Lossless Compression of Efficient Private Local Randomizers

Vitaly Feldman [1] **Kunal Talwar** [1]

Abstract

Locally Differentially Private (LDP) Reports are commonly used for collection of statistics and machine learning in the federated setting. In many cases the best known LDP algorithms require sending prohibitively large messages from the client device to the server (such as when constructing histograms over a large domain or learning a high-dimensional model). Here we demonstrate a general approach that, under standard cryptographic assumptions, compresses every efficient LDP algorithm with negligible loss in privacy and utility guarantees. The practical implication of our result is that in typical applications every message can be compressed to the size of the server's pseudo-random generator seed. From this general approach we derive low-communication algorithms for the problems of frequency estimation and high-dimensional mean estimation. Our algorithms are simpler and more accurate than existing low-communication LDP algorithms for these well-studied problems.

1 Introduction

We consider the problem of collecting statistics and machine learning in the setting where data is held on a large number of user devices. The data held on devices in this *federated* setting is often sensitive and thus needs to be analyzed with privacy preserving techniques. One of the key approaches to private federated data analysis relies on the use of locally differentially private (LDP) algorithms to ensure that the report sent by a user's device reveals little information about that user's data. Specifically, a randomized algorithm $\mathcal{R}\colon X \to Y$ is an ε-DP local randomizer if for every possible output $y \in Y$, and any two possible values of user data $x_1, x_2 \in X$, $\mathbf{Pr}[\mathcal{R}(x_1) = y]$ and $\mathbf{Pr}[\mathcal{R}(x_2) = y]$ are within a factor of e^{ε} (where the probability is taken solely with respect to the randomness of the algorithm $\mathcal{R}$).

[1]Apple. Correspondence to: Vitaly Feldman <vitaly.edu@gmail.com>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

The concept of a local randomizer dates back to the work of Warner (1965) where it was used to encourage truthfulness in surveys. In the context of modern data analysis it was introduced by Evfimievski et al. (2003) and then related to differential privacy in the seminal work of Dwork et al. (2006). Local randomizers are also used for collection of statistics and machine learning in several industrial applications (Erlingsson et al., 2014; Apple's Differential Privacy Team, 2017; Ding et al., 2017). Practical applications such as building a histogram over a large domain or training a model with millions of parameters (McMahan et al., 2018), require applying the randomizer to high dimensional data. Many of the standard and most accurate ways to randomize such data result in reports whose size scales linearly with the dimension of the problem. Communication from the user devices is often significantly constrained in practical applications. This limits the scope of problems in which we can achieve the best known utility-privacy tradeoff and motivates significant research interest in designing communication-efficient LDP algorithms.

1.1 Our contribution

In this work, we explore practical and theoretical aspects of designing low-communication LDP mechanisms. It has long been noted that, by design, the output of an LDP randomizer contains a limited amount of information about the input data. Thus it should be possible to reduce the communication down to the information content about the data using standard tools from information theory. We will refer to such reduction in communication as *compression* of the LDP randomizer. To avoid confusion, we note that the goal is not to compress the content of the original messages of the randomizer but rather to find a different randomizer that communicates the same information about the data point using shorter messages.

Unfortunately, standard information-theoretic approaches to such compression do not necessarily preserve privacy. Bassily & Smith (2015) and Bun et al. (2019) describe general privacy preserving techniques for compressing LDP protocols. Unfortunately, their techniques result either in the loss of accuracy (as only a fraction of the user population ends up contributing a report) or an increase in ε by a constant factor > 2. Increase by such a constant factor is likely to make the algorithm impractical in the $\varepsilon > 1$

regime since in some problems accuracy scales as $e^{-\varepsilon/2}$ when $\varepsilon > 1$. The $\varepsilon > 1$ regime has become particularly important since the introduction of the privacy amplification techniques based on anonymization and shuffling of LDP reports (Bittau et al., 2017; Erlingsson et al., 2019; Cheu et al., 2019; Balle et al., 2019; Feldman et al., 2020). Central privacy guarantees resulting from amplification by shuffling scale as $e^{-\varepsilon/2}$ making preservation of ε crucial in this setting.

We propose a general approach to compressing an arbitrary local randomizer that preserves both the privacy and accuracy (or utility) of the randomizer. At a high level it is based on replacing the true random bits used to generate the output with pseudo-random bits that can be described using a short seed. For a randomizer $\mathcal{R}\colon X \to Y$, we do this by first picking a fixed reference distribution ρ that is data-independent and ε-close (in the standard sense of differential privacy) to the output distributions of $\mathcal{R}$ for all possible inputs $x \in X$. Existence of such reference distribution is exactly the definition of the deletion version of local differential privacy (Erlingsson et al., 2020) and thus our results are easiest to describe in this model. A sample from ρ typically requires many random bits to generate but, by replacing random bits with pseudo-randomly generated ones, we will obtain a distribution over values in Y that can be described using a short seed. In addition, under standard cryptographic assumptions, a random sample from this distribution is computationally indistinguishable from ρ. Given an input x we can now emulate $\mathcal{R}(x)$ by performing rejection sampling relative to pseudo-random samples from ρ. A special case of this idea appears in the work of Mishra & Sandler (2006) who apply it the problem of estimating sets of counting queries.

