapter.

3.2 Technical overview

Misra-Gries sketch. Since our approach depends on the properties of the MG sketch, we describe it here. Readers familiar with the MG sketch

may wish to skip this paragraph. We describe the standard version; in Section 3.5 we use a slight modification, but we do not need that here.

Suppose we receive a sequence of elements from some universe. At any time, we will be storing at most k of these elements. Each stored item has an associated counter, other elements have implicitly their counter equal to 0. When we process an element, we do one of the following three updates: (1) if the element is being stored, increment its counter by 1, (2) if it is not being stored and the number of stored items is $< k$, store the element and set its counter to 1, (3) otherwise decrement all k counters by 1 and remove those that reach 0. The exact guarantees on the output will not be important now, and we will discuss them in Section 3.5.

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

Consider two neighboring data streams $S = (S_1, \cdots, S_n)$ and $S' = (S_1, \cdots, S_{i-1}, S_{i+1}, \cdots, S_n)$ for some $i \in [n]$. At step $i-1$, the state of the MG sketch on both inputs is exactly the same. MG_S then receives the item S_i while $MG_{S'}$ does not. This either increments one of the counters of MG_S (possibly by adding an element and raising its counter from 0 to 1) or decrements all its counters. In ℓ_1 distance, the vector of the counters thus changes by at most k. One can show by induction that this will stay this way: at any point in time, $\|MG_S - MG_{S'}\|_1 \leq k$. By a standard global sensitivity argument, one can achieve pure DP by adding noise of magnitude k/ε to each count. This is the approach used in [CLSX12]. Similarly, we could achieve (ε, δ)-DP by using the Gaussian mechanism [DR14] with noise magnitude proportional to the ℓ_2-sensitivity, which is $\sup_{S,S'} \|MG_S - MG_{S'}\|_2 \leq \sqrt{k}$. We want to instead achieve noise with magnitude $O(1/\varepsilon)$ at each count. To this end, we need to exploit the structure of $MG_S - MG_{S'}$.

What we just described requires that we add the noise to the counts of all items in the universe, also to those that are not stored in the sketch. This results in the maximum error of all frequencies depending on the universe's size, which we do not want. However, it is known that this can be easily solved under (ε, δ)-differential privacy by only adding noise to the stored items and then removing values smaller than an appropriately chosen threshold [KKMN09]. This may introduce additional error – for this reason, we end up with error $O(\log(1/\delta)/\varepsilon)$. As this is a somewhat standard technique, we ignore this in this section, we assume that the

sketches MG_S and $MG_{S'}$ store the same set of elements; the thresholding allows us to remove this assumption, while allowing us to add noise only to the stored items, at the cost of only getting approximate DP.

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

How do we use the structure of $MG_S - MG_{S'}$ to our advantage? We add noise twice. First, we independently add to each counter noise distributed as Laplace$(1/\varepsilon)$. Second, we add to all counters the same value, also distributed as Laplace$(1/\varepsilon)$. That is, we release $MG_S + \text{Laplace}(1/\varepsilon)^{\otimes k} + \text{Laplace}(1/\varepsilon)\mathbf{1}^k$ [3]. Intuitively speaking, the first noise hides the difference between S and S' in case (1) and the second noise hides the difference in case (2). We now sketch why this is so for worse constants: $2/\varepsilon$ in place of $1/\varepsilon$. When proving this formally, we use a more technical proof which leads to the better constant.

We now sketch why this is differentially private. Let m_S be the mean of the counters in MG_S for S being an input stream. We may represent MG_S as $(MG_S - m_S\mathbf{1}, m_S)$ (note that there is a bijection between this representation and the original sketch). We now argue that the ℓ_1-sensitivity of this representation is < 2 (treating the representation as a $(k+1)$-dimensional vector for the sake of computing the ℓ_1 distances). Consider the first case. In that case, the averages $m_S, m_{S'}$ differ by $1/k$. As such, $MG_S - m_S\mathbf{1}^k$ and $MG_{S'} - m_{S'}\mathbf{1}^k$ differ by $1/k$ at $k-1$ coordinates and by $1 - 1/k$ at one coordinate. The overall ℓ_1 change of the representation is thus

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

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

[2] We use $\mathbf{1}^k$ the denote the dimension k vector of all ones.

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

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

3.3 Preliminaries

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

Differential privacy. Differential privacy is a rigorous definition for describing the privacy loss of a randomized mechanism introduced by Dwork, McSherry, Nissim, and Smith [DMNS06]. Intuitively, differential privacy protects privacy by restricting how much the output distribution can change when replacing the input from one individual. This is captured by the definition of neighboring datasets. We use the add-remove neighborhood definition for differential privacy.

Definition 3.1 (Neighboring Streams)**.** Let S be a stream of length n. Two streams S and S' are neighboring denoted $S \sim S'$ if there exists an i such that $S = (S'_1, \ldots, S'_{i-1}, S'_{i+1}, \ldots, S'_{n+1})$ or $S' = (S_1, \ldots, S_{i-1}, S_{i+1}, \ldots, S_n)$.

Definition 3.2 (Differential Privacy [DR14])**.** A randomized mechanism $\mathcal{M} : \mathcal{U}^{\mathbb{N}} \to \mathcal{R}$ satisfies (ε, δ)-differential privacy if and only if for all pairs of neighboring streams $S \sim S'$ and all measurable sets of outputs $Z \subseteq \mathcal{R}$ it holds that

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

Samples from a Laplace distribution are used in many differentially private algorithms, most notably the Laplace mechanism [DMNS06]. We

write Laplace(b) to denote a random variable with a Laplace distribution with scale b centered around 0. We sometimes abuse notation and write Laplace(b) to denote the value of a random variable drawn from the distribution. Our mechanism also works with other noise distributions. We briefly discuss this in Section 3.5.2.

Definition 3.3 (Laplace distribution)**.** The probability density and cumulative distribution functions of the Laplace distribution centered around 0 with scale parameter b are $f_b(x) = \frac{1}{2b}e^{-|x|/b}$, and $\Pr[\text{Laplace}(b) \leq x] = \frac{1}{2}e^{x/b}$ if $x < 0$ and $1 - \frac{1}{2}e^{-x/b}$ for $x \geq 0$.

3.4 Related work

Chan et al. [CLSX12] show that the global ℓ_1-sensitivity of a Misra-Gries sketch is $\Delta_1 = k$. (They actually show that the sensitivity is $k + 1$ but they use a different definition of neighboring datasets that assumes n is known. Applying their techniques under our definition yields sensitivity k.) They achieve privacy by adding noise with scale k/ε to all elements in the universe and keep the top-k noisy counts. This gives an expected maximum error of $O(k \log(d)/\varepsilon)$ with ε-DP for d being the universe size. They use the algorithm as a subroutine for continual observation and merge sketches with an untrusted aggregator. Those settings are not a focus of our work but our algorithm can replace theirs as the subroutine.

Böhler and Kerschbaum [BK21] worked on differentially private heavy hitters with no trusted server by using secure multi-party computation. One of their algorithms adds noise to the counters of a Misra-Gries sketch. They avoid adding noise to all elements in the universe by removing noisy counts below a threshold which adds an error of $O(\log(1/\delta)/\varepsilon)$. This is a useful technique for hiding differences in keys between neighboring sketches that removes the dependency on d in the error. Unfortunately, as stated in the introduction their mechanism uses the wrong sensitivity. The sensitivity of the sketch is k. If the magnitude of noise and the threshold are increased accordingly the error of their approach is $O(k \log(k/\delta)/\varepsilon)$.

If we ignore the memory restriction in the streaming setting, the problem is the same as the top-k problem [MMNW11, DR19, CWGM20, QSZ21]. The problem we solve can also be seen as a generalization of the sparse histogram problem. This has been investigated in [CPST12, KKMN09, BV19, ALP22]. Notably, Balcer and Vadhan [BV19] provides a lower bound showing that for any (ε, δ)-differentially private mechanism

that outputs at most k counters, there exists input such that the expected error for some elements is $\Omega(\min(\log(d/k)/\varepsilon, \log(1/\delta)/\varepsilon, n))$ (assuming $\varepsilon^2 > \delta$). The noise that we add in fact matches the second branch of the min over all elements.

A closely related problem is that of implementing frequency oracles in the streaming setting under differential privacy. This has been studied in e.g. [ZQR+22, PT22, GGK+19]. These approaches do not directly return the heavy hitters. The simplest approach for finding the heavy hitters is to iterate over the universe which might be infeasible. However, there are constructions for finding heavy hitters with frequency oracles more efficiently (see Bassily et al. [BNSGT17]). However, as we discussed in the introduction, the approach of [BNSGT17] leads to worse maximum error than what we get unless the sketch size is very large and the universe size is small.

The heavy hitters problem has also received a lot of attention in local differential privacy, starting with the paper introducing the RAPPOR mechanism [EPK14] and continuing with [QYY+16, ZZC+22, WLJ19, BNS19a, WW22, BNSGT17]. This problem is practically relevant, it is used for example by Apple to find commonly used emojis [App]. The problem has also been recently investigated when using cryptographic primitives [ZKM+20].

[BGMZ22, Tět22] have recently given general frameworks for designing differentially private approximation algorithms; however, if used naively, they are not very efficient for releasing multiple values (not more efficient than using composition) and they are thus not suitable for the heavy hitters problem.

3.5 Differentially Private Misra-Gries

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

We first present our variant of the non-private Misra-Gries sketch in Algorithm 12 and later show how we add noise to achieve (ε, δ)-differential privacy. The algorithm we use differs slightly from most implementations of MG in that we do not remove elements that have

weight 0 until we need to re-use the counter. This will allow us to achieve privacy with slightly lower error.

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

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

In case (3) we always remove the smallest element with a count of zero. This allows us to limit the number of keys that differ between sketches for neighboring streams as shown in Lemma 3.1. The choice of removing the minimum element is arbitrary but the order of removal must be independent of the stream so that it is consistent between neighboring datasets. The limit on differing keys allows us to obtain a slightly lower error for our private mechanism. However, it is still possible to apply our mechanism with standard implementations of MG. We discuss this in Section 3.5.1.

Algorithm 12: Misra-Gries (MG)

Input: Positive integer k and stream $S \in \mathcal{U}^{\mathbb{N}}$

(1) $T \leftarrow \{d+1, \ldots, d+k\}$ // Start with k dummy counters
(2) $c_i \leftarrow 0$ for all $i \in T$
(3) **foreach** $x \in S$ **do**
(4) | **if** $x \in T$ **then** // Branch 1
(5) | | $c_x \leftarrow c_x + 1$
(6) | **else if** $c_i \geq 1$ *for all* $i \in T$ **then** // Branch 2
(7) | | $c_i \leftarrow c_i - 1$ for all $i \in T$
(8) | **else** // Branch 3
(9) | | Let $y \in T$ be the smallest key satisfying $c_y = 0$
(10) | | $T \leftarrow (T \cup \{x\}) \setminus \{y\}$
(11) | | $c_x \leftarrow 1$
(12) **return** T, c

The same guarantees about correctness hold for our version of the MG sketch, as for the original version. This can be easily shown, as

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

Fact 3.1 (Bose et al. [BKMT03]). Let $\hat{f}(x)$ be the frequency estimates given by an MG sketch of size k for n being the input size. Then for all $x \in \mathcal{U}$, it holds $\hat{f}(x) \in [f(x) - n/(k+1), f(x)]$, where $f(x)$ is the true frequency of x.

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

We now analyze the value of $MG_S - MG_{S'}$ for S, S' being neighboring inputs. We will then use this in order to prove privacy. As mentioned in Section 3.4, Chan et al. [CLSX12] showed that the ℓ_1-sensitivy for Misra-Gries sketches is k. They show that this holds after processing the element that differs for neighboring streams and use induction to show that it holds for the remaining stream. Our analysis follows a similar structure. We expand on their result by showing that the sets of stored elements for neighboring inputs differ by at most two elements when using our variant of Misra-Gries. We then show how all this can be used to get differential privacy with only a small amount of noise.

Lemma 3.1. *Let $T, c \leftarrow \mathrm{MG}(k, S)$ and $T', c' \leftarrow \mathrm{MG}(k, S')$ be the outputs of Algorithm 12 on a pair of neigboring streams S, S' such that S' is obtained by removing an element from S. Then $|T \cap T'| \geq k - 2$ and all counters not in the intersection have a value of at most 1. Moreover, it holds that either (1) $c_i = c'_i - 1$ for all $i \in T'$ and $c_j = 0$ for all $j \notin T'$ or (2) there exists an $i \in T$ such that $c_i = c'_i + 1$ and $c_j = c'_j$ for all $j \neq i$.*

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

(S1) $T = T'$ and $c_i = c'_i - 1$ for all $i \in T$.

(S2) There exist $x_1, x_2 \in \mathcal{U}$ such that $w = \{x_1\}$ and $w' = \{x_2\}$, $c_{x_1} = 0$, $c'_{x_2} = 1$ and $c_i = c'_i - 1$ for all $i \in T \cap T'$.

(S3) $T = T'$ and there exists $x_1 \in \mathcal{U}$ such that $c_{x_1} = c'_{x_1} + 1$ and $c_i = c'_i$ for all $i \in T \setminus \{x_1\}$.

(S4) There exist $x_1, x_2 \in \mathcal{U}$ such that $w = \{x_1\}$ and $w' = \{x_2\}$, $c_{x_1} = 1$, $c'_{x_2} = 0$ and $c_i = c'_i$ for all $i \in T \cap T'$.

(S5) There exist $x_1, x_2, x_3 \in \mathcal{U}$ such that $c_{x_1} = c'_{x_1} + 1$, $w = \{x_2\}$, $w' = \{x_3\}$, $c_{x_2} = 0$, $c'_{x_3} = 0$ and $c_i = c'_i$ for all $i \in T \cap T' \setminus \{x_1\}$.

(S6) There exist $x_1, x_2, x_3, x_4 \in \mathcal{U}$ such that $w = \{x_1, x_2\}$ and $w' = \{x_3, x_4\}$, $c_{x_1} = 1$, $c_{x_2} = c'_{x_3} = c'_{x_4} = 0$, $c_i = c'_i$ for all $i \in T \cap T'$ and $x_4 = c'_0$.

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

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

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

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

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

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

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

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

removed we remain in state S3. Otherwise, if x_1 is replaced in sketch 2 we transition to state S4.

State S4: Sketch 2 contains a counter with a value of zero in states S4, S5, and S6. Therefore Brach 2 is never executed on sketch 2 in the rest of the proof. If Branch 1 is executed for both sketches we stay in the same state as always but if $x = x_1$ Branch 1 is executed for sketch 1 and Branch 3 is executed for sketch 2. If $c'_0 = x_2$ then $T = T'$ after processing x and we transition to state S3. If $c'_0 \neq x_2$ another element is removed from sketch 2 which must also have a count of zero in sketch 1 and we go to state S5.

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

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

State S5: Since by definition both sketches contain counters with a value of zero, Branch 2 is never executed while in this state. If $x \in T \cap T'$ we remain in the same state as always. We have to consider the cases where $x = x_2$, $x = x_3$, and $x \notin T \cup T'$. The resulting state depends on the elements that are replaced in the sketch. For $x = x_2$ we transition to state S3 if $c'_0 = x_3$ and remain in state S5 otherwise. The same argument shows that we transition to state S3 or S5 based on c_0 if $x = x_3$. When $x \notin T \cup T'$ we execute Branch 3 on both sketches. We transition to state S3 only if $c_0 = x_2$ and $c'_0 = x_3$ since otherwise both sketches still have a zero counter that is not stored in the other sketch and we stay in state S5.

State S6: Similar to state S5, Branch 2 is never executed from this state. Here we have to consider the five cases where $x \in T \cap T'$, $x = x_1$, $x = x_2$, $x \in w'$, and $x \notin T \cup T'$. We know that x_4 is replaced whenever $x \notin T'$. If $x \in T \cap T'$ we execute Branch 1 on both sketches and remain in state S6. If $x = x_1$ we transition to state S5 and for $x = x_2$ we transition to state S4. When $x \in w'$ there are two possibilities. We always have $c_x = c'_x$ after updating. If $c_0 = x_2$ the sketches will share $k - 1$ keys and transition to

state S4. If $c_0 \neq x_2$ then another element that has a count of zero in both sketches is replaced in sketch 1. We know that either this element or the remaining zero-valued element of w' must be the smallest zero-valued element in sketch 2. Therefore we remain in state S6.

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

Next, we consider how to add noise to release the Misra-Gries sketch under differential privacy. Recall that Chan et al. [CLSX12] achieves privacy by adding noise to each counter which scales with k. We avoid this by utilizing the structure of sketches for neighboring streams shown in Lemma 3.1. We sample noise from Laplace$(1/\varepsilon)$ independently for each counter, but we also sample one more random variable from the same distribution which is added to all counters. Small values are then discarded using a threshold to hide differences in the sets of stored keys between neighboring inputs. This is similar to the technique used by e.g. [KKMN09]. The algorithm takes the output from MG as input. We sometimes write PMG(k, S) as a shorthand for PMG(MG(k, S)).

Algorithm 13: Private Misra-Gries (PMG)

Parameters: $\varepsilon, \delta > 0$
Input : Output from Algorithm 12: $T, c \leftarrow \text{MG}(k, S)$

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

We prove the privacy guarantees in three steps. First, we show that changing either a single counter or all counters by 1 does not change the output distribution significantly (Corollary 3.1). This assumes that, for neighboring inputs, the set of stored elements is exactly the same. By Lemma 3.1, we have that the difference between the sets of stored

keys is small and the corresponding counters are ≤ 1. Relying on the thresholding, we bound the probability of outputting one of these keys (Lemma 3.3). Finally, we combine these two lemmas to show that the privacy guarantees hold for all cases (we do this in Lemma 3.4).

Lemma 3.2. *Let us have $x, x' \in \mathbb{R}^k$ such that one of the following three cases holds*

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

Then we have for any measurable set Z that

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

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

$$\begin{aligned}\Pr[x + \text{Laplace}(1/\varepsilon)^{\otimes k} + \text{Laplace}(1/\varepsilon)\mathbf{1}^k \in Z] = \\ \mathbb{E}_{N \sim \text{Laplace}(1/\varepsilon)}\big[\Pr[\text{Laplace}(1/\varepsilon)^{\otimes k} \in Z - x - N\mathbf{1}^k | N]\big] \leq \\ e^{\varepsilon}\mathbb{E}_{N \sim \text{Laplace}(1/\varepsilon)}\big[\Pr[\text{Laplace}(1/\varepsilon)^{\otimes k} \in Z - x' - N\mathbf{1}^k) | N]\big] = \\ e^{\varepsilon} \Pr[x' + \text{Laplace}(1/\varepsilon)^{\otimes k} + \text{Laplace}(1/\varepsilon)\mathbf{1}^k \in Z]\end{aligned}$$

where to prove the inequality, we used that for any measurable set A, it holds $\Pr[\text{Laplace}(1/\varepsilon)^{\otimes k} \in A] \leq e^{\varepsilon} \Pr[\text{Laplace}(1/\varepsilon)^{\otimes k} \in A - \phi]$ for any $\phi \in \mathbb{R}^k$ with $\|\phi\|_1 \leq 1$ (see [DMNS06]). Specifically, we have set $A = Z - x - N\mathbf{1}^k$ and $\phi = x - x'$ such that $\|\phi\|_1 = 1$.

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

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

This allows us to argue like above:

$$\begin{aligned}
&\Pr[x + \text{Laplace}(1/\varepsilon)^{\otimes k} + \text{Laplace}(1/\varepsilon)\mathbf{1}^k \in Z] = \\
&\mathbb{E}_{N_1 \sim \text{Laplace}(1/\varepsilon)^{\otimes k}}\big[\Pr[x + N_1 + \text{Laplace}(1/\varepsilon)\mathbf{1}^k \in Z | N_1]\big] \leq \\
&e^{\varepsilon}\mathbb{E}_{N_1 \sim \text{Laplace}(1/\varepsilon)^{\otimes k}}\big[\Pr[x' + N_1 + \text{Laplace}(1/\varepsilon)\mathbf{1}^k \in Z | N_1]\big] = \\
&e^{\varepsilon}\Pr[x' + \text{Laplace}(1/\varepsilon)^{\otimes k} + \text{Laplace}(1/\varepsilon)\mathbf{1}^k \in Z]
\end{aligned}$$

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

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

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

Corollary 3.1. *Let T, c and T', c' be two sketches such that $T = T'$ and one of following holds:*

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

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

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

Proof. Consider first a modified algorithm PMG′ that does not perform the thresholding: that is, if we remove the condition on line 5. It can be easily seen that PMG′ only takes the vector c and releases $c + \text{Laplace}(1/\varepsilon)^{\otimes k} + \text{Laplace}(1/\varepsilon)\mathbf{1}^k$. We have just shown in Lemma 3.2 that this means that for any measurable Z',

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

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

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

as we wanted to show. □

Lemma 3.3. *Let T, c and T', c' be two sketches of size k and let $\hat{T} = T \cap T'$. If we have that $|\hat{T}| \geq k - 2$, $c_i = c'_i$ for all $i \in \hat{T}$, and for all $x \notin \hat{T}$, it holds $c_x, c'_x \leq 1$. Then for any measurable set Z, it holds*

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

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

As such, the Lemma holds if $\Pr[\tilde{T} \not\subseteq \hat{T}] + \Pr[\tilde{T}' \not\subseteq \hat{T}] \leq \delta$. That is, it suffices to prove that with probability at most δ any noisy count for elements not in $\hat{T}$ is at least $1 + 2\ln(3/\delta)/\varepsilon$. The noisy count for such a key can only exceed the threshold if one of the two noise samples added to the key is at least $\ln(3/\delta)/\varepsilon$. From Definition 3.3 we have $\Pr[\text{Laplace}(1/\varepsilon) \geq \ln(3/\delta)/\varepsilon] = \delta/6$. There are at most 4 keys not in $\hat{T}$ which are in $T \cup T'$ and therefore at most 6 noise samples affect the probability of outputting such a key (the 4 individual Laplace noises and the 2 global Laplace noises, one for each sketch). By a union bound the probability that any of these samples exceeds $\ln(3/\delta)/\varepsilon$ is at most δ. □

We are now ready to prove the privacy guarantee of Algorithm 13.

Lemma 3.4. *Algorithm 13 is (ε, δ)-differentially private for any k.*

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

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

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

We prove the guarantee above using an intermediate sketch that "lies between" T, c and T', c'. The sketch has support T' and we denote the counters as $\hat{c}$. By Lemma 3.1, we know that $|T \cap T'| \geq k - 2$ and all counters in c and c not in $T \cap T'$ are at most 1. We will now come up with some conditions on $\hat{c}$ such that if these conditions hold, the lemma follows. We will then prove the existence of such $\hat{c}$ below. Assume that $\hat{c}_i = c_i$ for all $i \in T \cap T'$ and $\hat{c}_j \leq 1$ for all $j \notin T' \setminus T$. Lemma 3.3 then tells us that

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

Assume also that one of the required cases for Corollary 3.1 holds between $\hat{c}$ and c'. We have

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

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

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

It remains to prove the existence of $\hat{c}$ such that $\hat{c}_i = c_i$ for all $i \in T \cap T'$ and $\hat{c}_j \leq 1$ for all $j \in T' \setminus T$ and such that one of the conditions $(1) - (3)$ of Corollary 3.1 holds between $\hat{c}$ and c'. We first consider neighboring streams where S' is obtained by removing an element from S. From Lemma 3.1 we have two cases to consider. If $c_i = c'_i - 1$ for all $i \in T'$ we simply set $\hat{c} = c$. Recall that we implicitly have $c_i = 0$ for $i \notin T$. Therefore the sketch satisfies the two conditions above since $\hat{c}_i = c_i$ for all $i \in \mathcal{U}$ and condition (2) of Corollary 3.1 holds. In the other case where

$c_i = c'_i + 1$ for exactly one $i \in T$ there are two possibilities. If $i \in T'$ we again set $\hat{c} = c$. When $i \notin T'$ there must exist at least one element $j \in T'$ such that $c'_j = 0$ and $j \notin T$. We set $\hat{c}_j = 1$ and $\hat{c}_i = c'_i$ for all $i \neq j$. In both cases $\hat{c}_i = c_i$ for all $i \in T \cap T'$ and $\hat{c}_j$ is at most one for $j \notin T$. There is exactly one element with a higher count in $\hat{c}$ than c' which means that condition (1) of Corollary 3.1 holds.

If S is obtained by removing an element from S' the cases from Lemma 3.1 are flipped. If $c_i - 1 = c'_i$ for all $i \in T$ and $c'_j = 0$ for $j \notin T$ we set $\hat{c}_i = c_i$ if $i \in T$ and $\hat{c}_i = 1$ otherwise. It clearly holds that $\hat{c}_i = c_i$ for all $i \in T \cap T'$ and $\hat{c}_j \leq 1$ for all $j \notin T$. Since $\hat{c}_i = c'_i + 1$ for all $i \in T'$ condition (3) of Corollary 3.1 holds. Finally, if $c_i + 1 = c'_i$ for exactly one $i \in T'$ we simply set $\hat{c} = c$. $\hat{c}_i = c_i$ clearly holds for all $i \in T \cap T'$, $\hat{c}_j = 0$ for all $j \notin T$, and condition (1) of Corollary 3.1 holds between $\hat{c}$ and c'. □

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

Lemma 3.5. *Let $\tilde{T}, \tilde{c} \leftarrow \mathrm{PMG}(T, c)$ denote the output of Algorithm 13 for any sketch T, c with $|T| = k$. Then with probability at least $1 - \beta$ we have*

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

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

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

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

Since $k+1$ samples are drawn we know by a union bound that the absolute value of all samples is bounded by $\ln((k+1)/\beta)/\varepsilon$ with probability at least $1 - \beta$. As such the absolute error from the Laplace samples is at

most $2\ln((k+1)/\beta)/\varepsilon$ for all $x \in T$ since two samples are added to each count. Removing noisy counts below the threshold potentially adds an additional error of at most $1 + 2\ln(3/\delta)/\varepsilon$. It is easy to see that $\tilde{c}_x = 0$ for all $x \notin T$ since the algorithm never outputs any such elements. □

Theorem 3.1. PMG(k, S) *satisfies* (ε, δ)*-differential privacy. Let* $f(x)$ *denote the frequency of* $x \in \mathcal{U}$ *in* S *and let* $\hat{f}(x)$ *denote the estimated frequency of* x *from the output of* PMG(k, S)*. For any element* x *with* $f(x) = 0$ *we have* $\hat{f}(x) = 0$ *and with probability at least* $1 - \beta$ *we have for all* $x \in \mathcal{U}$ *that*

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

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

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

If $f(x) = 0$ we know that $x \notin T$ where T is the keyset after running Algorithm 12. Since Algorithm 13 outputs a subset of T we have $\hat{f}(x) = 0$. The first part of the Theorem follows directly from Fact 3.1 and Lemmas 3.4 and 3.5.

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

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

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

3.5.1 Privatizing standard versions of Misra-Gries

The privacy of our mechanism as presented in Algorithm 13 relies on our variant of the Misra-Gries algorithm. Our sketch can contain elements

with a count of zero. However, elements with a count of zero are removed in the standard version of the sketch. As such, sketches for neighboring datasets can differ for up to k keys if one sketch stores k elements with a count of 1 and the other sketch is empty. It is easy to change Algorithm 13 to handle these implementations. We simply increase the threshold to $1 + 2\ln\left(\frac{k+1}{2\delta}\right)/\varepsilon$ since the probability of outputting any of the k elements with a count of 1 is bounded by δ.

3.5.2 Tips for practitioners

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

The output of the Misra-Gries algorithm is an associative array. In Algorithm 13 we add appropriate noise such that the associative array can be released under differential privacy. However, for some implementations of associative arrays such as hash tables the order in which keys are added affects the data structure. Using such an implementation naively violates differential privacy but it is easily solved either by outputting a random permutation of the key-value pairs or using a fixed order e.g. sorted by key.

We present our mechanism with noise sampled from the Laplace distribution. However, the distribution is defined for real numbers which cannot be represented on a finite computer. This is a known challenge and precision-based attacks still exist on popular implementations [HDH+22]. Since the output of MG is discrete the distribution can be replaced by the Geometric mechanism [GRS12] or one of the alternatives introduced in [BV19]. Our mechanism would still satisfy differential privacy but it might be necessary to change the threshold in Algorithm 13 slightly to ensure that Lemma 3.3 still holds. Our proof of Lemma 3.3 works for the Geometric mechanism from [GRS12] when increasing the threshold to $1 + 2\lceil\ln(6e^{\varepsilon}/((e^{\varepsilon}+1)\delta))/\varepsilon\rceil$.

Lastly, it is worth noting that the analysis for Lemma 3.3 is not tight. We bound the probability of outputting a small key by bounding the value of all relevant samples by $\ln(3/\delta)/\varepsilon$ which is sufficient to guarantee that the sum of any two samples does not exceed $2\ln(3/\delta)/\varepsilon$. This simplifies the proof and presentation significantly however one sample could exceed $\ln(3/\delta)/\varepsilon$ without any pair of samples exceeding $2\ln(3/\delta)/\varepsilon$. A tighter analysis would improve the constant slightly which might matter for practical applications.

3.6 Pure Differential Privacy

In this section, we discuss how to achieve ε-differential privacy. We cannot use our approach from Section 3.5 where we add the same noise to all keys because the set of stored keys can differ between sketches for neighboring datasets. Instead, we achieve privacy by adding noise to all elements of $\mathcal{U}$ scaled to the ℓ_1-sensitivity. Chan et al. [CLSX12] showed that the sensitivity of Misra-Gries sketches scales with the number of counters. We show that a simple post-processing step reduces the sensitivity of the sketch to 2 and the worst-case error of the sketch is still $n/(k+1)$ where $n = |S|$. This allows us to achieve an error of $n/(k+1) + O(\log(d)/\varepsilon)$.

The ℓ_1-sensitivity scales with the size of the sketch since the counts can differ by 1 for all k elements between neighboring datasets. This happens when the decrement step is executed on a given input one fewer or one more time than on a neighboring input. We get around this case by post-processing the sketch before adding noise. We first run the Misra-Gries algorithm on the stream but we count how many times the counters were decremented. That is, we count the number of times Branch 2 of Algorithm 12 was executed and denote this count as γ. The Misra-Gries algorithm decrements the counters at most $\lfloor n/(k+1) \rfloor$ times. We use this fact by first adding γ and then subtracting $n/(k+1)$ from each counter in the sketch. We then remove all elements with negative counters. Although we increase the error of the sketch for some datasets, the worst-case error guarantee is still the same as at most $n/(k+1)$ has been subtracted from each count. Next, we show how this post-processing step reduces the ℓ_1-sensitivity to 2.

Let $S \sim S'$ denote any pair of neighboring streams where S' is obtained by removing one element from S. Consider the effect of running the following procedure on the Misra-Gries sketches for both streams (1) add γ and γ' to the counters of MG_S and $\text{MG}_{S'}$, respectively (2) subtract $|S|/(k+1)$ from the counters in both sketches (3) remove any negative counters from both sketches. It can be shown that the new sketches are either identical or differ by 1 in a single counter. Specifically, we may use the argument from the proof of Lemma 3.1 to argue that we end in one of the 6 states introduced in that proof before running the procedure. One may verify that the claim holds in all 6 states. Specifically, we get $\gamma = \gamma' + 1$ in the first 2 states and $\gamma = \gamma'$ for the final 4 states. The post-processing step we introduced in the previous paragraph uses the length

of the stream which differs by 1 between S and S'. As such, there is an additional difference of $1/(k+1)$ for each counter. The ℓ_1-sensitivity is bounded by 2 since $1 + k/(k+1) < 2$.

We achieve ε-differential privacy by adding noise to our new sketch. We essentially use the same technique as Chan et al. [CLSX12] but the noise no longer scale linearly in k as the sensitivity is bounded by 2. Specifically, we add noise sampled from Laplace$(2/\varepsilon)$ independently to the count of each element from $\mathcal{U}$ and release the top-k noisy counts. A simple union bound shows us that with probability at least $1-\beta$ the absolute value of all samples is bounded by $2\ln(d/\beta)/\varepsilon$. Note that it might be infeasible to actually sample noise for each element when $\mathcal{U}$ is large; we refer to previous work on how to implement this more efficiently [CLSX12, CPST12, BV19].

It is worth noting that the low sensitivity of the post-processed sketch can also be utilized under (ε, δ)-differential privacy. We can use an approach similar to [KKMN09]. They add noise to all non-zero counters and remove noisy counts below a threshold to hide small counters. Applying the standard approach for histograms would require a threshold with a small dependence on k as neighboring sketches might disagree on all keys. However, [ALP22] extended the technique to real-valued vectors by probabilistically rounding elements with a value less than the ℓ_1-sensitivity. If we apply their technique directly we get a threshold of $4 + 2\ln(1/\delta)/\varepsilon$. This approach has error guarantees that match those from Theorem 3.1 up to constant factors. However, this approach has worse guarantees than Algorithm 13 when comparing to the non-private Misra-Gries sketch. By Lemma 3.5 the error of Algorithm 13 is $O(\log(1/\delta)/\varepsilon)$ with high probability (for sufficiently small δ). Here the error is $n/(k+1) + O(\log(1/\delta)/\varepsilon)$ since we subtract up to $n/(k+1)$ from the counters before adding noise.

3.7 Privatizing merged sketches

In practice, it is often important that we may merge sketches. This is for example commonly used when we have a dataset distributed over many servers. Each dataset consists of multiple streams in this setting, and we want to compute an aggregated sketch over all streams. We say that datasets are neighboring if we can obtain one from the other by removing a single element from one of the streams. If the aggregator is untrusted we must add noise to each sketch before performing any

merges. This is the setting in [CLSX12] and we can run their merging algorithm. However, since we add noise to each sketch the error scales with the number of sketches. In particular, the error from the thresholding step of Algorithm 13 scales linearly in the number of sketches for worst-case input. In the rest of this section, we consider the setting where aggregators are trusted. We can apply the post-processing step from the previous section to each sketch before aggregating the counters of each element. The ℓ_1-sensitivity of the aggregated sketch is still bounded by 2 so we can use the approach from the previous section. However, the aggregated sketch might have much more than k counters. Next we consider a merging algorithm where aggregators never store more than $2k$ counters.

Agarwal, Cormode, Huang, Phillips, Wei, and Yi [ACH+13] introduced the following simple merging algorithm in the non-private setting. Given two Misra-Gries sketches $T_1, c_1 \leftarrow \text{MG}(k, S_1)$ and $T_2, c_2 \leftarrow \text{MG}(k, S_2)$ they first compute the sum of all counters $c_1 + c_2$. There are up to $2k$ counters at this point. They subtract the value of the $k+1$'th largest counter from all elements. Finally, any non-positive counters are removed leaving at most k counters. They show that merged sketches have the same worst-case guarantee as non-merged Misra-Gries sketches. That is, if we compute a Misra-Gries sketch for each stream $(S_1, \ldots, S_m)$ and merge them into a single sketch, the frequency estimate of all elements is at most $N/(k+1)$ less than the true frequency. Here N is the total length of all streams. This holds for any order of merging and the streams do not need to have the same length.

Unfortunately, the structure between neighboring sketches where either a single counter or exactly k counters differ by 1 breaks down when merging. Therefore we cannot run Algorithm 13 on the merged sketch. However, as we show below, the global sensitivity of merged sketches is independent of the number of merges. The sensitivity only depends on the number of counters. We first show a property for a single merge operation; this will allow us to bound the sensitivity for any number of merges. Note that unlike in the previous section, we do not limit the number of keys that differ between sketches and we do not store keys with a count of zero.

Lemma 3.6. *Let T_1, c_1, T_1', c_1' and T_2, c_2 denote Misra-Gries sketches of size k and denote the sketches merged with the algorithm above as $\hat{T}, \hat{c} \leftarrow \text{Merge}(T_1, c_1, T_2, c_2)$ and $\hat{T}', \hat{c}' \leftarrow \text{Merge}(T_1', c_1', T_2, c_2)$. If $T_1' \subseteq T_1$ and*

$c_{1i} - c'_{1i} \in \{0,1\}$ for all $i \in \mathcal{U}$ then at least one of the following holds (1) $\hat{T}' \subseteq \hat{T}$ and $\hat{c}_i - \hat{c}'_i \in \{0,1\}$ for all $i \in \mathcal{U}$ or (2) $\hat{T} \subseteq \hat{T}'$ and $\hat{c}'_i - \hat{c}_i \in \{0,1\}$ for all $i \in \mathcal{U}$.

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

Corollary 3.2. *Let $(S_1, \ldots, S_m)$ and $(S'_1, \ldots, S'_m)$ denote two sets of streams such that $S_i \sim S'_i$ for one $i \in [m]$ and $S_j = S'_j$ for any $j \neq i$. Let T, c and T', c' be the result of merging Misra-Gries sketches computed on both sets of streams in any fixed order. Then c and c' differ by 1 for at most k elements and agree on all other counts.*

Proof. It is clearly true for sketches of a pair of neighboring datasets by Lemma 3.1. It holds by induction after each merging operation by Lemma 3.6. □

Since the ℓ_1-sensitivity is k we can use the algorithm in [CLSX12] that adds noise with magnitude k/ε to all elements in $\mathcal{U}$ and keeps the top-k noisy counts. If we only add noise to non-zero counts we can hide that up to k keys can change between neighboring inputs with a threshold. The two approaches have expected maximum error compared to the non-private merged sketch of $O(k \log(d)/\varepsilon)$ and $O(k \log(k/\delta)/\varepsilon)$, respectively.

Chapter 4

Differentially Private Vector Aggregation when Coordinates have Different Sensitivity

Poster presented at the Theory and Practice of Differential Privacy Workshop Series (TPDP 2022)

Joint work with: Rasmus Pagh

We consider the problem of releasing of a sum of vectors with differential privacy in the setting where coordinates have different sensitivity, represented by a (known) vector $\overrightarrow{\Delta}$. Each of n users provides a d-dimensional real-valued vector where the ith coordinate is in $[-\overrightarrow{\Delta}_i/2, \overrightarrow{\Delta}_i/2]$. The aim is to minimize the expected pth moment of the error introduced by the mechanism. Focusing on mechanisms that add independent Gaussian or Laplace noise to each coordinate of the sum, we consider the constrained optimization problem of minimizing the expected error. For example, in the case of $p = 1$ (average error across coordinates) it turns out to be optimal to scale the magnitude of Gaussian noise on coordinate i proportionally to $\overrightarrow{\Delta}_i^{2/3}$, and for $p = 2$ (mean squared error) one should scale the magnitude of Gaussian noise proportionally to $\sqrt{\overrightarrow{\Delta}_i}$. Compared to the Gaussian mechanism and the Mininum Enclosing Ellipsoid mechanism, this decreases the 2nd moment of the noise by a factor $d||\overrightarrow{\Delta}||_2^2/||\overrightarrow{\Delta}||_1^2$, which is between 1 and d depending on the skew of coordinates in $\overrightarrow{\Delta}$.

4.1 Introduction

Privately releasing the sum of vectors (or equivalently their mean) is a typical problem within the field of differential privacy. It occurs naturally in machine learning applications, for example when aggregating gradient vectors where each vector is derived from data that needs to be kept private [ACG+16]. The problem can be solved, in the curator model, by the Laplace mechanism [DMNS06] or the Gaussian mechanism (see [DR14, BW18]), both of which achieve differential privacy by adding i.i.d. noise to each coordinate. The magnitude of the noise needed scales with the maximum ℓ_1-norm of inputs (in the case of Laplace noise) or the maximum ℓ_2-norm of inputs (in the case of Gaussian noise). In this chapter we are interested in a situation where we have more information about the input vectors, in the form of *coordinate-wise* bounds on the magnitude of vector entries. Specifically, each of n users provides a d-dimensional real-valued vector where the ith coordinate is in $[-\overrightarrow{\Delta}_i/2, \overrightarrow{\Delta}_i/2]$, where $\overrightarrow{\Delta}$ is a known vector encoding *coordinate sensitivities*. The motivation is that many naturally occurring datasets have skewed distributions, and we want to be able to take advantage of this structure when designing mechanisms for vector aggregation.

The problem we study is a special case of a more general setting [HT10, NTZ16] in which the contribution of each user to a sum is in the convex hull of a point set $P \subseteq \mathbb{R}^d$. In our case the point set consists of the corners of a hyperrectangle with edge lengths $\{\overrightarrow{\Delta}_1, \ldots, \overrightarrow{\Delta}_d\}$[1]. Hardt and Talwar [HT10] introduce the *K-norm mechanism* which achieves pth moment noise magnitude (in our setting) proportional to $||\overrightarrow{\Delta}||_p^p$. Nikolov, Talwar, and Zhang [NTZ16] propose the *Least Squares Mechanism* and show that its second moment error is within a polylogarithmic factor from optimal. A simpler baseline algorithm, also discussed in their work, is the *Minimum Volume Ellipsoid Mechanism* which in our setting adds Gaussian noise scaled proportionally to $\overrightarrow{\Delta}_i$ to the ith coordinate, achieving noise with magnitude similar to the K-norm mechanism. This baseline solution has also been suggested in on-line forums as a potentially "best" solution (see [Mar20]). However, as noted by Nikolov et al. [NTZ16] this mechanism is not optimal, motivating the need for better mechanisms.

[1]The formulation in these papers is to express the mean of user inputs in the form of a convex combination of points in P, but this is equivalent to computing the sum of vectors since n is public knowledge.

Our contribution

In this chapter we consider mechanisms where independent noise is added to the ith coordinate of the sum (with either Gaussian or Laplace distribution) with magnitude depending on i and $\overrightarrow{\Delta}$. To minimize the pth moment of the error, we consider how the noise is best distributed among coordinates under the privacy constraint. Like the Minimum Volume Ellipsoid Mechanism we sample noise at each coordinate with magnitude based on the coordinatewise sensitivities. However, scaling the noise differently allows us to reduce the overall noise when the magnitudes of entries are skewed. At a high level, we reduce the noise for coordinates with low sensitivity by slightly increasing the noise at coordinates with high sensitivity.

We show that there is a choice of parameters for Gaussian noise such that the pth moment of the error, $\mathbb{E}\left[\|\mathcal{M}(x) - f(x)\|_p^p\right]$ where $f(x)$ is the true sum, is proportional to $\|\overrightarrow{\Delta}\|_{2p/(p+2)}^p$. In comparison, the standard Gaussian mechanism has error proportional to $d\|\overrightarrow{\Delta}\|_2^p$, which is never smaller and can be up to d times bigger. The Minimum Volume Ellipsoid Mechanism has error proportional to $d^{p/2}\|\overrightarrow{\Delta}\|_p^p$, which is also never smaller than our mechanism. Similarly, for Laplace noise we can obtain error proportional to $\|\overrightarrow{\Delta}\|_{p/(p+1)}^p$ while the standard Laplace mechanism has error proportional to $d\|\overrightarrow{\Delta}\|_1^p$.

4.2 Preliminaries

Problem setup. We consider the problem of releasing the sum of vectors under differential privacy. Given a matrix $x \in \mathbb{R}^{n\times d}$ with n rows, each a d-dimensional real-valued vector, our goal is to estimate the coordinatewise sum over all rows. Given an input dataset $x \in \mathbb{R}^{n\times d}$, we denote the sum over all rows as $f(x) := \sum_{i\in[n]} x_i$. We write $[n]$ to denote $\{1, \ldots, n\}$. We consider the problem in the setting with bounded sensitivity for each coordinate. We discuss this below in Definition 4.2.

Differential privacy. We define two datasets as neighbors if they differ for at most a single row. That is, $x, x' \in \mathbb{R}^{n\times d}$ are neighboring datasets if and only if $|\{i \in [n] : x_i \neq x'_i\}| \leq 1$. We denote neighboring dataset as $x \sim x'$. The goal of differential privacy is to preserve privacy by limiting

the impact of any one users data on the output distribution. We consider ε-differential privacy and ρ-zCDP in this chapter as defined below.

Definition 4.1 (Differential Privacy [BS16, DMNS06])**.** A randomized algorithm $\mathcal{M}\colon \mathbb{R}^{n\times d} \to \mathcal{R}$ is ε-differentially private if for all measurable sets of outputs $Z \subseteq \mathcal{R}$ and all pairs of neighboring datasets $x \sim x'$ it holds that

$$\Pr[\mathcal{M}(x) \in Z] \leq e^{\varepsilon} \cdot \Pr[\mathcal{M}(x') \in Z] \ .$$

A randomized mechanism $\mathcal{M}\colon \mathbb{R}^{n\times d} \to \mathcal{R}$ satisfies ρ-zCDP if and only if for all $\alpha > 1$ and all pairs of neighboring datasets $x \sim x'$ it holds that

$$D_\alpha\left(\mathcal{M}(x)||\mathcal{M}(x')\right) \leq \rho\alpha \ ,$$

where $D_\alpha(P||Q) := \frac{1}{\alpha-1}\ln\left(\mathbb{E}_{x\sim P}\left[(P(x)/Q(x))^{\alpha-1}\right]\right)$ denotes the α-Rényi divergence between two distributions P and Q.

The notion of sensitivity captures the biggest change to the output of a query over all pairs of neighboring datasets. In our setting the sensitivity is bounded for each coordinate. Each user provides a d-dimensional real-valued vector where the ith coordinate is in $\left[-\overrightarrow{\Delta}_i/2, \overrightarrow{\Delta}_i/2\right]$. As such the sum at the ith coordinate changes by at most $\overrightarrow{\Delta}_i$ for neighboring datasets. Throughout the chapter, we use the notion as specified in the definition below.

Definition 4.2 (Sensitivity)**.** Define $f(x) := \sum_{i\in[n]} x_i$ as the sum over all rows of x. We write $\overrightarrow{\Delta} = (\overrightarrow{\Delta}_1, \ldots, \overrightarrow{\Delta}_d)$ to denote the sensitivity at each coordinate such that

$$\overrightarrow{\Delta}_i = \max_{x\sim x'} |f(x)_i - f(x')_i| \ .$$

We denote the ℓ_2-sensitivity as

$$\Delta_2 = \max_{x\sim x'} \|f(x)_i - f(x')_i\|_2 \ .$$

It is easy to see that $\|f(x) - f(x')\|_2$ is maximized when $f(x) - f(x') = \overrightarrow{\Delta}$. Therefore we have $\Delta_2 = \|\overrightarrow{\Delta}\|_2$. The same argument shows that the ℓ_1-sensitivity is $\Delta_1 = \|\overrightarrow{\Delta}\|_1$.

Error measure. Let $f(x) := \sum_{i\in[d]} x_i$ be the true sum of all rows and let $\mathcal{M} : \mathbb{R}^{n\times d} \to \mathbb{R}^d$ be a randomized algorithm estimating $f(x)$. We measure the accuracy of an estimate in terms of the pth power of the absolute error. For any $p > 0$ the error of $\mathcal{M}(x)$ is $\|\mathcal{M}(x) - f(x)\|_p^p := \sum_{i\in[d]} |\mathcal{M}(x)_i - f(x)_i|^p$. We are concerned with the *pth moment* of the error of $\mathcal{M}$, that is, $\mathbb{E}\left[\|\mathcal{M}(x) - f(x)\|_p^p\right]$. This generalized error metric includes some common error metrics such as average absolute error and the mean squared error. Note that the problem setup is sometimes studied as the equivalent problem of estimating the mean of all vectors. Converting the error from the aggregation setting to the mean setting simply requires scaling by n^{-p}.

The Gaussian Mechanism. The Gaussian Mechanism is an often used algorithm for differentially private release of real-valued queries. The mechanism adds i.i.d. noise from the Gaussian distribution to each coordinate. The mechanism is shown in Algorithm 14.

Algorithm 14: The Gaussian Mechanism

Parameters: Scale $\sigma > 0$.
Input : Dataset $x \in \mathbb{R}^{n\times d}$.
Output : ρ-zCDP estimate of $f(x) := \sum_{i\in[n]} x_i$.

(1) Sample $\eta_i \sim \mathcal{N}\left(0, \sigma^2\right)$ independently for each $i \in [d]$.
(2) Return $f(x) + \eta$.

Lemma 4.1 ([BS16, Proposition 1.6])**.** *The Gaussian Mechanism satisfies ρ-zCDP if and only if $\sigma^2 \geq \Delta_2^2/(2\rho)$.*

Throughout the chapter we consider the Gaussian Mechanism in terms of ρ-zCDP. But our technique applies to other definitions of differential privacy as well. The Gaussian mechanism also satisfies Approximate Differential Privacy [DR14, BW18], Concentrated Differential Privacy [DR16], Rényi Differential Privacy [Mir17], and Gaussian Differential Privacy [DRS19]. For each of these privacy notions there exists an analogue version of Lemma 4.1.

Lemma 4.2. *For any of (μ, τ)-CDP, (ε, δ)-DP, (α, ε)-RDP, and μ-GDP, let σ_{opt} be the smallest value such that the Gaussian Mechanism satisfies the chosen privacy notion for input with ℓ_2-sensitivity of 1. Then the Gaussian Mechanism is differentially private under the chosen definition if and only if $\sigma \geq \sigma_{opt}\|\Delta\|_2$*

Our results for ρ-zCDP extends to any of the above privacy definitions by replacing $\sqrt{1/(2\rho)}$ with σ_{opt}.

We use the absolute moments of a Gaussian distribution shown below to calculate the expected error.

Lemma 4.3 ([Win12, Equation 18])**.** *The pth absolute moment of a zero centered Gaussian distribution for any $p > 0$ is*

$$\mathbb{E}\left[|\mathcal{N}(0,\sigma^2)|^p\right] = \sigma^p \cdot \frac{2^{p/2}\Gamma\left(\frac{p+1}{2}\right)}{\sqrt{\pi}} \ .$$

The expected error of the Gaussian Mechanism follows from the linearity of expectation.

Corollary 4.1. *Let $\mathcal{M}$ denote Algorithm 14 with parameter $\sigma = \Delta_2/\sqrt{2\rho}$. Then the expected error of $\mathcal{M}$ for any input $x \in \mathbb{R}^{n\times d}$ is*

$$\mathbb{E}\left[\|\mathcal{M}(x) - f(x)\|_p^p\right] = d \cdot \Delta_2^p \cdot \frac{\Gamma\left(\frac{p+1}{2}\right)}{\rho^{p/2}\sqrt{\pi}} \ .$$

The Laplace Mechanism. The Laplace Mechanism works similar to the Gaussian Mechanism by adding i.i.d. noise to each coordinate. The noise is sampled from the Laplace distribution with magnitude based on ℓ_1-sensitivity and ε. The mechanism is described in Lemma 4.4.

Lemma 4.4 (The Laplace Mechanism [DMNS06])**.** *The probability density function of the Laplace distribution centered around 0 with scale s is*

$$f_s(x) = \frac{1}{2s}e^{-|x|/s} \ .$$

The Laplace Mechanism satisfies ε-differential privacy by adding i.i.d. noise from $\text{Laplace}(\Delta_1/\varepsilon)$ *to each coordinate.*

We can find the pth absolute moment of the Laplace distribution as shown below.

Lemma 4.5. *The pth absolute moment of a zero centered Laplace distribution for any $p > 0$ is*

$$\mathbb{E}\left[|\text{Laplace}(s)|^p\right] = \int_{-\infty}^{\infty} \frac{1}{2s}e^{-|x|/s}|x|^p dx = s^p \cdot \Gamma(p+1) \ .$$

4.3 Calibrating elliptical Gaussian noise

The Gaussian Mechanism discussed in the previous section is calibrated based on the ℓ_2-sensitivity. However, in our setting we have additional information about the sensitivity at each individual coordinate. The standard mechanism adds noise of the same magnitude at each coordinate and as such we might add a lot of noise to entries even though their sensitivity is low.

We can think of the Gaussian Mechanism as adding spherical noise to the output. The mechanism satisfies differential privacy if $\|f(x) - f(x')\| \leq \Delta_2$ holds for all $x \sim x'$. That is, the vector $f(x) - f(x')$ always lies within the d-dimensional ball centered at the origin with radius Δ_2. We can easily generalize the Gaussian Mechanism to elliptical noise. The idea is to sample noise from Gaussian distributions where the magnitude can differ between coordinates. This allows us to add less noise at coordinates with small sensitivity by adding slightly more noise at coordinates with high sensitivity, keeping the overall privacy budget. The general mechanism, parameterized by a vector b is shown in Algorithm 15.

Algorithm 15: Elliptical Gaussian Mechanism

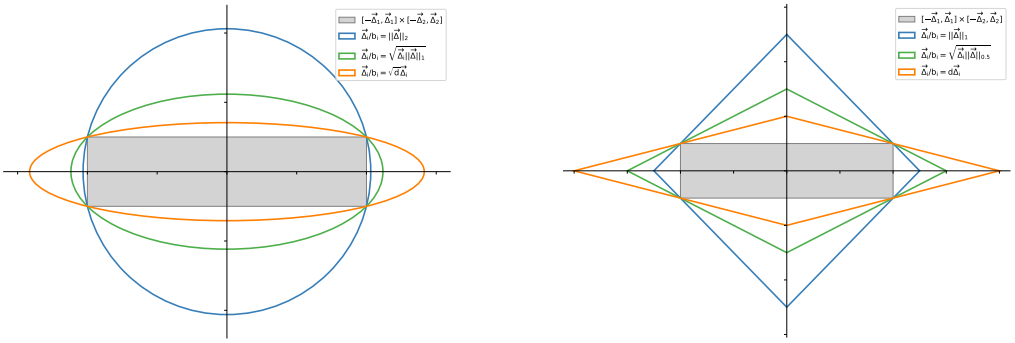
Parameters: $\rho > 0$ and $\overrightarrow{\Delta} \in \mathbb{R}^d_{>0}$ and $b \in \mathbb{R}^d_{>0}$ with $\|b\|_2 = 1$.
Input : Dataset $x \in \mathbb{R}^{n \times d}$.
Output : ρ-zCDP estimate of $f(x) := \sum_{i \in [n]} x_i$.

(1) Let $\sigma_i = \overrightarrow{\Delta}_i / (b_i \sqrt{2\rho})$.
(2) Sample $\eta_i \sim \mathcal{N}(0, \sigma_i^2)$ independently for each $i \in [d]$.
(3) Return $f(x) + \eta$.

Lemma 4.6. *Algorithm 15 satisfies ρ-zCDP.*

Proof. Consider the following equivalent algorithm. Construct a new dataset $\hat{x} \in \mathbb{R}^{n \times d}$ by scaling the entries of x such that $\hat{x}_{j,i} = x_{j,i} \cdot b_i / \overrightarrow{\Delta}_i$. Notice that the transformation means that any row that lies within the d-dimensional ellipsoid with radii $(b_1 / \overrightarrow{\Delta}_1, \ldots, b_d / \overrightarrow{\Delta}_d)$ centered at the origin is mapped to within the unit ball centered at the origin. In particular, for neighboring datasets $x \sim x'$ we have that

$$\|f(\hat{x}) - f(\hat{x}')\|_2 \leq \sqrt{\sum_{i \in [d]} \left(\overrightarrow{\Delta}_i \cdot b_i / \overrightarrow{\Delta}_i \right)^2} = 1 \ .$$



(a) Algorithm 15 with various b

(b) Algorithm 16 with various b

Figure 4.1: Two-dimensional example of noise from Algorithms 15 and 16 with three choices of b. The ellipses and parallelograms bound the set of input vectors for which the mechanism is private, all touching the corners of the axis aligned box that contains all input vectors. The coordinatewise sensitivities are 4 and 1, respectively. The magnitude of noise at each coordinate is scaled by the intersection point between the shape and the axis. For Algorithm 15 depicted in (a) the expected total squared error in both the blue and orange cases is $\mathbb{E}[\|\eta\|_2^2] = d\|\overrightarrow{\Delta}\|_2^2/(2\rho) = 17/\rho$. For the green case we have $\mathbb{E}[\|\eta\|_2^2] = \|\overrightarrow{\Delta}\|_1^2/(2\rho) = 12.5/\rho$. For Algorithm 16 depicted in (b) the expected total error in both the blue and orange cases is $\mathbb{E}[\|\eta\|_1] = d\|\overrightarrow{\Delta}\|_1/\varepsilon = 10/\varepsilon$. For the green case we have $\mathbb{E}[\|\eta\|_1] = \|\overrightarrow{\Delta}\|_{0.5}/\varepsilon = 9/\varepsilon$.

By Lemma 4.1 the Gaussian Mechanism with scale $\sigma = 1/\sqrt{2\rho}$ is ρ-zCDP for inputs with ℓ_2-sensitivity of 1. As such the mechanism is private for $\hat{x}$. Since differential privacy is preserved under post-processing we can scale back the output at each coordinate i by $\overrightarrow{\Delta}_i/b_i$, yielding the same output distribution as Algorithm 15. □

Figure 4.1a depicts a small example of how b affects the noise of Algorithm 15. If $\overrightarrow{\Delta}_i/b_i = \|\overrightarrow{\Delta}\|_2$ the mechanism adds noise of the same magnitude at each coordinate and it is identical to Algorithm 14. The mechanism presented by Mark [Mar20] adds noise proportional to the sensitivity at each coordinate. This corresponds to the case where $\overrightarrow{\Delta}_i/b_i = \sqrt{d}\,\overrightarrow{\Delta}_i$. As a "sweet spot" between these options we may choose $\overrightarrow{\Delta}_i/b_i \propto \sqrt{\overrightarrow{\Delta}_i}$ to reduce the noise at coordinates with low sensitivity with a sublinear dependence on the sensitivity for large coordinates. As we show later in Lemma 4.8 this choice of b is preferred for $p = 2$. The

error of Algorithm 15 for any choice of b is described in the following Lemma.

Lemma 4.7. *The expected error of Algorithm 15 for any $p > 0$ is*

$$\mathbb{E}\left[\|\eta\|_p^p\right] = \frac{\Gamma\left(\frac{p+1}{2}\right)}{\rho^{p/2}\sqrt{\pi}} \sum_{i\in[d]} \left(\overrightarrow{\Delta}_i / b_i\right)^p \quad .$$

Proof. It follows directly from linearity of expectation by computing the sum of the pth absolute moment of all coordinates. By Lemma 4.3 we have

$$\mathbb{E}\left[|\eta_i|^p\right] = (\sigma_i)^p \frac{2^{p/2}\Gamma\left(\frac{p+1}{2}\right)}{\sqrt{\pi}} \quad .$$

□

The values of ρ, p and $\overrightarrow{\Delta}$ are all fixed, but we are free to choose the value of b that minimizes the expected error. As shown in Lemma 4.7 the expected error scales linearly with $\sum_{i\in[d]} \left(\overrightarrow{\Delta}_i / b_i\right)^p$. The optimal choice for b is a function of both $\overrightarrow{\Delta}$ and p shown in Lemma 4.8.

Lemma 4.8. *Given $p > 0$, the expected error $\mathbb{E}\left[\|\eta\|_p^p\right]$ of Algorithm 15 is minimized when*

$$b_i = \frac{\overrightarrow{\Delta}_i^{p/(p+2)}}{\sqrt{\sum_{i\in[d]} \overrightarrow{\Delta}_i^{2p/(p+2)}}} \quad .$$

Proof. We use the method of Lagrange multipliers for the proof. The method is used to find local maxima or minima of a function subject to equality constraints. In our case, it turns out that there is only one minimum and as such we find the global minimum using the method. The expected error is minimized by finding the smallest value of the function $\sum_{i\in[d]} \left(\overrightarrow{\Delta}_i / b_i\right)^p$ subject to the restriction $\|b\|_2 = 1$, i.e, $\sum_{i\in[d]} b_i^2 - 1 = 0$. The minimum is at the stationary point of the Lagrangian function

$$\mathcal{L}(b, \lambda) = \sum_{i\in[d]} \left(\overrightarrow{\Delta}_i / b_i\right)^p + \lambda \left(\sum_{i\in[d]} b_i^2 - 1\right),$$

where $\lambda \in \mathbb{R}$ is the Lagrange multiplier.

We start by finding the partial derivative of the Lagrangian function with respect to b_i.

$$\frac{\partial}{\partial b_i}\mathcal{L}(b,\lambda) = \frac{\partial}{\partial b_i}\left(\overrightarrow{\Delta}_i/b_i\right)^p + \lambda b_i^2 = -p\,\overrightarrow{\Delta}_i^p/b_i^{p+1} + 2\lambda b_i \ .$$

We then find the root of this function with respect to b_i, that is,

$$-p\,\overrightarrow{\Delta}_i^p/b_i^{p+1} + 2\lambda b_i = 0 \Leftrightarrow b_i^{p+2} = p\,\overrightarrow{\Delta}_i^p/(2\lambda) \Leftrightarrow b_i = \overrightarrow{\Delta}_i^{p/(p+2)}/\gamma \ ,$$

where $\gamma = (p/(2\lambda))^{-1/(p+2)}$. We find the value of γ using the restriction on b that is

$$\sum_{i\in[d]} b_i^2 = 1 \quad \Leftrightarrow \quad \sum_{i\in[d]} \overrightarrow{\Delta}_i^{2p/(p+2)} = \gamma^2 \quad \Leftrightarrow \quad \gamma = \sqrt{\sum_{i\in[d]} \overrightarrow{\Delta}_i^{2p/(p+2)}} \ .$$

Substituting this value of γ in $b_i = \overrightarrow{\Delta}_i^{p/(p+2)}/\gamma$ proves the Lemma. □

We are now ready to prove our main result as shown in Theorem 4.1.

Theorem 4.1. *Given $\rho, p > 0$ and a dataset $x \in \mathbb{R}^{n\times d}$ with known coordinate-wise sensitivities $\overrightarrow{\Delta} = \left(\overrightarrow{\Delta}_1, \ldots, \overrightarrow{\Delta}_d\right)$. There exists an ρ-zCDP algorithm $\mathcal{M} : \mathbb{R}^{n\times d} \to \mathbb{R}^d$ that estimates $f(x) := \sum_{i\in[n]} x_i$ with expected error*

$$\mathbb{E}\left[\|\mathcal{M}(x) - f(x)\|_p^p\right] = \frac{\Gamma\left(\frac{p+1}{2}\right)}{\rho^{p/2}\sqrt{\pi}} \cdot \|\overrightarrow{\Delta}\|_{2p/(p+2)}^p \ .$$

Proof. We first find the value of $\left(\overrightarrow{\Delta}_i/b_i\right)^p$ when b_i is chosen according to Lemma 4.8.

$$\begin{aligned}\left(\overrightarrow{\Delta}_i/b_i\right)^p &= \left(\overrightarrow{\Delta}_i^{2/(p+2)}\sqrt{\sum_{i\in[d]}\overrightarrow{\Delta}_i^{2p/(p+2)}}\right)^p \\ &= \overrightarrow{\Delta}_i^{2p/(p+2)}\left(\sum_{i\in[d]}\overrightarrow{\Delta}_i^{2p/(p+2)}\right)^{p/2} \ .\end{aligned}$$

From this we find that

$$\sum_{i\in[d]} (\overrightarrow{\Delta}_i/b_i)^p = \left(\sum_{i\in[d]} \overrightarrow{\Delta}_i^{2p/(p+2)}\right)^{1+p/2} = \|\overrightarrow{\Delta}\|_{2p/(p+2)}^p \ .$$

The Theorem follows from Lemma 4.7 and the calculation above by running Algorithm 15 with b chosen according to Lemma 4.8. □

Throughout the chapter, we used the error measure $\|\eta\|_p^p$. Here we consider the expected error for two typical choices for p and compare the result with the standard Gaussian Mechanism.

For $p = 1$ we set $b_i \propto \overrightarrow{\Delta}_i^{1/3}$ and find that the expected total error of Algorithm 15 is

$$\mathbb{E}\left[\|\eta\|_1\right] = \frac{\|\overrightarrow{\Delta}\|_{2/3}}{\sqrt{\rho\pi}} \ .$$

To instead minimize the expectation total squared error we set $p = 2$ and $b_i \propto \overrightarrow{\Delta}_i^{1/2}$ such that

$$\mathbb{E}\left[\|\eta\|_2^2\right] = \frac{\|\overrightarrow{\Delta}\|_1^2}{2\rho} \ .$$

The ratio between the expected error of Algorithms 14 and 15 is $d\|\overrightarrow{\Delta}\|_2/\|\overrightarrow{\Delta}\|_{2/3}$ and $\left(\sqrt{d}\|\overrightarrow{\Delta}\|_2/\|\overrightarrow{\Delta}\|_1\right)^2$ for $p = 1$ and $p = 2$, respectively. The improvement is most significant when $\overrightarrow{\Delta}$ is skewed. If all entries of $\overrightarrow{\Delta}$ are equal the ratio is 1. This is of course expected because the two algorithms are identical since $\overrightarrow{\Delta}_i/b_i = \|\overrightarrow{\Delta}\|_2$. In the other extreme all but one entry of $\overrightarrow{\Delta}$ is zero in which case the error is improved by a factor of d. Since a coordinate with sensitivity 0 contains no information we can assume that all sensitivities are non-zero. As such Algorithm 15 always improves the expected error of Algorithm 14 by a factor in $[1, d)$ for $d > 1$. This is true for all $p > 0$ when choosing $b_i \propto \overrightarrow{\Delta}_i^{p/(p+2)}$.

4.4 The Laplace mechanism

In the previous section we showed how to improve the expected error of the Gaussian Mechanism in our setting by choosing different magnitude of noise at each coordinate. In this section we show that the

approach can be applied to other mechanisms. We consider the Laplace Mechanism which achieves ε-differential privacy. The noise scales with the ℓ_1-sensitivity. The magnitudes of the noise should be distributed differently from the Gaussian Mechanism where the noise is scaled to the ℓ_2-sensitivity. Note that the proofs in this section are shortened as they follow the same ideas as the corresponding proofs from the previous section.

Algorithm 16: Parallelotope Laplace Mechanism

Parameters: $\varepsilon > 0$ and $\overrightarrow{\Delta} \in \mathbb{R}^d_{>0}$ and $b \in \mathbb{R}^d_{>0}$ with $\|b\|_1 = 1$.
Input : Dataset $x \in \mathbb{R}^{n \times d}$.
Output : ε-differentially private estimate of $f(x) := \sum_{i \in [n]} x_i$.

(1) Let $s_i = \overrightarrow{\Delta}_i / (b_i \cdot \varepsilon)$.
(2) Sample $\eta_i \sim \text{Laplace}(s_i)$ independently for each $i \in [d]$.
(3) Return $f(x) + \eta$.

Lemma 4.9. *Algorithm 16 satisfies ε-differential privacy.*

Proof. The proof is similar to the proof of Lemma 4.6. We consider an equivalent algorithm by scaling the input such that $\hat{x}_{j,i} = x_{j,i} \cdot b_i / \overrightarrow{\Delta}_i$. For neighboring datasets $x \sim x'$ we have

$$\|f(\hat{x}) - f(\hat{x}')\|_1 \leq \sum_{i \in [d]} \left(\overrightarrow{\Delta}_i \cdot b_i / \overrightarrow{\Delta}_i \right) = 1 \ .$$

The mechanism that adds i.i.d. noise from Laplace$(1/\varepsilon)$ to each coordinate of $\hat{x}$ is ε-differentially private by Lemma 4.4. We can scale back each coordinate by $\overrightarrow{\Delta}_i / b_i$ as post-processing. □

Lemma 4.10. *The expected error of Algorithm 16 for any $p > 0$ is*

$$\mathbb{E}\left[\|\eta\|_p^p\right] = \frac{\Gamma(p+1)}{\varepsilon^p} \sum_{i \in [d]} \left(\overrightarrow{\Delta}_i / b_i \right)^p \ .$$

Proof. It follows directly from linearity of expectation by computing the sum of the pth absolute moment of all coordinates. By Lemma 4.5 we have

$$\mathbb{E}\left[|\eta_i|^p\right] = \left(\overrightarrow{\Delta}_i / (b_i \cdot \varepsilon) \right)^p \cdot \Gamma(p+1) \ .$$

□

Lemma 4.11. *Given $p > 0$, the expected error $\mathbb{E}\left[\|\eta\|_p^p\right]$ of Algorithm 16 is minimized when*

$$b_i = \frac{\overrightarrow{\Delta}_i^{p/(p+1)}}{\sum_{i\in[d]} \overrightarrow{\Delta}_i^{p/(p+1)}} \ .$$

Proof. The proof is similar to the proof of Lemma 4.8. We still want to minimize $\left(\overrightarrow{\Delta}_i / b_i\right)^p$ but here it is subject to $\|b\|_1 = 1$. That is, we find the minimum by finding the stationary point of the Lagrangian function

$$\mathcal{L}(b, \lambda) = \sum_{i\in[d]} \left(\overrightarrow{\Delta}_i / b_i\right)^p + \lambda \left(\sum_{i\in[d]} b_i - 1\right).$$

Following the same steps as proof of Lemma 4.8 we find that $b_i \propto \overrightarrow{\Delta}_i^{p/(p+1)}$ for all $i \in [d]$. We use the restriction $\sum_{i\in[d]} b_i = 1$ to find that $b_i = \overrightarrow{\Delta}_i^{p/(p+1)} / \sum_{i\in[d]} \overrightarrow{\Delta}_i^{p/(p+1)}$. □

Theorem 4.2. *Given $\varepsilon, p > 0$ and a dataset $x \in \mathbb{R}^{n\times d}$ with known coordinate-wise sensitivities $\overrightarrow{\Delta} = \left(\overrightarrow{\Delta}_1, \ldots, \overrightarrow{\Delta}_d\right)$. There exists an ε-differentially private algorithm $\mathcal{M} : \mathbb{R}^{n\times d} \to \mathbb{R}^d$ that estimates $f(x) := \sum_{i\in[n]} x_i$ with expected error*

$$\mathbb{E}\left[\|\mathcal{M}(x) - f(x)\|_p^p\right] = \frac{\Gamma(p+1)}{\varepsilon^p} \cdot \|\overrightarrow{\Delta}\|_{p/(p+1)}^p \ .$$

Proof. We find the value of $\sum_{i\in[d]} \left(\overrightarrow{\Delta}_i / b_i\right)^p$ when b_i is chosen according to Lemma 4.11 as such

$$\begin{aligned}\sum_{i\in[d]} (\overrightarrow{\Delta}_i / b_i)^p &= \sum_{i\in[d]} \left(\overrightarrow{\Delta}_i^{1/(p+1)} \sum_{i\in[d]} \overrightarrow{\Delta}_i^{p/(p+1)}\right)^p \\ &= \left(\sum_{i\in[d]} \overrightarrow{\Delta}_i^{p/(p+1)}\right)^{1+p} = \|\overrightarrow{\Delta}\|_{p/(p+1)}^p \ .\end{aligned}$$

The Theorem follows from Lemma 4.10 and the calculation above by running Algorithm 16 with b chosen according to Lemma 4.11. □

Chapter 5

PLAN: Variance-Aware Differentially Private Mean Estimation

Unpublished as of August 2023

Joint work with: Martin Aumüller, Boel Nelson, and Rasmus Pagh

Differentially private mean estimation is an important building block in privacy-preserving algorithms for data analysis and machine learning. Though the trade-off between privacy and utility is well understood in the worst case, many datasets exhibit structure that could potentially be exploited to yield better algorithms. In this chapter we present *Private Limit Adapted Noise (*PLAN*)*, a family of differentially private algorithms for mean estimation in the setting where inputs are independently sampled from a distribution $\mathcal{D}$ over $\mathbb{R}^d$, with coordinate-wise standard deviations $\sigma \in \mathbb{R}^d$. Similar to mean estimation under Mahalanobis distance, PLAN tailors the shape of the noise to the shape of the data, but unlike previous algorithms the privacy budget is spent non-uniformly over the coordinates. Under a concentration assumption on $\mathcal{D}$, we show how to exploit skew in the vector σ, obtaining a (zero-concentrated) differentially private mean estimate with ℓ_2 error proportional to $\|\sigma\|_1$. Previous work has either not taken σ into account, or measured error in Mahalanobis distance — in both cases resulting in ℓ_2 error proportional to $\sqrt{d}\|\sigma\|_2$, which can be up to a factor $\sqrt{d}$ larger. To verify the effectiveness of PLAN, we empirically evaluate accuracy on both synthetic and real-world data.

5.1 Introduction

Differentially private mean estimation is an important building block in many algorithms, notably in for example implementations of private stochastic gradient descent [ACG+16, PSY+19, DMR+22]. While differential privacy is an effective and popular definition for privacy-preserving data processing, privacy comes at a cost of accuracy, and striking a good trade-off between utility and privacy can be challenging. Achieving a good trade-off is especially hard for high-dimensional data, as the required *noise* that ensures privacy increases with the number of dimensions. Making matters worse, differential privacy operates on a worst-case basis. Adding noise naïvely may result in a *one-size-fits-none* noise scale that, although private, fails to give meaningful utility for many datasets.

Motivating example. Consider the following toy example: take a constant $q \in (0, 1)$ and let $\lambda \ll 1/d$ be a value close to zero. We consider a set of vectors $x^{(1)}, \ldots, x^{(n)} \in \mathbb{R}^d$ where, independently,

$$x^{(j)} = \begin{cases} (1, \lambda, \ldots, \lambda) & \text{with probability } q \\ (0, 0, \ldots, 0) & \text{with probability } 1 - q \end{cases} .$$

Though the nominal dimension is d, the distribution is "essentially 1-dimensional". Thus, we should be able to release a private estimate of the mean $(q, \lambda q, \ldots, \lambda q)$ with about the same precision as a scalar value with sensitivity 1. However, previous private mean estimation techniques either:

- add the same amount of noise to all d dimensions, making the ℓ_2 norm of the error a factor $\Theta(\sqrt{d})$ larger, or
- split the privacy budget evenly across the d dimensions, making the noise added to the first coordinate a factor $\Theta(\sqrt{d})$ larger than in the scalar case.

Instead, we would like to adapt to the situation, and spend most of the privacy budget on the first coordinate while still keeping the noise on the remaining $d - 1$ coordinates low. Dimension reduction using private PCA (see e.g. [ADK+19, DTTZ14, HP14]) could be used, but we aim for something simpler and more general: to spend more budget on coordinates with higher variance.

We explore the design space of such variance-aware budget allocations and show that, perhaps surprisingly, the intuitive approach of splitting the budget across coordinates proportional to their spread (which is optimal for Mahalanobis distance [BGS+21]) is *not optimal* for ℓ_p errors. Using this insight, we design a family of algorithms, Private Limit Adapted Noise (PLAN), that distributes the privacy budget optimally. The contributions of this work cover both theoretical and practical ground:

- We present Private Limit Adapted Noise (PLAN) (Section 5.3), a family of algorithms for differentially private mean estimation for ℓ_2 error, and its accompanying privacy and utility analysis (Section 5.4).
- We formalize, introduce, and exemplify the notion of *(σ, p)-well concentrated* distributions (Section 5.4 & Section 5.5) that allows for the study of variance-aware algorithms for mean estimation.
- We generalize the analysis of PLAN to hold for arbitrary ℓ_p error (Section 5.4.6).
- We prove that PLAN matches the utility of the current state-of-the-art algorithm [HLY21] for differentially private mean estimation up to constant factors in expectation — without requiring computationally expensive random rotations (Section 5.6).
- We implement two instantiations of PLAN, for ℓ_1 and ℓ_2 error respectively, and empirically evaluate the algorithms on synthetic and real-world datasets (Section 5.7).

Scope. We consider the setting where we have independent samples $x^{(1)}, \ldots, x^{(n)}$ from a distribution $\mathcal{D}$ over $\mathbb{R}^d$, and use $\sigma \in \mathbb{R}^d$ to denote the vector where σ_i is the standard deviation of the ith coordinate of a sample from $\mathcal{D}$. For mean estimation, error is often expressed in terms of *Mahalanobis distance* (Section 5.2.2), which is natural if $\mathcal{D}$ is a Gaussian distribution. Our objective is instead to estimate the mean μ of $\mathcal{D}$ with small ℓ_p error. This is natural in other settings, for example if input vectors:

- represent probability distributions over $\{1, \ldots, d\}$, so that ℓ_1 error corresponds to variation distance, or

- are binary, so that ℓ_1 and ℓ_2 error corresponds to mean error and root mean square error, respectively, over d counting queries.

Our exposition will focus primarily on the case of ℓ_2 error, which is most easily compared to previous work.

PLAN's privacy guarantees are expressed as ρ-zero-Concentrated Differential Privacy (zCDP), where similar results follow for other notions, such as approximate differential privacy (Lemma 5.1). We consider the number of inputs, n, fixed — the neighboring relation is changing one vector among the inputs.

In the *non-private* setting, the empirical mean $\frac{1}{n}\sum_j x^{(j)}$ is known to yield the smallest ℓ_2 error of size $O(\|\sigma\|_2/\sqrt{n})$. We are interested in the situation where the privacy parameter ρ is small enough that the *sampling error* of the empirical mean is dominated by the error introduced by differential privacy.

Our results. We show, both theoretically and empirically, that a careful chosen privacy budgeting can improve the ℓ_p error compared to existing methods when the vector σ is skewed. Some limitation, beyond bounded variance, is needed for the distribution $\mathcal{D}$. We say that $\mathcal{D}$ is *σ-well concentrated* if, roughly speaking, the norm of distance to the mean μ of vectors sampled from $\mathcal{D}$ is unlikely to be much larger than vectors sampled from a multivariate exponential distribution with standard deviations given by σ or some root of σ. In the case $p = 2$ our upper bound is particularly simple to state:

Theorem 5.1. *(simplified version) Suppose $\mathcal{D}$ is σ-well concentrated and that we know $\hat{\sigma}$ such that $\|\sigma - \hat{\sigma}\|_\infty < \|\sigma\|_1/d$. Then for $n = \tilde{\Omega}\left(\max\left(\sqrt{d/\rho}, \rho^{-1}\right)\right)$ there is a mean estimation algorithm that satisfies ρ-zCDP and has expected ℓ_2 error*

$$\tilde{O}(1 + \|\sigma\|_2/\sqrt{n} + \|\sigma\|_1/(n\sqrt{\rho}))$$

with high probability, where $\tilde{O}$ suppresses polylogarithmic factors in error probability, d, n, and a bound on the ℓ_∞ norm of input vectors.

Comparing Theorem 5.1 to applying the Gaussian mechanism directly provides a useful example. Given the vectors $x^{(1)}, \ldots, x^{(n)}$ sampled from $\mathcal{N}(\mu, \Sigma)$ for $\Sigma = \mathrm{diag}(\sigma^2)$ where $\|\mu\|_2 = O(\|\sigma\|_2)$, and clipping at $C = \Theta(\|\sigma\|_2)$, would give an ℓ_2 error bound of

$$\tilde{O}(\|\sigma\|_2/\sqrt{n} + \sqrt{d}\|\sigma\|_2/(n\sqrt{\rho})) \ .$$

Crucially, Theorem 5.1 is never worse, but can be better than applying the Gaussian mechanism directly. The value of $\|\sigma\|_1$ is in the interval $[\|\sigma\|_2, \sqrt{d}\|\sigma\|_2]$, where a smaller value of $\|\sigma\|_1$ indicates larger skew. Thus, Theorem 5.1 is strongest when σ has a skewed distribution such that $\|\sigma\|_1$ is close to the lower bound of $\|\sigma\|_2$, assuming that the privacy parameter ρ is small enough that the error due to privacy dominates the sampling error.

5.2 Preliminaries

We provide privacy guarantees via zero-Concentrated Differential Privacy (zCDP) in the *bounded setting*, where the dataset has a fixed size, as defined in this section. Since previous work measure error using different distance measures, we give both the definition for ℓ_p error (which we use), as well as Mahalanobis distance. Lastly, we give an overview of private quantile estimation which is a central building block of our algorithm.

5.2.1 Differential privacy

Differential privacy [DMNS06] is a statistical property of an algorithm that limits information leakage by introducing controlled randomness to the algorithm. Formally, differential privacy restricts how much the output distributions can differ between any neighboring datasets. We say that a pair of datasets are neighboring, denoted $x \sim x'$, if and only if there exists an $j \in [n]$ such that $x^{(i)} = x'^{(i)}$ for all $i \neq j$.

Definition 5.1 ([DMNS06] (ε, δ)-Differential Privacy)**.** A randomized mechanism $\mathcal{M}$ satisfies (ε, δ)-DP if and only if for all pairs of neighboring datasets $x \sim x'$ and all set of outputs Z we have

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

Definition 5.2 ([BS16] zero-Concentrated Differential Privacy (zCDP))**.** Let $\mathcal{M}$ denote a randomized mechanism satisfying ρ-zCDP for any $\rho > 0$. Then for all $\alpha > 1$ and all pairs of neighboring datasets $x \sim x'$ we have

$$D_\alpha \left(\mathcal{M}(x) || \mathcal{M}(x')\right) \leq \rho\alpha,$$

where $D_\alpha(X||Y)$ denotes the α-Rényi divergence between two distributions X and Y.

Lemma 5.1 ([BS16] zCDP to (ε, δ)-DP conversion)**.** *If $\mathcal{M}$ satisfies ρ-zCDP, then $\mathcal{M}$ is (ε, δ)-DP for any $\delta > 0$ and $\varepsilon = \rho + 2\sqrt{\rho \log(1/\delta)}$.*

Lemma 5.2 ([BS16] Composition)**.** *If M_1 and M_2 satisfy ρ_1-zCDP and ρ_2-zCDP, respectively. Then $M = (M_1, M_2)$ satisfies $(\rho_1 + \rho_2)$-zCDP.*

The Gaussian Mechanism adds noise from a Gaussian distribution independently to each coordinate of a real-valued query output. The scale of the noise depends on the privacy parameter and the ℓ_2-sensitivity denoted Δ_2. A query q has sensitivity Δ_2 if for all $x \sim x'$ we have $\|q(x) - q(x')\|_2 \leq \Delta_2$.

Lemma 5.3 ([BS16] The Gaussian Mechanism)**.** *If $q : \mathbb{R}^{n \times d} \to \mathbb{R}^d$ is a query with sensitivity Δ_2 then releasing $\mathcal{N}(q(x), \frac{\Delta_2^2}{2\rho}\mathbf{I})$ satisfies ρ-zCDP.*

5.2.2 Distance measures

Here we introduce the distance measures for the error of a mean estimate. Let $\tilde{\mu} = \mathcal{M}(x)$ denote the mean estimate of a distribution with mean μ. We measure the utility of our mechanism in terms of expected ℓ_p error as defined below. Here p is a parameter of our mechanism where the most common values for p are 1 (Manhattan distance), and 2 (Euclidean distance).

Definition 5.3 (ℓ_p error)**.** For any real value $p \geq 1$ the ℓ_p error is

$$\|\tilde{\mu} - \mu\|_p = \left(\sum_{i=1}^{d} |\tilde{\mu}_i - \mu_i|^p \right)^{1/p}$$

Note that we sometimes present error guarantees in the form of the pth moment, i.e. $\mathbb{E}[\|\tilde{\mu} - \mu\|_p^p]$, when it follows naturally from the analysis. Notice that the pth moment bounds the expected ℓ_p error for any $p \geq 1$ as $\mathbb{E}[\|\tilde{\mu} - \mu\|_p] \leq (\mathbb{E}[\|\tilde{\mu} - \mu\|_p^p])^{1/p}$.

As stated in Section 5.8 several previous work on mean estimates measure error in Mahalanobis distance. Although this is not the focus of our work we include the definition here for completeness. A key difference between Mahalanobis distance measure and ℓ_p error is that the directions are weighted by the uncertainty of the underlying data. We weight the error in all coordinates equally.

Definition 5.4 (Mahalanobis distance)**.** The error in Mahalanobis distance of a mean estimate for a distribution with covariance matrix Σ is defined as

$$\|\tilde{\mu} - \mu\|_\Sigma = \|\Sigma^{-1/2}(\tilde{\mu} - \mu)\|_2$$

5.2.3 Private quantile estimation

Our work builds on using differentially private quantiles, whose rank error (i.e. how many elements away from the desired quantile the output is) we will define for our utility analysis. When estimating the q'th quantiles of a sequence $z^{(1)} \dots z^{(n)}$ with $z^{(j)} \in \mathbb{R}^d$, we ideally return a vector $\tilde{z} \in \mathbb{R}^d$ such that for each coordinate i we have $|\{j \in [n] : z_i^{(j)} \leq \tilde{z}_i\}|/n = q$. We use the notation $\textsc{PrivQuantile}_\rho^M(z^{(1)} \dots z^{(n)}, q)$ to denote a ρ-zCDP mechanism that estimates the q'th quantiles of $z^{(1)} \dots z^{(n)}$ where each coordinate is bounded to $[-M, M]$.

There are multiple applicable choices for the instantiation of PrivQuantile from the literature, e.g. [Smi11, HLY21, KSS22]. In this chapter we will use the binary search based quantile estimator [Smi11, HLY21] for our utility analysis (Section 5.4), and the exponential mechanism based quantile estimator by Kaplan et al. [KSS22] in our empirical evaluation (Section 5.7). Note that the choice of instantiation of PrivQuantile has no effect on the privacy analysis of PLAN. The binary search based method gives us cleaner theoretical results because the error guarantee of the exponential mechanism based technique is highly data-dependent, and the error is large for worst-case input. Empirically, however, the exponential mechanism based quantile is more robust, which is why we use it in our experiments.

The binary search based quantile subroutine performs a binary search over the interval $[-M, M]$. At each step of the search, branching is based on a quantile estimate for the midpoint of the current interval. Since branching must be performed privately the privacy budget needs to be partitioned in advance, this limits the binary search to T iterations. Huang et al. [HLY21] describes the algorithm for discrete input where T is the base 2 logarithm of the size of the input domain. Treating T as a parameter of the algorithm gives us the following guarantees for 1-dimensional input.

Lemma 5.4 (Follows from [HLY21, Theorem 1])**.** PrivQuantile *satisfies ρ'-zCDP and with probability at least $1 - \beta$ it returns an interval containing at least one point with rank error bounded by $\sqrt{T \log(T/\beta)/(2\rho')}$.*

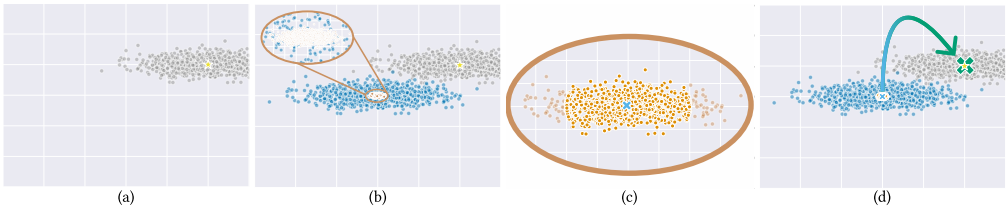


Figure 5.1: Step-by-step illustration of PLAN (Algorithm 17): (a) Raw data, with statistical mean (yellow star), (b) Recentering (blue) and scaling (orange) corresponding to Line 4, (c) Clipping, as determined by Line 6, (d) Private mean (green cross).

When estimating the quantiles of multiple dimensions, we split the privacy budget evenly across each dimension such that $\rho' = \rho/d$ for each invocation of PRIVQUANTILE. By composition releasing all quantiles satisfies ρ-zCDP.

Lemma 5.5. *With probability at least* $1-\beta$, $\text{PRIVQUANTILE}_{\rho}^{M}$ *returns a point that for all coordinates is within a distance of* $M2^{-T}$ *of a point with rank error at most* $\sqrt{dT\log(Td/\beta)/2\rho}$ *for the desired quantile.*

Proof. Running binary search for T iterations splits up the range $[-M, M]$ in 2^T evenly sized intervals. By a union bound there is a point with claimed rank error in each of the intervals returned by PRIVQUANTILE. Returning the midpoint of each interval ensures we are at most distance $M2^{-T}$ from said point. □

Throughout this chapter, we set $T = \log_2(M)$ unless otherwise specified such that the error distance of Lemma 5.5 is 1.

5.3 Algorithm

Conceptually, Private Limit Adapted Noise (PLAN) is a data-aware family of algorithms that exploits variance in the input data to tailor noise to the specific data at hand. Since it is a family of algorithms, a PLAN needs to be instantiated for a given problem domain, i.e. for a specific ℓ_p error and data distribution. In Section 5.7 we showcase two such instantiations, using ℓ_1 error for binary data, and using ℓ_2 for Gaussian data. Pseudocode for PLAN is shown in Algorithm 17, and a step-by-step illustration of PLAN is provided in Figure 5.1.

5.3.1 plan overview

Based on the input data (Line 1, Figure 5.1 (a)) PLAN first computes a private, rough, approximate mean $\tilde{\mu}$ (Line 3). PLAN then recenters the data around $\tilde{\mu}$, and scales the data (Line 4, Figure 5.1 (b)) according to an estimate on the variance $\hat{\sigma}^2$ — this scaling allows the privacy budget to be spent unevenly across dimensions. Next, PLAN clips inputs to a carefully chosen ellipsoid (Line 6, Figure 5.1 (c)) centered around $\tilde{\mu}$ that most data points fall within. The size of the ellipsoid is determined privately based on the data (Line 5). Finally, PLAN adds sufficient Gaussian noise (Line 7) to make the contribution of each clipped input differentially private, and transforms the results back to the original format (Figure 5.1 (d)). Each of these techniques has been used for private mean estimation before — we show that choosing the ellipsoid differently leads to smaller ℓ_p error.

Algorithm 17: PLAN

1: **Input:** $x^{(1)}, \ldots, x^{(n)} \in \mathbb{R}^d$, estimate $\hat{\sigma}^2 \in \mathbb{R}^d$
2: **Parameters:** $M, p, \rho_1, \rho_2, \rho_3$
3: $\tilde{\mu} \leftarrow \text{PrivQuantile}^{M}_{\rho_1}(x^{(1)}, \ldots, x^{(n)}, 1/2)$
4: $y^{(j)} \leftarrow (x^{(j)} - \tilde{\mu})\, \hat{\Sigma}^{-1/(p+2)}$ for $j = 1, \ldots, n$
5: $k \leftarrow \sqrt{n} + \Theta\left(\sqrt{1/\rho_2}\right)$
6: $C \leftarrow \text{PrivQuantile}^{M\sqrt{d}}_{\rho_2}(\|y^{(1)}\|_2, \ldots, \|y^{(n)}\|_2, \frac{n-k}{n})$
7: sample $\eta \sim \mathcal{N}(\mathbf{0}, \frac{2C^2}{\rho_3}\mathbf{I})$
8: **return** $\tilde{\mu} + \frac{1}{n}\left(\sum_j \min\left\{\frac{C}{\|y^{(j)}\|_2}, 1\right\} y^{(j)} + \eta\right) \hat{\Sigma}^{1/(p+2)}$

5.3.2 plan building blocks

Like INSTANCE-OPTIMAL MEAN ESTIMATION (IOME) [HLY21], our algorithm computes a rough estimate of the mean as a private estimate $\tilde{\mu} \in \mathbb{R}^d$ of the coordinate-wise median. This step uses the assumption that all coordinates are in the interval $[-M, M]$. It is known that a finite output domain is needed for private quantile selection to be possible [BNSV15]. Suppose ℓ_p is the error measure we are aiming to minimize. The next step is to scale coordinates by multiplying with the diagonal matrix $\hat{\Sigma}^{-1/(p+2)}$, where $\hat{\Sigma} = \text{diag}(\hat{\sigma}^2)$ is the diagonal matrix of variance estimates $\hat{\sigma}^2$.

Note that since $\hat{\Sigma}$ is a diagonal matrix, it is not necessarily close to the covariance matrix of $\mathcal{D}$ outside of the diagonal. Note that for $p = 2$ the

exponent of $\hat{\Sigma}$ is $-1/4$, which is different from the exponent of $-1/2$ that would be used in order to compute an estimate with small Mahalanobis distance.

The scaling stretches the ith coordinate by a factor $\hat{\sigma}_i^{-2/(p+2)}$. Since $\hat{\sigma}_i^2 \approx \sigma_i^2$ this changes the standard deviation on the ith coordinate from σ_i to roughly $\sigma_i^{p/(p+2)}$. Next, we compute vectors $y^{(j)}$ that represent the (stretched) differences $x^{(j)} - \tilde{\mu}$. Conceptually, we now want to estimate the mean of $y^{(1)}, \ldots, y^{(n)}$, which in turn implies an estimate of $(\frac{1}{n}\sum_j x^{(j)}) - \tilde{\mu}$. The mean of $y^{(1)}, \ldots, y^{(n)}$ is estimated using the Gaussian mechanism. In order to find a suitable scaling of the noise we privately find a quantile C of the lengths $\|y^{(1)}\|_2, \ldots, \|y^{(n)}\|_2$ (all shorter than $M\sqrt{d}$ by assumption) such that approximately k vectors have length larger than C, and clip vectors to length at most C. Clipping, private mean estimation, scaling, and adding back $\tilde{\mu}$ are all condensed in the estimator in Line 8 of Algorithm 17.

5.4 Analysis

Assumptions:

- We assume that for a known parameter M, all inputs are in $x^{(j)} \in [-M, M]^d$ for $j = 1, \ldots, n$. (If this is not the case, the algorithm will clip inputs to this cube, introducing additional clipping error.) Also, we assume that data has been scaled sufficiently such that $\sigma_i \geq 1$ for $i = 1, \ldots, d$ (this can be enforced by adding independent noise of variance 1 to each coordinate of inputs).

- We are given a vector $\hat{\sigma} \in \mathbb{R}^d$ such that

$$\sigma_i \leq \hat{\sigma}_i < \sigma_i + \|\sigma\|_1/d \ . \tag{5.1}$$

  If no such vector is known we will have to compute it, spending part of the privacy budget, but we consider this a separate question.

Definition 5.5. Consider a distribution $\mathcal{D}$ over $\mathbb{R}^d$, denote the mean and standard deviation of the ith coordinate by μ_i and σ_i, respectively. We say that $\mathcal{D}$ is *σ-well concentrated* if for any vector $\hat{\sigma} \in \mathbb{R}^d$ with $\sigma_i \leq \hat{\sigma}_i <$

$\sigma_i + \|\sigma\|_1/d$, the following holds for $t > 1$

$$\Pr_{X\sim\mathcal{D}}\left[\sum_{i=1}^{d}(X_i-\mu_i)^2 > t\|\sigma\|_2^2\right] = \exp(-\tilde{\Omega}(t)) \tag{5.2}$$

$$\Pr_{X\sim\mathcal{D}}\left[\sum_{i=1}^{d}(X_i-\mu_i)^2/\hat{\sigma}_i > t\|\sigma\|_1\right] = \exp(-\tilde{\Omega}(t)) \ . \tag{5.3}$$

Intuitively, these assumptions require concentration of measure of the norms before and after scaling.

5.4.1 Analysis outline

We will show that PLAN returns a private mean estimate that, assuming $\mathcal{D}$ is *σ-well concentrated*, has small expected ℓ_p error with probability at least $1-\beta$. All probabilities are over the joint distribution of the input samples and the randomness of the PLAN mechanism. For simplicity we focus on the case $p = 2$, but the analysis extends to any $p \geq 1$ as we will show at the end of this section.

Privacy. It is not hard to see that PLAN satisfies ρ-zCDP with $\rho = \rho_1 + \rho_2 + \rho_3$: The computation of $\tilde{\mu}$ satisfies ρ_1-zCDP and the computation of C satisfies ρ_2-zCDP by definition of PRIVQUANTILE. Finally, given the values C and $\tilde{\mu}$ the ℓ_2-sensitivity of $\sum_j \min\left\{\frac{C}{\|y^{(j)}\|_2}, 1\right\} y^{(j)}$ with respect to an input $x^{(j)}$ is $2C$, so adding Gaussian noise with variance $2C^2/\rho_3$ gives a ρ_3-zCDP mean estimate. By composition, and since the returned estimator is a post-processing of these private values, the estimator is ρ-zCDP with $\rho = \rho_1 + \rho_2 + \rho_3$.

Utility. It is known that mean estimation in $\mathbb{R}^d$ requires $n = \Omega(\sqrt{d/\rho})$ samples to achieve meaningful utility [HLY21, KLSU19]. Thus we will assume that n is sufficiently large, $n = \tilde{\Omega}(\sqrt{d/\rho})$, where the $\tilde{\Omega}$ notation hides polylogarithmic factors in d, β, and M. Let $\hat{\Sigma}$ denote the scaling matrix $\mathrm{diag}(\hat{\sigma}^2)$. In the following, we assume that ρ_1, ρ_2, ρ_3 are fixed fractions of ρ.

The utility analysis has three parts:

1. First, we argue that $|\tilde{\mu}_i - \mu_i| < 3\sigma_i$ for all $i = 1, \ldots, d$ with high probability.

2. Let J denote the set of indices for which $\|y^{(j)}\|_2 > C$. We argue that with high probability,

$$\sum_{j \in J} \|x^{(j)} - \tilde{\mu}\|_2 = \tilde{O}\left(\left(k + \sqrt{\tfrac{1}{\rho}}\right) \|\sigma\|_2\right),$$

bounding the error due to clipping.

3. Finally, we argue that with high probability

$$\left\| \eta \hat{\Sigma}^{\frac{1}{p+2}} \right\|_2^2 = \tilde{O}(\|\sigma\|_1^2/\rho),$$

bounding the error due to Gaussian noise.

5.4.2 Part 1: Bounding $|\tilde{\mu} - \mu|$

In the following K is a universal constant chosen to be sufficiently large.

Lemma 5.6. *Assuming that* $n > K \max(\sqrt{d \log(M) \log(\log(M) d/\beta)/\rho}, \log(d/\beta))$ *then with probability* $1 - \beta$*, the coordinate-wise median* $\tilde{\mu}_i$ *satisfies* $\tilde{\mu}_i \in [\mu_i - 3\sigma_i, \mu_i + 3\sigma_i]$ *for all* $i = 1, \ldots, d$*.*

Proof. Sample $X = (X_1, \ldots, X_d) \sim \mathcal{D}$. By Chebychev's inequality, for each i with $1 \le i \le d$:

$$\Pr[|X_i - \mu_i| \le 2\sigma_i] \ge 1 - \frac{\sigma_i^2}{(2\sigma_i)^2} = 3/4 \ .$$

Fixing i, these events are independent for $j = 1, \ldots, n$ so by a Chernoff bound with probability $1 - \exp(-\Omega(n))$ there are at least $\frac{2}{3}n$ indices j such that $|X_i^{(j)} - \mu_i| \le 2\sigma_i$. Condition on the event that this is true for $i = 1, \ldots, d$.

If the rank error of the median estimate $\tilde{\mu}_i$ is smaller than $n/6$, $\tilde{\mu}_i \in [\mu_i - 3\sigma_i, \mu_i + 3\sigma_i]$, using the assumption that $\sigma_i \ge 1$. By Lemma 5.5 this is true for every i with probability at least $1 - \beta$, given our assumption on n and K if K is a sufficiently large constant. □

5.4.3 Part 2: Bounding clipping error

Let $J = \{j \in \{1, \ldots, n\} \mid \|y^{(j)}\|_2 > C\}$ denote the set of indices of vectors affected by clipping in PLAN. By the triangle inequality and assuming the bound of part 1 holds,

$$\begin{aligned}\sum_{j \in J} \left\| (x^{(j)} - \tilde{\mu}) \right\|_2 &\leq \sum_{j \in J} \left\| (x^{(j)} - \mu) \right\|_2 + |J| \cdot \|\mu - \tilde{\mu}\|_2 \\ &\leq \sum_{j \in J} \left\| (x^{(j)} - \mu) \right\|_2 + 3|J| \cdot \|\sigma\|_2 \ .\end{aligned}$$

By assumption (5.2) the probability that $\|x^{(j)} - \mu\|_2^2 > t\|\sigma\|_2^2$ is exponentially decreasing in t, so setting $t = \log^2(n/\beta)$ we have $\|x^{(j)} - \mu\|_2 = \tilde{O}(\|\sigma\|_2)$ for all $j = 1, \ldots, n$ with probability at least $1 - \beta$. Using the triangle inequality again we can now bound

$$\sum_{j \in J} \|x^{(j)} - \mu\|_2 = \tilde{O}(|J| \cdot \|\sigma\|_2) \ .$$

Since $\left\| y^{(j)} \right\|_2 \leq M\sqrt{d}$, by Lemma 5.4, with probability at least $1 - \beta$ (over the random choices of the private quantile selection algorithm) there are at most $\tilde{O}\left(k + \sqrt{\frac{1}{\rho}}\right)$ vectors $y^{(j)}$ for which $\|y^{(j)}\|_2 > C$. So with probability at least $1 - \beta$ we have $|J| = \tilde{O}\left(k + \sqrt{\frac{1}{\rho}}\right)$, and using the bounds above we get

$$\sum_{j \in J} \|x^{(j)} - \tilde{\mu}\|_2 = \tilde{O}(|J| \cdot \|\sigma\|_2) = \tilde{O}\left(\left(k + \sqrt{\tfrac{1}{\rho}}\right) \|\sigma\|_2\right) \ .$$

5.4.4 Part 3: Bounding noise

By triangle inequality:

$$\begin{aligned}\|y^{(j)}\|_2 &= \left\| (x^{(j)} - \tilde{\mu})\, \hat{\Sigma}^{-\frac{1}{p+2}} \right\|_2 \\ &\leq \left\| (x^{(j)} - \mu)\, \hat{\Sigma}^{-\frac{1}{p+2}} \right\|_2 + \left\| (\mu - \tilde{\mu})\, \hat{\Sigma}^{-\frac{1}{p+2}} \right\|_2\end{aligned}$$

For $p = 2$, the ith coordinate of $(x^{(j)} - \mu)\, \hat{\Sigma}^{-1/(p+2)}$ equals $(x_i^{(j)} - \mu_i)/\sqrt{\hat{\sigma}_i}$, so assumption (5.3) implies that the length of the first vector is

$\tilde{O}(\sqrt{\|\sigma\|_1})$ with probability at least $1-\beta/2$. From part 1 we know (again for $p=2$) that the ith coordinate of $(\mu-\tilde{\mu})\,\hat{\Sigma}^{-1/(p+2)}$ has absolute value at most

$$3\sigma_i/\sqrt{\hat{\sigma}_i} \le 3\sqrt{\sigma_i},$$

where the inequality uses the lower bound on $\hat{\sigma}_i$ in assumption (5.1). So $\|(\mu-\tilde{\mu})\,\hat{\Sigma}^{-1/(p+2)}\|_2 \le 3\sqrt{\|\sigma\|_1}$, and we get that $\|y^{(j)}\|_2 = \tilde{O}(\sqrt{\|\sigma\|_1})$ for all j with probability at least $1-\beta/2$.

The value of C is bounded by the maximum length of a vector $\|y^{(j)}\|_2$, and thus by a union bound, with probability at least $1-\beta$, $C^2 = \tilde{O}(\|\sigma\|_1)$. The scaled noise vector $\eta\,\hat{\Sigma}^{1/(p+2)}$ has distribution $\mathcal{N}(0, \frac{2C^2}{\rho_3}\hat{\Sigma}^{1/(p+2)})$, so using $\hat{\sigma}_i \le \sigma_i + \|\sigma\|_1/d$ from assumption (5.1), for $p=2$:

$$\begin{aligned}
\mathbb{E}[\|\eta\,\hat{\Sigma}^{1/(p+2)}\|_2^2] &= \tfrac{2C^2}{\rho_3}\sum_{i=1}^{d}\hat{\Sigma}_{ii}^{2/(p+2)} \\
&= \tfrac{2C^2}{\rho_3}\sum_{i=1}^{d}\hat{\sigma}_i \\
&\le \tfrac{2C^2}{\rho_3}\sum_{i=1}^{d}(\sigma_i + \|\sigma\|_1/d) = \tilde{O}(\|\sigma\|_1^2/\rho_3)\ .
\end{aligned}$$

5.4.5 Proof of Theorem 5.1

The output of PLAN can be written as

$$\tfrac{1}{n}\sum_{j=1}^{n} x^{(j)} - \tfrac{1}{n}\sum_{j\in J}\tfrac{\|y^{(j)}\|_2 - C}{\|y^{(j)}\|_2}\,(x^{(j)}-\tilde{\mu}) + \tfrac{1}{n}\eta\,\hat{\Sigma}^{1/(p+2)}$$

Using that $\frac{\|y^{(j)}\|_2 - C}{\|y^{(j)}\|_2} < 1$ for $j \in J$ and the triangle inequality, the ℓ_2 estimation error of PLAN can thus be bounded as

$$\left\|\tfrac{1}{n}\sum_{j=1}^{n} x^{(j)} - \mu\right\|_2 + \tfrac{1}{n}\sum_{j\in J}\|x^{(j)}-\tilde{\mu}\|_2 + \tfrac{1}{n}\|\eta\,\hat{\Sigma}^{1/(p+2)}\|_2$$

The first term is the sampling error, which is $\tilde{O}(\|\sigma\|_2/\sqrt{n})$ with probability at least $1-\beta$. The sum over J was shown in part 2 to be $\tilde{O}\left(\left(k+\sqrt{\tfrac{1}{\rho}}\right)\|\sigma\|_2\right)$, so for $k \le \sqrt{n}$ and using the assumption $n = \tilde{\Omega}(\max(\sqrt{d/\rho}, \rho^{-1}))$, this term is bounded by $\tilde{O}(\|\sigma\|_2/\sqrt{n})$. Finally, the noise term in part 3 is $\tilde{O}(\|\sigma\|_1/(n\sqrt{\rho}))$.

We have shown a more detailed version of Theorem 5.1:

Theorem 5.2. *For sufficiently large $n = \tilde{\Omega}(\sqrt{d/\rho})$,* PLAN *with parameters $k = \sqrt{n}$ and $\rho_1 = \rho_2 = \rho_3 = \rho/3$ is ρ-zCDP. If inputs are independently sampled from a σ-well concentrated distribution, the mean estimate has expected ℓ_2 error $\tilde{O}(1 + ||\sigma||_2/\sqrt{n} + ||\sigma||_1/(n\sqrt{\rho}))$ with probability at least $1 - \beta$, where $\tilde{O}$ suppresses polylogarithmic dependencies on $1/\beta$, n, d, and the bound M on the ℓ_∞ norm of inputs.*

5.4.6 General ℓ_p error

To address the general case we need a definition of well-concentrated that depends on p. For simplicity we assume that p is a positive integer constant.

Definition 5.6. For integer constant $p \geq 1$ consider a distribution $\mathcal{D}$ over $\mathbb{R}^d$, where the ith coordinate has mean μ_i and pth central moment σ_i^p. We say that $\mathcal{D}$ is (σ, p)-*well concentrated* if for any vector $\hat{\sigma} \in \mathbb{R}^d$ with $\sigma_i \leq \hat{\sigma}_i < \left(\sigma_i^{\frac{2p}{p+2}} + \|\sigma^{\frac{2p}{p+2}}\|_1/d\right)^{\frac{p+2}{2p}}$, the following holds for $t > 1$:

$$\Pr_{X\sim\mathcal{D}}\left[\sum_{i=1}^{d} |X_i - \mu_i|^p > t\|\sigma\|_p^p\right] = \exp(-\tilde{\Omega}(t)) \tag{5.4}$$

$$\Pr_{X\sim\mathcal{D}}\left[\sum_{i=1}^{d} \frac{(X_i - \mu_i)^2}{\hat{\sigma}_i^{4/(p+2)}} > t\|\sigma^{p/(p+2)}\|_2^2\right] = \exp(-\tilde{\Omega}(t)) \ . \tag{5.5}$$

Theorem 5.3. *For sufficiently large $n = \tilde{\Omega}(\max(\sqrt{d/\rho}, \rho^{-1}))$,* PLAN *with parameters $k = \sqrt{n}$ and $\rho_1 = \rho_2 = \rho_3 = \rho/3$ is ρ-zCDP. If inputs are independently sampled from a (σ, p)-well concentrated distribution, the mean estimate has expected ℓ_p error*

$$\tilde{O}\left(1 + \frac{||\sigma||_p}{\sqrt{n}} + \frac{\|\sigma\|_{2p/(p+2)}}{n\sqrt{\rho}}\right)$$

with probability at least $1 - \beta$, where $\tilde{O}$ suppresses polylogarithmic dependencies on $1/\beta$, n, d, and the bound M on the ℓ_∞ norm of inputs.

The proof of this theorem follows along the lines of the proof in this section and can be found in Appendix 5.B. In comparison, the standard Gaussian mechanism for ℓ_2 sensitivity $\|\sigma\|_2$ has expected ℓ_p error $\frac{d^{1/p}\|\sigma\|_2}{\sqrt{\rho}}$.

5.5 Examples of well-concentrated distributions

In the following we give examples for some (σ, p)-well concentrated distributions. We start with a general result.

Lemma 5.7. *For integer constant $p \geq 1$, consider a distribution $\mathcal{D}$ over $\mathbb{R}^d$ were the ith coordinate has mean μ_i and standard deviation σ_i such that for $X \sim \mathcal{D}$, the pth moment $\mathbb{E}\left[|X_i - \mu_i|^p\right] \leq K\sigma_i^p$ for some constant K. Then $\mathcal{D}$ is (σ, p)-well concentrated.*

Proof. We first prove the bound (5.4) from Definition 5.6. Sample X from $\mathcal{D}$ and define $Y_i = |X_i - \mu_i|^p - \mathbb{E}[|X_i - \mu_i|^p]$ and note that $\sum Y_i \leq \sum |X_i - \mu_i|^p$. Each Y_i is zero-centered, so we may apply Bernstein's inequality (Lemma 5.11)

$$\Pr\left[\sum_{i=1}^{d} Y_i > t\|\sigma\|_p^p\right] \leq \exp\left(-\frac{t^2\|\sigma\|_p^{2p}/2}{\sum_{i=1}^{d}\mathbb{E}[Y_i^2] + Mt\|\sigma\|_p^p/3}\right).$$

We distinguish two cases depending on which term is dominating the denominator. In the first case,

$$\sum_{i=1}^{d}\mathbb{E}\left[Y_i^2\right] \leq \sum_{i=1}^{d}\mathbb{E}\left[(X_i - \mu_i)^{2p}\right] \leq K\|\sigma^{2p}\|_1 \leq \|\sigma\|_p^{2p},$$

where we applied the triangle inequality. This means that the probability of $\sum |X_i - \mu_i|^p$ to exceed $t\|\sigma\|_p^p$ is $\exp\left(-\Omega\left(t^2\right)\right)$. In the second case,

$$\frac{t^2\|\sigma\|_p^{2p}/2}{Mt\|\sigma\|_p^p/3} = \frac{3t\|\sigma\|_p^p}{2M}.$$

By Chebychev's inequality almost surely $Y_i \leq C\max_i \sigma_i^p \leq \|\sigma\|_p^p$. Thus, the probability is bounded by $\exp(-\Omega(t))$ in this case.

For the second property (5.5), note that $\sum(X_i - \mu_i)^2/\hat{\sigma}_i^{4/(p+2)}$ is maximized for $\hat{\sigma}_i = \sigma_i$. For this choice, the calculations are analogous to the ones above. □

We will now proceed with studying the Gaussian distribution for ℓ_2 error and a sum of Poisson trials for ℓ_1 error. These are the distributions that we will empirically study in Section 5.7.

5.5.1 Gaussian data

Lemma 5.8. *Let $p = 2$ and fix $\mu \in \mathbb{R}^d$ and $\Sigma \in \mathbb{R}^{d\times d}$. The multivariate normal distribution $\mathcal{N}(\mu, \Sigma)$ is $(\sigma, 2)$-well concentrated.*

Before presenting the proof, we remark that the independent case with diagonal covariance matrix Σ can easily be handled by Chernoff-type bounds. Furthermore, Lemma 5.7 holds for $\mathcal{N}(\mu, \Sigma)$ for diagonal covariance matrix. In the lemma, we handle the general case of (non)-diagonal covariance matrices, thus allowing for dependence among the variables.

Proof. Consider property (5.2) of Definition 5.5 and fix any $i \in \{1, \ldots, d\}$. Let $\sigma_i^2 = \Sigma_{ii}$ and sample $X_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$. Since X is normally distributed, for all $t > 0$ it holds that [MU05, Theorem 9.3]

$$\Pr[|X - \mu_i| \geq t] \leq 2\exp(-t^2/(2\sigma_i^2)).$$

Fix a value $t > 0$ and let $t' = t\ln(d)\sigma_i^2$. We proceed to bound the deviation as follows:

$$\begin{aligned}\Pr[(X-\mu_i)^2 \geq t'] &= \Pr[|X-\mu_i| \geq \sqrt{t'}] \\ &\leq 2\exp\left(-\left(t\ln(d)\sigma_i^2\right) / \left(2\sigma_i^2\right)\right) \\ &= 1/d\exp(-\tilde{\Omega}(t)).\end{aligned}$$

Consider the case that we sample $X_1, \ldots, X_d$ from $\mathcal{N}(\mu, \Sigma)$. By a union bound over $X_1, \ldots, X_d$, we may assume that with probability at least $1 - \exp\left(-\tilde{\Omega}(t)\right)$, for all $i \in \{1, \ldots, d\}$ we have $(X_i - \mu_i)^2 \leq t\ln(d)\sigma_i^2$, which implies that their sum is at most $t\ln(d)\|\sigma\|_2^2 = \tilde{O}(t\|\sigma\|_2^2)$.

Next, consider property (5.3) of Definition 5.5. Sample $X \sim \mathcal{N}(\mu_i, \sigma_i^2)$ and consider the transformation $(X - \mu_i)^2/\sigma_i$. Setting $t' = t\ln(d)\sigma_i$, the same calculations as above show that with probability at most $1/d\exp(-\tilde{\Omega}(t))$, $(X - \mu_i)^2/\sigma_i$ is larger than $t\ln(d)\sigma_i$.

Consider the case that we sample X from $\mathcal{N}(\mu, \Sigma)$. Again using a union bound, with probability at least $1 - \exp(-\tilde{\Omega}(t))$, all scaled values are within $t\ln(d)\sigma_i$. Conditioned on this, $\sum_{i=1}^d (X_i - \mu_i)^2/\hat{\sigma}_i \leq t\ln(d)\|\sigma\|_1$, which finishes the proof. □

5.5.2 Binary data

We next consider binary strings of length d in which bit i is set independently and at random with probability q_i. Lemma 5.7 is not applicable in this case, because the pth moment is roughly equal to $\max\{q_i, 1-q_i\}$.

Lemma 5.9. *Let $d \geq 1$ be an integer, and let $\mathcal{D}_{binary}$ be the distribution over length-d binary strings such that the bit in position i is set with probability q_i. If for each $i \in \{1, \ldots, d\}$, $\sigma_i^2 = q_i(1-q_i) \geq 1/d^{2/5}$, then $\mathcal{D}_{binary}$ is $(\sigma, 1)$-well concentrated.*

Proof. Since we assumed in Section 5.4 that $\sigma_i \geq 1$, we consider the mapping $x \mapsto d^{1/5}x := \bar{x}$. Consider Property (5.4) of Definition 5.6. Since $\bar{\sigma}_i^2 \geq 1$, $\|\bar{\sigma}\|_1 \geq d$. Using a generalized Chernoff bound (Lemma 5.13) we conclude that $\sum |d^{1/5}(X_i - \mu_i)|$ exceeds $t\|\bar{\sigma}\|_1$ with probability at most

$$\exp\left(-(t\|\bar{\sigma}\|_1)^2 / \left(2d^{2/5}d\right)\right) = \exp\left(-\Omega(t^2)\right).$$

Next, consider Property (5.5). Define $Y_i = \left(d^{1/5}(X_i - q_i)\right)^2 / \sigma_i^{4/3}$. Note that Y_i takes values in an interval of length $d_i \leq d^{2/5}$. $|\sum Y_i - \mathbb{E}[\sum Y_i]|$ exceeds $t\|\bar{\sigma}^{2/3}\|_1$ with probability at most

$$\exp\left(-\frac{t^2\|\bar{\sigma}^{2/3}\|_1^2}{2\sum_{i=1}^d d_i^2}\right) \leq \exp\left(-\frac{t^2\|\bar{\sigma}^{2/3}\|_1^2}{2d^{9/5}}\right) \leq \exp\left(-t^2 d^{1/5}/2\right)$$
$$= \exp\left(-\Omega(t^2)\right).$$

□

5.6 Generic Bounds in the Absence of Variance estimates

Algorithm 17 requires as input estimates $\hat{\sigma}^2$ on the coordinate-wise variances. If the input is σ-well concentrated, Theorem 5.2 provided bounds on the expected ℓ_2 error of the algorithm. In this section, we consider the case that no such estimates are known and we run the algorithm without the scaling step. This is similar in spirit to using the "shifted-clipped-mean estimator" of Huang et al. [HLY21]. The following theorem shows that even without carrying out the random rotation (see Section 3.3 in [HLY21]), we match their bounds up to constant terms in expectation. This result makes their algorithm useful in settings where a

random rotation impacts performance negatively, e.g., when vectors are sparse.

Let $D \subseteq \mathbb{R}^d$ be the collection of n vectors $x^{(1)}, \ldots, x^{(n)}$. Let $w(D) = \max_{x,y \in D} \|x - y\|_2$ be the diameter of the dataset. For simplicity we assume that $w(D) \geq 1$ and $M = O(w(D))$. In comparison to the distributional setting studied before, there is no sampling error involved in mean estimation and we only measure clipping error and the error due to noise.

Theorem 5.4. *Let $d \geq 1, \rho > 0$, and D be of size $n = \tilde{\Omega}\left(\sqrt{\frac{d}{\rho}}\right)$. If Algorithm 17 is run with $k = \tilde{\Theta}\left(\sqrt{\frac{d}{\rho}}\right)$, $\rho_1 = \rho_2 = \rho_3 = \rho/3$, and $\hat{\sigma}_i^2 = 1$ for all $i \in \{1, \ldots, d\}$ then the expected ℓ_2 error due to clipping and noise is*

$$\tilde{O}\left(\sqrt{\frac{d}{\rho}}\frac{w(D)}{n}\right) \quad .$$

Before proceeding with the proof, we introduce some helpful notation.

Definition 5.7. Let $\mu \in \mathbb{R}^d$ be the coordinate-wise median of D. For $0 \leq \alpha < 1/2$, we say that $\tilde{\mu} \in \mathbb{R}^d$ is α-good if each $\tilde{\mu}_i$ has rank error at most αn from μ_i.

Lemma 5.10. *Given $0 \leq \alpha < 1/2$, let $\tilde{\mu}$ be α-good. For each $x^{(j)} \in D$, $1 \leq j \leq n$, $\|x^{(j)} - \hat{\mu}\|_2^2 \leq \frac{w(D)^2}{1/2-\alpha}$.*

Proof. Fix $x^{(j)}$ and compute

$$\begin{aligned}
(n-1)w(D)^2 &\geq \sum_{j \neq j'} \|x^{(j)} - x^{(j')}\|_2^2 \\
&= \sum_{i=1}^{d} \sum_{j \neq j'} \left(x_i^{(j)} - x_i^{(j')}\right)^2 \\
&\geq \sum_{i=1}^{d} (1/2 - \alpha)(n-1) \left(x_i^{(j)} - \hat{\mu}_i\right)^2 \\
&= (1/2 - \alpha)(n-1)\|x^{(j)} - \hat{\mu}\|_2^2.
\end{aligned}$$

The lemma follows by re-ordering terms. □

Proof of Theorem 5.4. We instantiate PrivQuantile as the binary search based method of [HLY21] with $T = \log(Mn\sqrt{\rho})$. Fix some value for α,

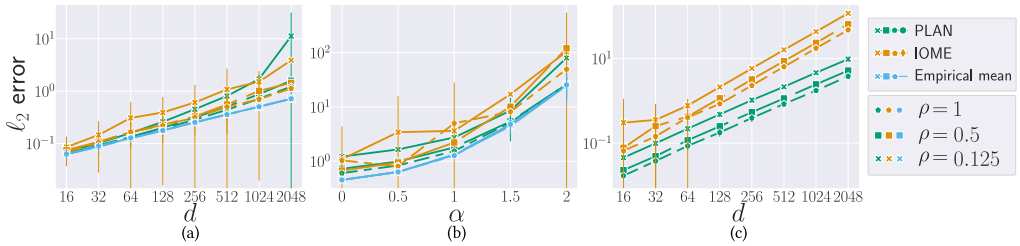


Figure 5.2: ℓ_2 error for synthetic Gaussian data when varying (a) dimensions with data without a skew, (b) skewness of the variances, and (c) dimensions for skewed data — note that we compute error relative to the empirical mean rather than the statistical mean in this experiment as sampling error dominates in this setting. Also notice the different scales on the y-axis.

say 1/3. By Lemma 5.5 the coordinate-wise median computed on Line 3 of Algorithm 17 is within ℓ_2 distance $\sqrt{d/\rho}/n$ of an α-good point with high probability. We assume for simplicity that $\tilde{\mu}$ is itself α-good as an additional error of $\sqrt{d/\rho}/n$ is dominated by the error from clipping and noise.

Since $\tilde{\mu}$ is α-good, the maximum length of a shifted vector $x^{(j)} - \tilde{\mu}$ is $O(w(D))$, so $C = O(w(D))$ on Line 6 of Algorithm 17. As in Section 5.4.4, the expected ℓ_2 error due to noise is at most $1/n \cdot \sqrt{2C^2(\sum_{i=1}^{d} \hat{\sigma}_i)/\rho} = \tilde{O}\left(\sqrt{\frac{d}{\rho}}\frac{w(D)}{n}\right)$.

In the same way as in the calculations carried out in Section 5.4.3, we can bound the error due to clipping for all vectors $\|y^{(j)}\|_2 > C$. Setting $k = \tilde{\Theta}(\sqrt{d/\rho})$ shows that this clipping error is

$$\tilde{O}\left(\sqrt{d/\rho} \cdot w(D)/n\right).$$

□

5.7 Empirical Evaluation

To put PLAN's utility into context, we measure error in diverse experimental settings. We use the empirical mean as a baseline, since it reflects an inevitable lower bound, i.e. the sampling error $\tilde{O}(\|\sigma\|_2/\sqrt{n})$. Additionally we compare PLAN to INSTANCE-OPTIMAL MEAN ESTIMATION (IOME) [HLY21], which has been shown to perform at least as good as

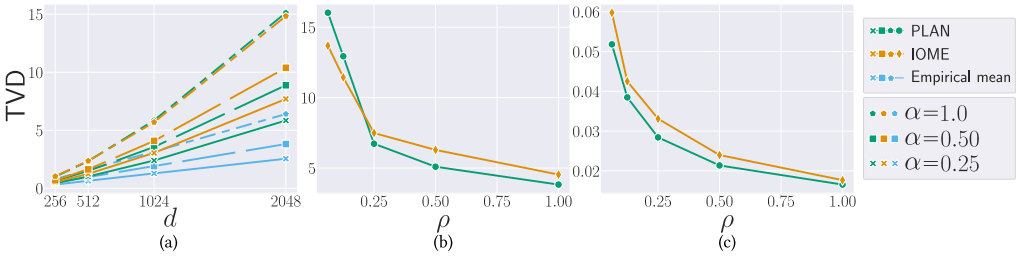


Figure 5.3: (a) Synthetic binary data, varying the ratio of 0s to 1s (b) Kosarak dataset (c) POS dataset

CoinPress [BDKU20] in empirical settings [HLY21] hence representing the current state-of-the-art for differentially private mean estimation. Since PLAN works for (σ, p)-well concentrated distributions, we evaluate accuracy for Gaussian and binary data, representing ℓ_2 and ℓ_1 error, respectively. We run our experiments with synthetic data as input. For the binary case, we also evaluate our error on the Kosarak dataset [BKT18] which represents user visits (or, conversely, non-visits) to webpages, as well as the Point of Sale (POS) dataset[1] which represents user purchases.

5.7.1 Implementation

We evaluate the empirical accuracy of PLAN by instantiating Algorithm 17 in Python 3. Our implementation contains two different instantiations of PLAN: one version for binary data ($p = 1$), and one version for data from multivariate Gaussian distributions ($p = 2$). The pseudocode for both instantiations of PLAN is shown in Listing 5.1. Both instances use the PrivQuantile search by Kaplan et al. [KSS22]. The implementation of IOME uses the original source code from Huang et al. [HLY21].

[1] https://github.com/cpearce/HARM/blob/master/datasets/BMS-POS.csv

```
1  def PLAN(data, n, d, M, p, rho, beta) {
2   rho1, rho2, rho3 = divideBudget(rho)
3   mu = center(M, rho1*0.25, data, beta/3)
4   std = estimateStd(M, rho1*0.75, data, beta/3)
5   scaleFactors = std**(-1/(p + 2))
6   y = (data-mu) * scaleFactors #coordinate-wise
7   k = sqrt(n) + rankError(M, n, d, rho2, beta/3)
8   z = clip(M, y, rho2, (n-k)/n)
9   return ((z+noise(p, rho3))/scaleFactors)+mu
10 }
```

Listing 5.1: Pseudocode for the PLAN instantiation for n vectors in $\mathbb{R}^d$, with each coordinate being in the range $[-M, M]$, targeting the expected ℓ_p error with privacy budget ρ and a failure probability of at most β. Note that budget is spent on estimating the standard deviation unlike in Algorithm 17 where $\hat{\sigma}^2$ is an input parameter.

Estimating σ^2. Note that Algorithm 17 assumes an estimate of the variances as input. In the absence of public knowledge, these parameters have to be estimated on the actual data in a differentially private way, as mentioned in Listing 5.1.

In the Gaussian case, given $X, Y \sim \mathcal{N}(\mu_i, \sigma_i^2)$, $\mathbb{E}[(X-Y)^2/2] = \mathbb{E}[X^2] - \mathbb{E}[X]^2 = \sigma_i^2$. Since $(X-Y)^2/2$ follows a generalized Chi-squared distribution, we use PRIVQUANTILE for each coordinate to differentially privately estimate the median $\tilde{\mu}_i$ of a sequence of values $(X-Y)^2/2$, and estimate the mean as $\tilde{\mu}_i/(9/7)^3$. In the binary case where each coordinate is 1 with probability q_i, we estimate the variance by private estimation of the mean $\tilde{q}_i$ using the Gaussian mechanism with ℓ_2-sensitivity $1/n$, and use $\hat{\sigma}_i^2 = \tilde{q}_i(1-\tilde{q}_i)$. In both cases, we regularize the estimate $\hat{\sigma}$ on the standard deviation by adding $\|\hat{\sigma}\|_1/d$ to each coordinate.

In Section 5.C, we generalize the estimator on Gaussian data to distributions with bounded fourth moment and provide more details on the empirical evaluation.

Bounding the clipping universe. Having an estimate on σ^2 provides an opportunity to trim the universe used when searching for the clipping radius (Algorithm 17 Line 6). Specifically, instead of using $M\sqrt{d}$ as the upper bound to cover the entire universe, we use the tighter bound $\sqrt{\log(d)\log(1/\beta)}\|\sigma\|_1$.

5.7.2 Experiment design

Parameter input space. The following parameters need to be chosen for each execution of PLAN: the universe M, the ℓ_p error norm, and the privacy budget ρ as well as the partitioning of ρ into ρ_1, ρ_2, ρ_3. Since IOME uses a binary search for their quantile selection, the amount of steps to use also needs to be chosen. IOME sets the amount of steps to 10 by default, but an empirical investigation shows that this value is too low for many of our settings — the binary quantile search ends early which causes inaccurate results. To level the playing field, we use 20 steps to ensure that IOME does not suffer any disadvantages from the binary quantile search failing.

Input data. We will use both synthetic and real-world datasets to evaluate PLAN. When generating synthetic data, the following parameters need to be chosen: the dataset size n, the dimensionality d, the means μ, and the variances σ^2. Since PLAN and IOME both ignore potential correlations in data, we use covariance matrices of the form $\Sigma = \text{diag}(\sigma^2)$.

Gaussian data

To show the effectiveness of PLAN on Gaussian data, we design three diverse experiments. The first experiment (Gaussian A) reflects the parameter settings used in previous work by Huang et al. [HLY21] where data has no skew, which is the case IOME is intended for. The second experiment (Gaussian B) simulates data ranging from no to significant skew across dimensions, showcasing how PLAN improves with increasing skew. Finally, the third experiment (Gaussian C) highlights how PLAN scales as dimensionality increases for data with a skew. For each experiment, we vary ρ between 1 and 0.125 to show how accuracy scales in higher and lower privacy regimes. We summarize the settings used in the experiments in Appendix 5.D. All experiment settings are run 50 times for each algorithm. IOME is run with 20 steps for the binary quantile selection. Note that ρ_1 is split between recentering and variance prediction for PLAN.

Budget division. Our algorithm needs to perform two preprocessing steps: estimating μ for re-centering, and estimating σ^2 for scaling the noise. We fix the initial estimation of μ and σ^2 to use 25% of the total privacy budget — the same proportion used for preprocessing as in

Huang et al. [HLY21]. In the same spirit, we set the budget to determine the clipping threshold (ρ_2) to 25% of the remaining budget, and use the larger part (ρ_3) for the Gaussian noise.

Choosing valid settings. Just like for IOME, M needs to be set such that μ is within the universe. We will use two different approaches to set M: the approach from [HLY21] ($M = \sqrt{50}d$), and a more pessimistic approach where we assume all values have the worst-case standard deviation across all dimensions, and create more leeway by scaling with a constant ($M = 100d \max\{\sigma\}$).

Additionally, the rank error needs to be tuned such that PRIVQUANTILE search can be expected to return a quantile close to the requested one. Since PLAN calls PRIVQUANTILE multiple times, PLAN needs to tolerate the worst-case rank error for all calls. We set n such that the rank error is at most $0.1n$ for each value of ρ.

Gaussian A: *no skew*. To show that PLAN performs comparatively to IOME we run it on data where variance is the same across all dimensions. In this setting we expect PLAN to perform similarly to IOME. We reuse the experiment settings used by Huang et al. [HLY21] for a fair comparison.

Gaussian B: *varying skewness*. To show how PLAN improves as the input data's skew increases, we vary the skewness of the variance. We introduce a parameter α, and simulate a Zipfian like skew to the data, and set the variances $\sigma^2 = ((d/d)^\alpha, (d/(d-1))^\alpha, \ldots, (d/1)^\alpha)$ for $\alpha \in \{0, 0.5, \ldots, 2\}$. In this setting we expect PLAN to outperform IOME for $\alpha > 0$.

Gaussian C: *varying dimensionality*. To show how PLAN's advantage scales compared to IOME, we vary dimensionality as we expect an improvement up to a factor $\sqrt{d}$. Since PLAN's advantage is based on data having a skew, we set $\sigma^2 = (d/d)^\alpha, (d/(d-1))^\alpha, \ldots, (d/1)^\alpha$, where $\alpha = 2$. Note that PLAN's improvement is in *noise error*, as sampling error is unavoidable — we compute the error relative to the empirical mean in this experiment to showcase the difference in noise error.

Binary data

To diversify our experimental scope, we consider binary data represented by n bitvectors of length d in which each bit $i, 1 \leq i \leq d$, is set independently to 1 with probability q_i. We usually think about these bitvectors as sets representing a selection of items from $\{1, \ldots, d\}$. To vary the error measure, we focus on the ℓ_1 error. This is akin to computing the total

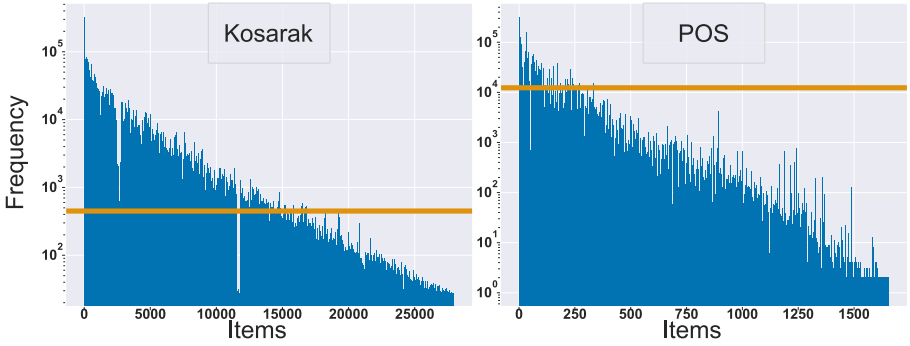


Figure 5.4: Histogram for Kosarak (left) and POS (right). The orange line is the smallest allowed variance according to Lemma 5.9 which we clip to.

variation distance (TVD) $1/2\|x - y\|_1$, but we avoid the normalization of x and y to have unit ℓ_1 norm.

We design three experiments: the first (Binary) varies the skewness in the probabilities to make controlled experiments on the accuracy of PLAN. The two remaining experiments (Kosarak, POS) use real-world datasets that naturally exhibit skew between coordinates.

Binary: *Varying skewness.* This experiment follows the same design principle as Gaussian B: to show how skewness affects the performance of PLAN. Given n, d, and ρ, we choose two probabilities $q_1 = 0.5$ (high variance) and $q_2 = 0.01$ (low variance). Given $\alpha \in [0, 1]$, we sample the first $\lceil \alpha d \rceil$ bits with probability q_1 each, and the remaining positions with probability q_2. The low variance setting is slightly below the minimum threshold of $1/d^{2/5}$ discussed in Lemma 5.9 to test the robustness of our implementation. We clip all estimated variances to $1/d^{2/5}$ from below.

Kosarak: *Website visits.* The *Kosarak dataset*[2] represents click-stream data of a Hungarian news portal. There are $n = 75\,462$ users and a collection of $d = 27\,983$ websites. In total, users clicked on 4 194 414 websites (each user clicked on 55.6 websites on average), and there is a large skew between the websites, see Figure 5.4.

POS: *Shopping baskets.* The *POS dataset* contains merchant transactions on $d = 1\,657$ categories from $n = 515\,596$ users. In total, there are 3 367 019 transactions (around 6.5 on average per user). Again, there is large skew in the different categories, see Figure 5.4. The dataset is particularly challenging because the minimum variance $d^{-2/5}$ (cf. Lemma 5.9) has to be clipped on many coordinates.

[2] `http://fimi.uantwerpen.be/data/`

5.7.3 Results

For the Gaussian case, we ran our experiments using Python 3.11.3 on a MacBook Pro with 24GB RAM, and the Apple M2 chip (8-core CPU). For the binary case, we had to run the experiments on a more powerful machine to support IOME on the real-world datasets. While Kosarak and POS are sparse datasets, IOME requires that the entire dataset (not just the sparse representation) is loaded into memory to perform a random rotation the algorithm uses as a preprocessing step. As a consequence, we ran the binary experiments using Python 3.6.9 on a machine with 512GB RAM, on a Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz (56-core CPU). Even on this more powerful machine, a single run of IOME took at least 26 minutes on Kosarak, and at least 9 minutes on POS. In comparison, PLAN spent an average of 12, and 9 seconds running on Kosarak and POS, respectively.

Gaussian data

All results are shown in Figure 5.2. Figure 5.2 (a) shows Gaussian A, where PLAN and IOME have comparable accuracy until $d = 2048$, when PLAN performs worse as we reach a setting where the assumptions on rank error no longer hold. This is expected behavior as PLAN spends some additional budget estimating σ^2 in comparison to IOME.

Figure 5.2 (b) shows Gaussian B, where PLAN performs better than IOME for $\alpha > 0$. For $\alpha = 0$ the error between PLAN and IOME is similar. This is expected behavior, as $\alpha = 0$ represents the same input data as in Gaussian A. Notice how PLAN approaches the empirical mean as α grows for both $\rho = 0.5$ and $\rho = 1$.

Figure 5.2 (c) shows Gaussian C, where we compare against the empirical mean since sampling error is larger than the noise error for PLAN in this case. As expected, PLAN increases its advantage over IOME as d grows.

Binary data

All results are shown in Figure 5.3. Figure 5.3 (a) shows Binary, where PLAN has an advantage over IOME for $\alpha < 1$ which increases as α decreases. This is the expected behavior, as PLAN is able to exploit the skew in variance whereas IOME treats every dimension the same.

Figure 5.3 (b) shows Kosarak. As we can see, PLAN outperforms IOME for sufficiently large values of ρ. For small ρ ($\rho \leq 0.125$), PLAN is running

in an invalid setting — our assumptions on rank error are not fulfilled in these cases.

Figure 5.3 (c) shows POS. PLAN has a slight advantage compared to INSTANCE-OPTIMAL MEAN ESTIMATION in this case, which decreases as ρ grows.

5.8 Related Work

Our work builds on concepts from multiple areas within the literature on differential privacy. We provide an overview of the most closely related work.

Statistical private mean estimation. There is a large, recent literature on statistical estimation for d-dimensional distributions under differential privacy, mainly focusing on the case of Gaussian or subgaussian distributions (see e.g. [AKT+22, AL22, BDKU20, BGS+21, BHS23, DFM+20, DHK23, HKM22, KV18, KLSU19, KMS+22, KMV22]). The error on mean estimates is generally expressed in terms of Mahalanobis distance, which is natural if we want the error to be preserved under affine transformations. Some of these efficient estimators are even *robust* against adversarial changes to the input data [AKT+22, KMV22]. Other estimators work even for rather heavy-tailed distributions [KSU20]. What all these estimators have in common is that the nominal dimension d influences the privacy-utility trade-off such that higher-dimensional vectors have a worse trade-off. To our best knowledge, the algorithm among these that has been shown to work best in practical (non-adversarial) settings is the COINPRESS algorithm of Biswas, Dong, Kamath, and Ullman [BDKU20].

Adapting to the data. The best private mean estimation algorithms are near-optimal for worst-case d-dimensional distributions in view of known lower bounds [CWZ21]. However, it is natural to consider ways of improving the privacy-utility trade-off whenever the input distribution has some structure. One way of going beyond the worst case is by privately identifying low-dimensional structure (see e.g. [ADK+19, DTTZ14, HP14, SS21]). Such methods effectively reduce the mean estimation problem to an equivalent problem with a dimension smaller than d. However, we are not aware of any work showing this approach to be practically relevant for mean estimation.

Another approach for adapting to the data is *instance optimality*, introduced by Asi and Duchi [AD20] and studied in the context of mean estimation by Huang et al. [HLY21] who use ℓ_2 error (or mean square error) as the utility metric. The goal is optimality, i.e. matching lower bounds, for a class of inputs with a given diameter but no further structure. Huang et al. [HLY21] found that their private mean estimation algorithm often has smaller error than CoinPress in practice. Because of this, and since they also aim to minimize an ℓ_p error, this algorithm was chosen as our main point of comparison.

Neither of the mentioned approaches takes *skew* in the data distribution into account, so we believe this is a novel aspect of our work in the context of mean estimation. However, we mention that privacy budgeting in skewed settings has recently been studied in the context of multi-task learning Krichene, Jain, Song, Sundararajan, Thakurta, and Zhang [KJS+23]. Also, the related setting of mean estimation with heterogeneous data (where the sensitivity with respect to different clients' data can differ) was recently studied by Cummings, Feldman, McMillan, and Talwar [CFMT22].

Clipping. An important aspect of private mean estimation for unbounded distributions, in theory and practice, is how to perform *clipping* to reduce the sensitivity. This has in particular been studied in the context of differentially private stochastic gradient descent [MRTZ18, PSY+19, ATMR21, BWZK22]. Though clipping introduces bias, Kamath, Mouzakis, Regehr, Singhal, Steinke, and Ullman [KMR+23] have shown that this is unavoidable without additional assumptions.

Huang et al. [HLY21] used a clipping method designed to cut off a carefully chosen, small fraction of the data points. The clipping done in PLAN follows the same pattern, though it is applied only after carefully scaling data according to the coordinate variances. Thus, it corresponds to clipping to an axis-aligned ellipsoid.

To formally bound clipping error one can either express the error in terms of the diameter of the dataset or analyze the error under some assumption on the data distribution. Both approaches are explored in Huang et al. [HLY21], but in this chapter we have chosen to focus on the latter.

5.9 Conclusion and future work

We introduce Private Limit Adapted Noise (PLAN), a family of algorithms for differentially private mean estimation of d-dimensional data. PLAN exploits skew in data's variance to achieve better ℓ_p error. In the case of ℓ_2 error we achieve a particularly clean bound, namely error proportional to $\|\sigma\|_1$. This is never worse than the error of $\sqrt{d}\|\sigma\|_2$ obtained by previous methods and gives an improvement up to a factor of up to $\sqrt{d}$ when σ is skewed. While the privacy guarantees hold for any input, the error bounds hold for independently sampled data from distributions that follow a well-defined assumption on concentration.

Finally, we implement two PLAN instantiations and empirically evaluate their utility. Practice follows theory — PLAN outperforms the current state-of-the-art for skewed datasets, and is able to perform competitively for datasets without skewed variance. To aid practitioners in implementing their own PLAN, we summarize some practical advice based on our lessons learned.

Advice for practitioners. When implementing a PLAN instantiation, practitioners should pose the following questions:

1. Is there a suitable estimator for the variances σ_i^2 of the data distribution?
2. Can a tighter bound on the clipping universe ($M\sqrt{d}$) be used?
3. How robust is PLAN for my given settings, i.e., will the rank error be too high and cause PLAN to fail?

We give examples of how to answer these questions in our evaluation. To answer Question 1, we derive *private* variance estimators tuned to the data distribution. As for Question 2, when the distribution is (σ, p)-well concentrated, much better bounds on the universe size can be derived by using Assumption (5.5) in Definition 5.6. Finally, answering Question 3, robustness must carefully be evaluated using the assumption on minimum ρ values and maximum dimensionality d. Both parameters in conjunction give minimum requirements to the required sample size n.

5.A Useful statements from probability theory

Lemma 5.11 (Bernstein's inequality)**.** *Let $X_1, \ldots, X_n$ be independent zero-mean random variables. Suppose that $|X_i| \leq M$ almost surely for all i. Then for all $t > 0$,*

$$\Pr\left[\sum_{i=1}^{n} X_i \geq t\right] \leq \exp\left(-\frac{t^2/2}{\sum_{i=1}^{n} \mathbb{E}[X_i^2] + Mt/3}\right).$$

Lemma 5.12 ([Win12, Equation (18)])**.** *The pth absolute moment of a zero-centered Gaussian distribution for any $p > 0$ is*

$$\mathbb{E}\left[\left|\mathcal{N}\left(0, \sigma^2\right)\right|^p\right] = \sigma^p \cdot \frac{2^{p/2}\Gamma\left(\frac{p+1}{2}\right)}{\sqrt{\pi}}.$$

Lemma 5.13 (Generalized Chernoff-Hoeffding Bound [DP09])**.** *Let $X := \sum_{1 \leq i \leq n} X_i$ where $X_i, 1 \leq i \leq n$ are independently distributed in $[a_i, b_i]$ for $a_i, b_i \in \mathbb{R}$. Then for all $t > 0$*

$$\Pr[|X - \mathbb{E}[X]| \geq t] \leq 2\exp\left(\frac{-t^2/2}{\sum_i (a_i - b_i)^2}\right).$$

5.B Proof of Theorem 5.3

Theorem 5.5. *For sufficiently large $n = \tilde{\Omega}(\max(\sqrt{d/\rho}, \rho^{-1}))$, PLAN with parameters $k = \sqrt{n}$ and $\rho_1 = \rho_2 = \rho_3 = \rho/3$ is ρ-zCDP. If inputs are independently sampled from a (σ, p)-well concentrated distribution, the mean estimate has expected ℓ_p error*

$$\tilde{O}\left(1 + \frac{||\sigma||_p}{\sqrt{n}} + \frac{\|\sigma\|_{2p/(p+2)}}{n\sqrt{\rho}}\right)$$

with probability at least $1 - \beta$, where $\tilde{O}$ suppresses polylogarithmic dependencies on $1/\beta$, n, d, and the bound M on the ℓ_∞ norm of inputs.

Proof. We proceed in the same three steps as in the proof of Theorem 5.2 in Section 5.4.

By Lemma 5.6, with probability at least $1 - \beta$, all $|\mu_i - \tilde{\mu}_i| \leq \sigma_i$. In this case, $\|\mu - \tilde{\mu}\|_p = O(\|\sigma\|_p)$.

Let $J = \{j \in \{1, \ldots, n\} \mid \|y^{(j)}\|_2 > C\}$ denote the set of indices of vectors affected by clipping in PLAN. By the triangle inequality and assuming the bound of part 1 holds,

$$\begin{aligned}\sum_{j \in J} \left\| (x^{(j)} - \tilde{\mu}) \right\|_p &\leq \sum_{j \in J} \left\| (x^{(j)} - \mu) \right\|_p + |J| \cdot \|\mu - \tilde{\mu}\|_p \\ &\leq \sum_{j \in J} \left\| (x^{(j)} - \mu) \right\|_p + 2|J| \cdot \|\sigma\|_p \ .\end{aligned}$$

By assumption (5.4) the probability that $\|x^{(j)} - \mu\|_p^p > t\|\sigma\|_p^p$ is exponentially decreasing in t, so setting $t = \tilde{\Omega}(\log^p(n/\beta))$ we have $\|x^{(j)} - \mu\|_p = \tilde{O}(\|\sigma\|_p)$ for all $j = 1, \ldots, n$ with probability at least $1 - \beta$. Using the triangle inequality again we can now bound

$$\sum_{j \in J} \|x^{(j)} - \mu\|_p = \tilde{O}(|J| \cdot \|\sigma\|_p) \ .$$

The same line of argument as in Section 5.4.3 shows that $|J| = \tilde{O}(k + \sqrt{1/\rho})$. Thus, the clipping error can be bounded by $O((k + \sqrt{1/\rho})\|\sigma\|_p)$, and setting $k = n^{-1/2}$ balances the clipping error with the sampling error $\frac{\|\sigma\|_p}{\sqrt{n}}$.

Lastly, we consider the error due to noise. First, we find a bound on the clipping threshold C. By the triangle inequality, we may bound

$$\begin{aligned}\|y^{(j)}\|_2 &= \left\| (x^{(j)} - \tilde{\mu})\, \hat{\Sigma}^{-\frac{1}{p+2}} \right\|_2 \\ &\leq \left\| (x^{(j)} - \mu)\, \hat{\Sigma}^{-\frac{1}{p+2}} \right\|_2 + \left\| (\mu - \tilde{\mu})\, \hat{\Sigma}^{-\frac{1}{p+2}} \right\|_2\end{aligned}$$

The ith coordinate of the first vector $(x^{(j)} - \mu)\, \hat{\Sigma}^{-1/(p+2)}$ equals $(x_i^{(j)} - \mu_i)/\hat{\sigma}_i^{2/(p+2)}$, so assumption (5.5) implies that the length of the first vector is $\tilde{O}(\sqrt{\|\sigma^{2p/(p+2)}\|_1})$ with high probability. By using that the ith coordinate of $|\mu - \tilde{\mu}| \leq \sigma$, $(\mu - \tilde{\mu})\, \hat{\Sigma}^{-1/(p+2)}$ has absolute value $\tilde{O}(\sqrt{\|\sigma^{2p/(p+2)}\|_1})$ as well. The clipping value of C is bounded by the maximum length of a vector $\|y^{(j)}\|_2$, and thus $C^2 = \tilde{O}(\|\sigma^{2p/(p+2)}\|_1)$, with probability at least $1 - \beta$.

The scaled noise vector $\eta\,\hat{\Sigma}^{1/(p+2)}$ has distribution $\mathcal{N}(0, \frac{2C^2}{\rho_3}\hat{\Sigma}^{2/(p+2)})$. Using $\hat{\sigma}_i < \left(\sigma_i^{\frac{2p}{p+2}} + \|\sigma^{\frac{2p}{p+2}}\|_1/d\right)^{\frac{p+2}{2p}}$ and Lemma 5.12, we conclude

$$\begin{aligned}\mathbb{E}[\|\eta\,\hat{\Sigma}^{1/(p+2)}\|_p^p] &= O\left(\frac{2^p C^p}{\rho_3^{p/2}}\sum_{i=1}^{d}\hat{\Sigma}_{ii}^{p/(p+2)}\right)\\ &= O\left(\frac{2^p C^p}{\rho_3^{p/2}}\sum_{i=1}^{d}\hat{\sigma}_i^{2p/(p+2)}\right)\\ &= \tilde{O}\left(\frac{2^p\|\sigma\|_{2p/(p+2)}^p}{\rho_3^{p/2}}\right)\quad.\end{aligned}$$

The result of Theorem 5.3 is achieved by putting together the different error terms as in the proof of Theorem 5.2.

□

5.C Algorithms for Variance Estimation

While PLAN (Algorithm 17) assumes that estimates on the standard deviations are known, such estimates have to be computed in a differentially private manner. Two such ways were described in Section 5.7 and we will provide more details and empirical results in this section.

We remark that the standard attempt to estimate the variance from a mean estimate $\tilde{\mu}$ is

$$\left(1/n\sum_{i=1}^{n}x_i^2\right) - \tilde{\mu}^2$$

If $x_i \in [-M, M]$, the sensitivity of this function is M^2/n, which, depending on the application, means that too much noise must be added.

5.C.1 A Generic Variance Estimation Algorithm

Given a distribution $\mathcal{D}$ with mean μ and variance σ^2, we showed in Section 5.7 that for $X, Y \sim \mathcal{D}$, $\mathbb{E}\left[(X-Y)^2/2\right] = \sigma^2$. Algorithm 18 is a generalization of the approach used for Gaussian in Section 5.7. For Gaussian data, we made use of the fact that $(X-Y)^2/2$ is χ_2 distributed and it is well-known how to translate an approximate median to an approximate mean.

Algorithm 18: VarianceEstimate

Input: Samples $x^{(1)}, \dots, x^{(n)} \in \mathbb{R}$ from $\mathcal{D}$
Parameters: M, ρ, k
Split $x^{(1)}, \dots, x^{(n)}$ into $n' = \lfloor \frac{n}{2k} \rfloor$ groups $G_1, \dots, G_{n'}$.
For each $i \in \{1, \dots, n'\}$: For $G_i = (\hat{x}_i^{(1)}, \dots, \hat{x}_i^{(2k)})$, let $\hat{y}^{(i)} = \sum_{j=1}^{k} (\hat{x}_i^{(2j)} - \hat{x}_i^{(2j+1)})^2/2$.
return $\tilde{\sigma}^2 \leftarrow \frac{\text{PrivQuantile}_\rho^M(\hat{y}^{(1)}, \dots, \hat{y}^{(n')}, 1/2)}{k}$.

Lemma 5.14. *Let $\beta > 0, \rho > 0$, and $n = \tilde{\Omega}(\rho^{-1/2})$. Let $\mathcal{D}$ be a distribution over $\mathbb{R}$ with mean μ and variance $\sigma^2 \geq 1$. For a constant κ, assume that $\mathbb{E}[(X - Y)^4] \leq \kappa\sigma^4$. With probability at least $1 - \beta$, Algorithm 18 using $k = 16\kappa$ returns an estimate $\hat{\sigma}^2$ such that $\sigma^2/2 \leq \hat{\sigma}^2 \leq 3\sigma^2/2$.*

Proof. Fix a group G_i, $1 \leq i \leq n'$. Since $\mathbb{E}[(X - Y)^2/2] = \sigma^2$, we know that $\mathbb{E}(\hat{y}^(i)) = k\sigma^2$. By Chebychev's inequality,

$$\Pr(|\hat{y}^{(i)} - \mathbb{E}[\hat{y}^{(i)}]| > k\sigma^2/2) \leq \frac{\text{Var}(\hat{y}^{(i)})}{(k\sigma^2/2)^2} \leq \frac{k \cdot \mathbb{E}[(X - Y)^4]}{(k\sigma^2/2)^2} \leq \frac{1}{4},$$

using our assumption on $\mathbb{E}[(X - Y)^4]$ and k.

As in the proof of Lemma 5.6, with probability at least $1 - \exp(-\Omega(n'))$, there are more than $2/3n'$ groups i for which

$$\frac{|\hat{y}^{(i)} - \mathbb{E}[\hat{y}^{(i)}]|}{k} \leq \sigma^2/2.$$

As long as the private quantile selection returns an element $\hat{\sigma}^2$ with rank error at most $n'/6$, $\sigma^2/2 \leq \hat{\sigma}^2 \leq 3/2\sigma^2$. By Lemma 5.4, assuming $n > K\sqrt{1/\rho}\log(1/\beta)\log(M^2) = \tilde{\Omega}(\rho^{-1/2})$, this is true with probability at least $1 - \beta$. □

In contrast to the naïve estimator mentioned above, this estimator has only a logarithmic dependency on the input universe. In contrast to it, it does not improve from increased sample size above a *minimum sample size* in relation to the parameter k that is necessary to guarantee that the rank error is at most $n/(6 \cdot 2k)$.

By running Algorithm 18 on each coordinate independently with target probability $1 - \beta/d$, and using a union bound, we may summarize:

Corollary 5.1. *Let $\beta > 0, \rho > 0$, and $n = \tilde{\Omega}(\sqrt{d/\rho})$. Let $\mathcal{D}$ be a distribution over $\mathbb{R}^d$ with mean μ_i and variance $\sigma_i^2 \geq 1$ on each coordinate $i \in \{1, \ldots, d\}$. For a constant κ, assume that $\mathbb{E}[(X - Y)^4] \leq \kappa\sigma^4$. For each i, with probability at least $1 - \beta$, Algorithm 18 on each coordinate using $k = 16\kappa$ returns an estimate $(\hat{\sigma}_1^2, \ldots \hat{\sigma}_d^2)$ such that $\sigma_i^2/2 \leq \hat{\sigma}_i{}^2 \leq 3\sigma_i^2/2$.*

Why hidden in the $\tilde{\Omega}(.)$ notation, only having $n' = n/(2k)$ to choose a private quantile might be incompatible with the rank error of the input domain and the dimensionality. In this case, we can use more groups to "boost" n', but we need to adjust ρ because each element is potentially present multiple times. To cover variances that are smaller than 1, let $\sigma_{\min}^2$ be a minimum bound on the variance. Then, use PrivQuantile with $T = \Theta(\log(M/\sigma_{\min}^2))$ to quantize the input space in steps of $\sigma_{\min}^2$, which gives a logarithmic depends on $1/\sigma_{\min}^2$.

5.C.2 Variance estimation for Gaussian Data

In the case that we know that the data is distributed as $\mathcal{N}(\mu, \sigma^2)$, we can tune the variance estimation more towards the distribution as follows. Given an estimate $\tilde{\mu}$ on μ, we can estimate the variance as follows:

$$\tilde{\sigma} \leftarrow \textsc{PrivQuantile}_\rho^M(x^{(1)}, \ldots, x^{(n)}, .841) - \tilde{\mu}$$

It is a well-known property of the Gaussian distribution that the .841 quantile is approximately the value $\mu + \sigma$. However, when estimating the quantile privately we have to adjust for the rank error of the quantile selection, so aiming for this exact quantile may be unwise.

We run the following experiment: we sample $n = 4000$ from $\mathcal{N}(10, \sigma^2)$ with $\sigma^2 \in \{0.001, 1\}$. For $\rho \in \{10^{-3}, 10^{-2}\}$ we compare (i) three different methods that use different quantiles of the input data (.75, .841, and .9) to (ii) two different instantiations of Algorithm 18 for $k = 1$ and $k = 4$. Since we know that $(X - Y)^2$ is χ_2 distributed, we use the mean to median transformation and divide the approximate median by $(1 - (2/(9k)))^3$. Each parameter setting is run 100 times and we report on the average relative error $\frac{|\hat{\sigma}^2 - \sigma^2|}{\sigma^2}$. Table 5.1 reports on empirical results for the variance estimation. We summarize that Algorithm 18 is more accurate than direct estimation for both values of k, and guarantees very small relative error even for small σ^2.

ρ	σ^2	**Method**	**Relative error**
0.001	0.001	direct-075	0.350444
	0.001	direct-0841	0.038144
	0.001	direct-09	0.240120
	0.001	general (k=1)	0.035163
	0.001	general (k=4)	**0.019068**
	1	direct-075	0.330807
	1	direct-0841	0.031329
	1	direct-09	0.296298
	1	general (k=1)	0.022601
	1	general (k=4)	**0.010411**
0.01	0.001	direct-075	0.319298
	0.001	direct-0841	0.007646
	0.001	direct-09	0.276414
	0.001	general (k=1)	0.023152
	0.001	general (k=4)	**0.008120**
	1	direct-075	0.319716
	1	direct-0841	0.010308
	1	direct-09	0.284934
	1	general (k=1)	0.019798
	1	general (k=4)	**0.006361**

Table 5.1: Comparison of variance estimation algorithms. Variants named direct directly use a quantile of the input to estimate the standard deviation. Variants named general use Algorithm 18 with $k = 1$ and $k = 4$, and use the median to mean conversion for χ_2 distributed random variables.

5.D Settings for Experiments

Name	n	ρ	M	d	σ^2	μ
Gaussian A	4000	$\{1, 0.5, 0.125\}$	$\sqrt{50}d$	$\{16, 32, \ldots, 2048\}$	$(1, \ldots, 1)$ $\left(\left(\frac{d}{d}\right)^{\alpha}, \ldots, \left(\frac{d}{d-1}\right)^{\alpha}\right)$	0^d
Gaussian B	10 000	$\{1, 0.5, 0.125\}$	$\frac{100d}{\max \sigma}$	2048	$\ldots, \left(\frac{d}{1}\right)^{\alpha}$ $\left(\left(\frac{d}{d}\right)^{2}, \ldots, \left(\frac{d}{d-1}\right)^{2}\right)$	10^d
Gaussian C	10 000	$\{1, 0.5, 0.125\}$	$\frac{100d}{\max \sigma}$	$\{16, 32, \ldots, 2048\}$	$\ldots, \left(\frac{d}{1}\right)^{2}\Big)$	10^d
Binary	4096	$\{1, 0.5, 0.125\}$	$\lVert\hat{\sigma}^{2/3}\rVert_2$	$\{256, 512, \ldots, 2048\}$	$\sigma_i^2 = \mu_i(1-\mu_i)$	$\mu_i = \begin{cases} \frac{1}{2} & i \leq \alpha d \\ \frac{1}{100} & \text{else} \end{cases}$
Kosarak	75 462	$\{1, 0.5, 0.25, 0.125, 0.0625\}$	$\lVert\hat{\sigma}^{2/3}\rVert_2$	27 983	*Fixed dataset*	
POS	515 596	$\{1, 0.5, 0.25, 0.125, 0.0625\}$	$\lVert\hat{\sigma}^{2/3}\rVert_2$	1 657	*Fixed dataset*	
Global setting: $\rho_1 = 0.25\rho/d, \rho_2 = 0.25(\rho - \rho_1), \rho_3 = \rho - \rho_1 - \rho_2$.						

Chapter 6

Conclusion and Open Problems

We conclude the thesis by revisiting our main results and discussing open problems and directions for future work.

In Chapter 2 we introduced a differentially private data structure for representing sparse vectors with asymptotically optimal space complexity and per-entry error. The access time is $O(\log(d))$ or $O(\log(1/\delta))$ for pure and approximate differential privacy, respectively. The main open problem that we leave is if it is possible to achieve similar space and error with constant time access. In Section 2.8 we demonstrated how to achieve optimal *expected* error with constant time access and space within a logarithmic factor of optimal. However, this method does not have strong tail bounds on the error. While our data structure can be used to represent sparse vectors a solution for the special case of histograms would in and of itself be significant because that is a typical use case. Lolck and Pagh [LP23] recently made progress towards this goal. They give a more compact integer encoding which is still robust to noise by combining ideas from error-correcting codes and Grey codes.

Our mechanism presented in Chapter 3 adds error of magnitude $O(\log(1/\delta)/\varepsilon)$ to a Misra-Gries sketch and as such achieves optimal error guarantees for histograms up to a small constant. A natural direction for future work is to consider the setting where users can contribute up to m distinct elements. The preferred solution in the non-streaming setting is the Gaussian mechanism. The noise at each entry is scaled proportional to the ℓ_2-sensitivity, that is $\sqrt{m}$ since each count changes by at most 1. However, our mechanism must add noise

linearly proportional to m. Simply adding Gaussian noise instead of Laplace noise does not solve this problem because a single count in the Misra-Gries sketch can change by m between neighboring streams. As such it remains an open problem if we can achieve error that scales with $\sqrt{m}$ in the streaming setting.

In Chapter 4 we gave a closed-form expression for distributing noise between coordinates of different magnitudes for the Gaussian and Laplace mechanisms. We used this idea for the Gaussian mechanism in the design of a mechanism for differentially private mean estimation in Chapter 5. We performed experiments to support our claim that the mechanism outperforms state-of-the-art when the magnitudes of coordinates are sufficiently skewed. We see several possible directions for future work.

- Our mechanism tailors noise to the skew of the magnitude at each coordinate. An interesting avenue to explore would be to capture skew in the input vector that is not necessarily visible in the standard basis. For example, one could use private PCA to rotate the space into a basis in which coordinates are nearly independent, and then apply PLAN.
- Though our algorithm chooses optimal parameters within a class of mechanisms, we have not ruled out that an entirely different approach could have better performance. We conjecture that for any choice of σ there exists an input distribution for which our mechanism achieves an optimal trade-off up to logarithmic factors.
- We rely on having access to reasonable estimates of coordinate variances (the diagonal of the covariance matrix). It would be interesting to study this problem in its own right since it is likely that estimating the entire covariance matrix is strictly harder than estimating the diagonal.
- Finally, another direction would be to apply some of our techniques to differentially private stochastic gradient descent (DP-SGD). Previous work has successfully applied similar ideas by using coordinate-wise adaptive clipping of the gradient [PSY+19]. However, most implementations of DP-SGD use spherical Gaussian noise. Therefore it would be interesting to further explore techniques that add noise of different magnitudes to each coordinate.

Bibliography

[AAC+22] 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. *CoRR*, abs/2204.08986, 2022.

[ACG+16] Martín Abadi, Andy Chu, Ian J. Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *CCS*, pages 308–318. ACM, 2016.

[ACH+13] Pankaj K. Agarwal, Graham Cormode, Zengfeng Huang, Jeff M. Phillips, Zhewei Wei, and Ke Yi. Mergeable summaries. *ACM Trans. Database Syst.*, 38(4), dec 2013.

[AD20] Hilal Asi and John C Duchi. Instance-optimality in differential privacy via approximate inverse sensitivity mechanisms. In *Advances in Neural Information Processing Systems*, volume 33, pages 14106–14117. Curran Associates, Inc., 2020.

[ADK+19] Kareem Amin, Travis Dick, Alex Kulesza, Andres Munoz, and Sergei Vassilvitskii. Differentially Private Covariance Estimation. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[AKT+22] Daniel Alabi, Pravesh K. Kothari, Pranay Tankala, Prayaag Venkat, and Fred Zhang. Privately Estimating a Gaussian: Efficient, Robust and Optimal, December 2022.

[AL22] Hassan Ashtiani and Christopher Liaw. Private and polynomial time algorithms for learning Gaussians and beyond. In *Proceedings of Thirty Fifth Conference on Learning Theory*, pages 1075–1076. PMLR, June 2022.

[Alm02] Sven Erick Alm. Simple random walk. Unpublished manuscript: `http://www2.math.uu.se/~sea/kurser/stokprocmn1/slumpvandring_eng.pdf`, 2002. [Online; accessed 14-August-2023].

[ALNP23] Martin Aumüller, Christian Janos Lebeda, Boel Nelson, and Rasmus Pagh. PLAN: variance-aware private mean estimation. *CoRR*, abs/2306.08745, 2023.

[ALP22] Martin Aumüller, Christian Janos Lebeda, and Rasmus Pagh. Representing sparse vectors with differential privacy, low error, optimal space, and fast access. *Journal of Privacy and Confidentiality*, 12(2), Nov. 2022.

[App] Apple. Differential privacy overview - apple. `https://www.apple.com/privacy/docs/Differential_Privacy_Overview.pdf`. [Online; accessed 12-August-2023].

[Arr06] Michael Arrington. Aol proudly releases massive amounts of private data. `https://techcrunch.com/2006/08/06/aol-proudly-releases-massive-amounts-of-user-search-data/`, 2006. [Online; accessed 14-August-2023].

[ATMR21] Galen Andrew, Om Thakkar, Brendan McMahan, and Swaroop Ramaswamy. Differentially Private Learning with Adaptive Clipping. In *Advances in Neural Information Processing Systems*, volume 34, pages 17455–17466. Curran Associates, Inc., 2021.

[BDD+21] Mark Bun, Damien Desfontaines, Cynthia Dwork, Moni Naor, Kobbi Nissim, Aaron Roth, Adam Smith, Thomas Steinke, Jonathan Ullman, and Salil Vadhan. Statistical inference is not a privacy violation. DifferentialPrivacy.org, 06 2021. `https://differentialprivacy.org/inference-is-not-a-privacy-violation/`.

[BDKU20] Sourav Biswas, Yihe Dong, Gautam Kamath, and Jonathan Ullman. CoinPress: Practical Private Mean and Covariance Estimation. *Advances in Neural Information Processing Systems*, 33:14475–14485, 2020.

[BGMZ22] Jeremiah Blocki, Elena Grigorescu, Tamalika Mukherjee, and Samson Zhou. How to make your approximation algorithm private: A black-box differentially-private transformation for tunable approximation algorithms of functions with low sensitivity. *arXiv preprint arXiv:2210.03831*, 2022.

[BGS+21] Gavin Brown, Marco Gaboardi, Adam Smith, Jonathan Ullman, and Lydia Zakynthinou. Covariance-Aware Private Mean Estimation Without Private Covariance Estimation. In *Advances in Neural Information Processing Systems*, volume 34, pages 7950–7964. Curran Associates, Inc., 2021.

[BHS23] Gavin Brown, Samuel B. Hopkins, and Adam Smith. Fast, Sample-Efficient, Affine-Invariant Private Mean and Covariance Estimation for Subgaussian Distributions, January 2023.

[BK21] Jonas Böhler and Florian Kerschbaum. Secure multi-party computation of differentially private heavy hitters. In Yongdae Kim, Jong Kim, Giovanni Vigna, and Elaine Shi, editors, *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*, pages 2361–2377. ACM, 2021.

[BKMT03] Prosenjit Bose, Evangelos Kranakis, Pat Morin, and Yihui Tang. Bounds for frequency estimation of packet streams. In Jop F. Sibeyn, editor, *SIROCCO 10: Proceedings of the 10th Internaltional Colloquium on Structural Information Complexity, June 18-20, 2003, Umeå Sweden*, volume 17 of *Proceedings in Informatics*, pages 33–42. Carleton Scientific, 2003.

[BKT18] Austin R. Benson, Ravi Kumar, and Andrew Tomkins. A Discrete Choice Model for Subset Selection. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 37–45, Marina Del Rey CA USA, February 2018. ACM.

[BNS19a] Mark Bun, Jelani Nelson, and Uri Stemmer. Heavy hitters and the structure of local privacy. *ACM Transactions on Algorithms (TALG)*, 15(4):1–40, 2019.

[BNS19b] Mark Bun, Kobbi Nissim, and Uri Stemmer. Simultaneous private learning of multiple concepts. *J. Mach. Learn. Res.*, 20:94:1–94:34, 2019.

[BNSGT17] Raef Bassily, Kobbi Nissim, Uri Stemmer, and Abhradeep Guha Thakurta. Practical locally private heavy hitters. *Advances in Neural Information Processing Systems*, 30, 2017.

[BNSV15] Mark Bun, Kobbi Nissim, Uri Stemmer, and Salil Vadhan. Differentially Private Release and Learning of Threshold Functions. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 634–649, October 2015.

[Bre21] Brennan Center. Alabama v. u.s. dep't of commerce. `https://www.brennancenter.org/our-work/court-cases/alabama-v-us-dept-commerce`, 2021. [Online; accessed 6-August-2023].

[BS16] Mark Bun and Thomas Steinke. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Theory of Cryptography Conference*, pages 635–658. Springer, 2016.

[BV19] Victor Balcer and Salil Vadhan. Differential privacy on finite computers. *Journal of Privacy and Confidentiality*, 9(2), Sep. 2019.

[BW18] Borja Balle and Yu-Xiang Wang. Improving the gaussian mechanism for differential privacy: Analytical calibration and optimal denoising. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 403–412. PMLR, 2018.

[BWZK22] Zhiqi Bu, Yu-Xiang Wang, Sheng Zha, and George Karypis. Automatic Clipping: Differentially Private Deep Learning Made Easier and Stronger, July 2022.

[BZ06] Michael Barbaro and Tom Zeller. A face is exposed for aol searcher no. 4417749. `https://www.nytimes.com/2006/`

08/09/technology/09aol.html, 2006. [Online; accessed 14-August-2023].

[CFMT22] Rachel Cummings, Vitaly Feldman, Audra McMillan, and Kunal Talwar. Mean Estimation with User-level Privacy under Data Heterogeneity. *Advances in Neural Information Processing Systems*, 35:29139–29151, December 2022.

[CKR21] Rachel Cummings, Gabriel Kaptchuk, and Elissa M. Redmiles. "I need a better description": An investigation into user expectations for differential privacy. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, CCS '21, page 3037–3052, New York, NY, USA, 2021. Association for Computing Machinery.

[CLSX12] T-H Hubert Chan, Mingfei Li, Elaine Shi, and Wenchang Xu. Differentially private continual monitoring of heavy hitters from distributed streams. In *International Symposium on Privacy Enhancing Technologies Symposium*, pages 140–159. Springer, 2012.

[CM05] Graham Cormode and S. Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75, 2005.

[CPST12] Graham Cormode, Cecilia M. Procopiuc, Divesh Srivastava, and Thanh T. L. Tran. Differentially private summaries for sparse data. In *ICDT*, pages 299–311. ACM, 2012.

[CW79] Larry Carter and Mark N. Wegman. Universal classes of hash functions. *J. Comput. Syst. Sci.*, 18(2):143–154, 1979.

[CWGM20] Ricardo Silva Carvalho, Ke Wang, Lovedeep Gondara, and Chunyan Miao. Differentially private top-k selection via stability on unknown domain. In *Conference on Uncertainty in Artificial Intelligence*, pages 1109–1118. PMLR, 2020.

[CWZ21] T. Tony Cai, Yichen Wang, and Linjun Zhang. The cost of privacy: Optimal rates of convergence for parameter estimation with differential privacy. *The Annals of Statistics*, 49(5):2825–2850, October 2021.

[DFM+20] Wenxin Du, Canyon Foot, Monica Moniot, Andrew Bray, and Adam Groce. Differentially Private Confidence Intervals, January 2020.

[DHK23] John Duchi, Saminul Haque, and Rohith Kuditipudi. A Fast Algorithm for Adaptive Private Mean Estimation, January 2023.

[DHKP97] Martin Dietzfelbinger, Torben Hagerup, Jyrki Katajainen, and Martti Penttonen. A reliable randomized algorithm for the closest-pair problem. *J. Algorithms*, 25(1):19–51, 1997.

[DMNS06] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. pages 265–284, 2006.

[DMR+22] Sergey Denisov, H Brendan McMahan, Keith Rush, Adam Smith, and Abhradeep Thakurta. Improved Differential Privacy for SGD via Optimal Private Linear Operators on Adaptive Streams. *Advances in Neural Information Processing Systems*, 2022.

[DN03] Irit Dinur and Kobbi Nissim. Revealing information while preserving privacy. In *PODS*, pages 202–210. ACM, 2003.

[DP09] Devdatt P. Dubhashi and Alessandro Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, 2009.

[DR14] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–407, 2014.

[DR16] Cynthia Dwork and Guy N Rothblum. Concentrated differential privacy. *arXiv preprint arXiv:1603.01887*, 2016.

[DR19] David Durfee and Ryan M Rogers. Practical differentially private top-k selection with pay-what-you-get composition. *Advances in Neural Information Processing Systems*, 32, 2019.

[DRS19] Jinshuo Dong, Aaron Roth, and Weijie J. Su. Gaussian differential privacy. *CoRR*, abs/1905.02383, 2019.

[DTTZ14] Cynthia Dwork, Kunal Talwar, Abhradeep Thakurta, and Li Zhang. Analyze Gauss: Optimal bounds for privacy-preserving principal component analysis. In *Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing*, STOC '14, pages 11–20, New York, NY, USA, May 2014. Association for Computing Machinery.

[Elk13] Noam D. Elkies. Upper limit on the central binomial coefficient. `https://mathoverflow.net/questions/133732/upper-limit-on-the-central-binomial-coefficient`, 2013. [Online; accessed 15-September-2021].

[EPK14] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, pages 1054–1067, 2014.

[GGK+19] Badih Ghazi, Noah Golowich, Ravi Kumar, Rasmus Pagh, and Ameya Velingker. On the power of multiple anonymous messages. *IACR Cryptol. ePrint Arch.*, page 1382, 2019.

[GKOV15] Quan Geng, Peter Kairouz, Sewoong Oh, and Pramod Viswanath. The staircase mechanism in differential privacy. *IEEE Journal of Selected Topics in Signal Processing*, 9(7):1176–1184, 2015.

[GKP94] Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete Mathematics: A Foundation for Computer Science, 2nd Ed*. Addison-Wesley, 1994.

[GKS08] Srivatsava Ranjit Ganta, Shiva Prasad Kasiviswanathan, and Adam D. Smith. Composition attacks and auxiliary information in data privacy. In *KDD*, pages 265–273. ACM, 2008.

[GLW21] Sivakanth Gopi, Yin Tat Lee, and Lukas Wutschitz. Numerical composition of differential privacy, 2021.

[GRS12] Arpita Ghosh, Tim Roughgarden, and Mukund Sundararajan. Universally utility-maximizing privacy mechanisms. *SIAM Journal on Computing*, 41(6):1673–1693, 2012.

[Hag98] Torben Hagerup. Sorting and searching on the word RAM. In *STACS*, volume 1373 of *Lecture Notes in Computer Science*, pages 366–398. Springer, 1998.

[HDH+22] Samuel Haney, Damien Desfontaines, Luke Hartman, Ruchit Shrestha, and Michael Hay. Precision-based attacks and interval refining: how to break, then fix, differential privacy on finite computers. *CoRR*, abs/2207.13793, 2022.

[HKM22] Samuel B. Hopkins, Gautam Kamath, and Mahbod Majid. Efficient mean estimation with pure differential privacy via a sum-of-squares exponential mechanism. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2022, pages 1406–1417, New York, NY, USA, June 2022. Association for Computing Machinery.

[HLY21] Ziyue Huang, Yuting Liang, and Ke Yi. Instance-optimal Mean Estimation Under Differential Privacy. *Advances in Neural Information Processing Systems*, 34:25993–26004, 2021.

[HP14] Moritz Hardt and Eric Price. The Noisy Power Method: A Meta Algorithm with Applications. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[HT10] Moritz Hardt and Kunal Talwar. On the geometry of differential privacy. In *Proceedings Symposium on Theory of Computing (STOC)*, pages 705–714. ACM, 2010.

[KHP15] Fragkiskos Koufogiannis, Shuo Han, and George J. Pappas. Optimality of the Laplace Mechanism in differential privacy. *CoRR*, abs/1504.00065, 2015.

[KJH20] Antti Koskela, Joonas Jälkö, and Antti Honkela. Computing tight differential privacy guarantees using FFT. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 2560–2569. PMLR, 26–28 Aug 2020.

[KJS+23] Walid Krichene, Prateek Jain, Shuang Song, Mukund Sundararajan, Abhradeep Thakurta, and Li Zhang. Multi-Task

Differential Privacy Under Distribution Skew, February 2023.

[KKM+21] Christopher T. Kenny, Shiro Kuriwaki, Cory McCartan, Evan Rosenman, Tyler Simko, and Kosuke Imai. The impact of the U.S. census disclosure avoidance system on redistricting and voting rights analysis. *CoRR*, abs/2105.14197, 2021.

[KKMN09] Aleksandra Korolova, Krishnaram Kenthapadi, Nina Mishra, and Alexandros Ntoulas. Releasing search queries and clicks privately. In *WWW*, pages 171–180. ACM, 2009.

[KLSU19] Gautam Kamath, Jerry Li, Vikrant Singhal, and Jonathan Ullman. Privately Learning High-Dimensional Distributions. In *Proceedings of the Thirty-Second Conference on Learning Theory*, pages 1853–1902. PMLR, June 2019.

[KMR+23] Gautam Kamath, Argyris Mouzakis, Matthew Regehr, Vikrant Singhal, Thomas Steinke, and Jonathan Ullman. A Bias-Variance-Privacy Trilemma for Statistical Estimation, January 2023.

[KMS+22] Gautam Kamath, Argyris Mouzakis, Vikrant Singhal, Thomas Steinke, and Jonathan Ullman. A Private and Computationally-Efficient Estimator for Unbounded Gaussians. In *Proceedings of Thirty Fifth Conference on Learning Theory*, pages 544–572. PMLR, June 2022.

[KMV22] Pravesh Kothari, Pasin Manurangsi, and Ameya Velingker. Private Robust Estimation by Stabilizing Convex Relaxations. In *Proceedings of Thirty Fifth Conference on Learning Theory*, pages 723–777. PMLR, June 2022.

[KOV15] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. The composition theorem for differential privacy. In *ICML*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 1376–1385. JMLR.org, 2015.

[KSS22] Haim Kaplan, Shachar Schnapp, and Uri Stemmer. Differentially Private Approximate Quantiles. In *Proceedings of the 39th International Conference on Machine Learning*, pages 10751–10761. PMLR, June 2022. ISSN: 2640-3498.

[KSU20] Gautam Kamath, Vikrant Singhal, and Jonathan Ullman. Private Mean Estimation of Heavy-Tailed Distributions. In *Proceedings of Thirty Third Conference on Learning Theory*, pages 2204–2235. PMLR, July 2020.

[KV18] Vishesh Karwa and Salil Vadhan. Finite Sample Differentially Private Confidence Intervals. In *9th Innovations in Theoretical Computer Science Conference (ITCS 2018)*, page 9 pages. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik GmbH, Wadern/Saarbruecken, Germany, 2018.

[LP23] David Rasmussen Lolck and Rasmus Pagh. Shannon meets gray: Noise-robust, low-sensitivity codes with applications in differential privacy. *CoRR*, abs/2305.02816, 2023.

[LT23] Christian Janos Lebeda and Jakub Tetek. Better differentially private approximate histograms and heavy hitters using the misra-gries sketch. In *PODS*, pages 79–88. ACM, 2023.

[Mar20] Mark. Differential privacy guarantees of gaussian noise, when each coordinate has different sensitivity. `https://crypto.stackexchange.com/questions/85581/differential-privacy-guarantees-of-gaussian-noise-when-each-coordinate-has-diff`, 2020. [Online; accessed 06-May-2022].

[McK18] Laura McKenna. Disclosure Avoidance Techniques Used for the 1970 through 2010 Decennial Censuses of Population and Housing. Working Papers 18-47, Center for Economic Studies, U.S. Census Bureau, Nov 2018.

[MG82] J. Misra and David Gries. Finding repeated elements. *Science of Computer Programming*, 2(2):143–152, 1982.

[Mir17] Ilya Mironov. Rényi differential privacy. In *2017 IEEE 30th computer security foundations symposium (CSF)*, pages 263–275. IEEE, 2017.

[MMNW11] Darakhshan J. Mir, S. Muthukrishnan, Aleksandar Nikolov, and Rebecca N. Wright. Pan-private algorithms via statistics on sketches. In Maurizio Lenzerini and Thomas Schwentick,

editors, *Proceedings of the 30th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2011, June 12-16, 2011, Athens, Greece*, pages 37–48. ACM, 2011.

[MRTZ18] H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning Differentially Private Recurrent Language Models, February 2018.

[MT07] Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *FOCS*, pages 94–103. IEEE Computer Society, 2007.

[MTZ19] Ilya Mironov, Kunal Talwar, and Li Zhang. Rényi differential privacy of the sampled gaussian mechanism, 2019.

[MU05] Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005.

[NTZ16] Aleksandar Nikolov, Kunal Talwar, and Li Zhang. The geometry of differential privacy: The small database and approximate cases. *SIAM J. Comput.*, 45(2):575–616, 2016.

[PDD+22] Prottay Protivash, John Durrell, Zeyu Ding, Danfeng Zhang, and Daniel Kifer. Reconstruction attacks on aggressive relaxations of differential privacy. *CoRR*, abs/2209.03905, 2022.

[PSY+19] Venkatadheeraj Pichapati, Ananda Theertha Suresh, Felix X. Yu, Sashank J. Reddi, and Sanjiv Kumar. AdaCliP: Adaptive Clipping for Private SGD, October 2019.

[PT22] Rasmus Pagh and Mikkel Thorup. Improved utility analysis of private countsketch. In *Advances in Neural Information Processing Systems*, volume 35, pages 25631–25643, 2022.

[QSZ21] Gang Qiao, Weijie Su, and Li Zhang. Oneshot differentially private top-k selection. In *International Conference on Machine Learning*, pages 8672–8681. PMLR, 2021.

[QYY+16] Zhan Qin, Yin Yang, Ting Yu, Issa Khalil, Xiaokui Xiao, and Kui Ren. Heavy hitter estimation over set-valued data with local differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 192–203, 2016.

[Rén61] Alfréd Rényi. On measures of entropy and information. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*, volume 4, pages 547–562. University of California Press, 1961.

[Smi11] Adam Smith. Privacy-preserving statistical estimation with optimal convergence rates. In *Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing*, STOC '11, pages 813–822, New York, NY, USA, June 2011. Association for Computing Machinery.

[SS21] Vikrant Singhal and Thomas Steinke. Privately Learning Subspaces. In *Advances in Neural Information Processing Systems*, volume 34, pages 1312–1324. Curran Associates, Inc., 2021.

[Ste22] Thomas Steinke. Composition of differential privacy & privacy amplification by subsampling. *CoRR*, abs/2210.00597, 2022.

[Swe02] Latanya Sweeney. k-anonymity: A model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl. Based Syst.*, 10(5):557–570, 2002.

[Swe15] Latanya Sweeney. Only you, your doctor, and many others may know. *Technology Science*, 2015092903(9):29, 2015.

[Tět22] Jakub Tětek. Additive noise mechanisms for making randomized approximation algorithms differentially private. *arXiv preprint arXiv:2211.03695*, 2022.

[The18] The New York Times. Cambridge analytica and facebook: The scandal and the fallout so far. `https://www.nytimes.com/2018/04/04/us/politics/cambridge-analytica-scandal-fallout.html`, 2018. [Online; accessed 10-January-2022].

[Uni21a] United States Census Bureau. About the decennial census of population and housing. https://www.census.gov/programs-surveys/decennial-census/about.html, 2021. [Online; accessed 6-August-2023].

[Uni21b] United States Census Bureau. The census bureau's simulated reconstruction-abetted re-identification attack on the 2010 census. https://www.census.gov/data/academy/webinars/2021/disclosure-avoidance-series/simulated-reconstruction-abetted-re-identification-attack-on-the-2010-census.html, 2021. [Online; accessed 11-August-2023].

[Uni21c] United States Census Bureau. Das 2020 redistricting production code release. https://github.com/uscensusbureau/DAS_2020_DHC_Production_Code, 2021.

[Uni22] United States Census Bureau. 2020 census undercounts in six states, overcounts in eight. https://www.census.gov/library/stories/2022/05/2020-census-undercount-overcount-rates-by-state.html, 2022. [Online; accessed 13-August-2023].

[Uni23] United States Census Bureau. Why the census bureau chose differential privacy. https://www.census.gov/library/publications/2023/decennial/c2020br-03.html, 2023. [Online; accessed 11-August-2023].

[War65] Stanley L Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309):63–69, 1965.

[WBK18] Yu-Xiang Wang, Borja Balle, and Shiva Kasiviswanathan. Subsampled rényi differential privacy and analytical moments accountant, 2018.

[WG50] Ernest L Wynder and Evarts A Graham. Tobacco smoking as a possible etiologic factor in bronchiogenic carcinoma: a study of six hundred and eighty-four proved cases. *Journal of the American medical association*, 143(4):329–336, 1950.

[Win12] Andreas Winkelbauer. Moments and absolute moments of the normal distribution. *arXiv preprint arXiv:1209.4340*, 2012.

[WLJ19] Tianhao Wang, Ninghui Li, and Somesh Jha. Locally differentially private heavy hitter identification. *IEEE Transactions on Dependable and Secure Computing*, 18(2):982–993, 2019.

[WW22] Hao Wu and Anthony Wirth. Asymptotically optimal locally private heavy hitters via parameterized sketches. In *International Conference on Artificial Intelligence and Statistics*, pages 7766–7798. PMLR, 2022.

[yCDGL23] James Hsin yu Chiang, Bernardo David, Mariana Gama, and Christian Janos Lebeda. Correlated-Output Differential Privacy and Applications to Dark Pools. In *Advances in Financial Technologies (AFT)*, 2023.

[ZDW22] Yuqing Zhu, Jinshuo Dong, and Yu-Xiang Wang. Optimal accounting of differential privacy via characteristic function. In Gustau Camps-Valls, Francisco J. R. Ruiz, and Isabel Valera, editors, *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pages 4782–4817. PMLR, 28–30 Mar 2022.

[Zel06] Tom Zeller. Aol technology chief quits after data release. `https://www.nytimes.com/2006/08/21/technology/21cnd-aol.html`, 2006. [Online; accessed 14-August-2023].

[ZKM+20] Wennan Zhu, Peter Kairouz, Brendan McMahan, Haicheng Sun, and Wei Li. Federated heavy hitters discovery with differential privacy. In *International Conference on Artificial Intelligence and Statistics*, pages 3837–3847. PMLR, 2020.

[ZQR+22] Fuheng Zhao, Dan Qiao, Rachel Redberg, Divyakant Agrawal, Amr El Abbadi, and Yu-Xiang Wang. Differentially private linear sketches: Efficient implementations and applications. In *Advances in Neural Information Processing Systems*, volume 35, pages 12691–12704, 2022.

[ZWC+22] Mingxun Zhou, Tianhao Wang, T.-H. Hubert Chan, Giulia Fanti, and Elaine Shi. Locally differentially private sparse vector aggregation. In *IEEE Symposium on Security and Privacy*, pages 422–439. IEEE, 2022.

[ZZC+22] Dan Zhao, Suyun Zhao, Hong Chen, Ruixuan Liu, Cuiping Li, and Wenjuan Liang. Efficient protocols for heavy hitter identification with local differential privacy. *Frontiers of Computer Science*, 16(5):1–11, 2022.