A crucial question is whether this scheme satisfies ε differential privacy. We show that the answer is yes if the pseudo-random generator (PRG) used is strong enough to fool a certain test that looks at the ratio of the output density of $\mathcal{R}(x)$ to ρ. This ratio is typically efficiently computable whenever the randomizer itself is efficiently computable. Thus under standard cryptographic assumptions, the privacy is preserved (up to a negligible loss). Similarly, when the processing of the reports on the server side is done by an efficient algorithm the utility will be preserved. See Theorem 3.4 for a formal statement. Asymptotically, this result implies that if we assume that there exists an exponentially strong PRG, then the number of bits that needs to be communicated is logarithmic in the running time of the rejection sampler we defined. An immediate practical implication of this result is that in most applications the output of the local randomizer can be compressed to the size of the seed of the system (PRG) without any observable effect on utility or privacy. This size is typically less than 1024 bits. We remark that when implementing a randomizer in practice, true randomness is replaced with pseudo-randomly generated bits with an (implicit) assumption that this does not affect privacy or utility guarantees. Thus the assumptions underlying our analysis are similar to those that are already present in practical implementations of differentially private algorithms.

We demonstrate that this approach also extends to the (more common) replacement notion of local differential privacy and also to (ε, δ)-DP randomizers. In the latter case the randomizer needs to be modified to allow subsampling via simple truncation. This step adds δ to both privacy and utility guarantees of the algorithm. For replacement DP this version also requires a more delicate analysis and a stronger set of tests for the PRG. A detailed description of these results is given in Section 3.

An important property of our analysis is that we do not need to follow the general recipe for specific randomizers. Firstly, for some randomizers it is possible to directly sample from the desired distribution over seeds instead of using rejection sampling that requires e^{ε} trials (in expectation). In addition, it may be possible to ensure that privacy and utility are preserved without appealing to general cryptographically secure PRGs and associated computational assumptions. In particular, one can leverage a variety of sophisticated results from complexity theory, such as k-wise independent PRGs and PRGs for bounded-space computation (Nisan, 1992), to achieve unconditional and more efficient compression.

We apply this fine-grained approach to the problem of frequency estimation over a discrete domain. In this problem the domain $X = [k]$ and the goal is to estimate the frequency of each element $j \in [k]$ in the dataset. This is one of the central and most well-studied problems in private (federated) data analysis. However, for $\varepsilon > 1$, existing approaches either require communication on the order of k bits, or do not achieve the best known accuracy in some important regimes (see Sec. 1.2 for an overview).

The best accuracy is achieved for this problem is achieved by the (asymmetric) RAPPOR algorithm (Erlingsson et al., 2014) (which has two versions depending on whether it is used with replacement or deletion privacy) and also by the closely related Subset Selection algorithm (Wang et al., 2016; Ye & Barg, 2018). We observe that a pairwise-independent PRG suffices to fool both the privacy and utility conditions for this randomizer. Thus we can compress RAPPOR to $O(\log k + \varepsilon)$ bits losslessly and unconditionally using a standard construction of a pairwise-independent PRG (Luby et al., 2006). The structure of the PRG also allows us to sample the seeds efficiently without rejection sampling. The details of this construction appear in Section 3.

As an additional application of our techniques we consider the problem of estimating the mean of d-dimensional vectors

in ℓ_2-norm. This problem is a key part of various machine learning algorithms, most notably stochastic gradient descent. In the $\varepsilon > 1$ regime, the first low-communication (specifically, $\lceil \varepsilon \rceil \log_2 d$ bits) and asymptotically optimal algorithm was recently given by Chen et al. (2020). It is however less accurate empirically and more involved than the algorithm of Bhowmick et al. (2019) that communicates a d dimensional vector. Using our general result we can losslessly compress the algorithm from (Bhowmick et al., 2019) to $O(\log d + \varepsilon)$ bits. One limitation of this approach is the $O(e^{\varepsilon} d)$ complexity of rejection sampling in this case which can be prohibitive for large ε. However we show a simple reduction of the $\varepsilon > 1$ case to $\varepsilon < 1$ which increases communication but a factor of $\lceil \varepsilon \rceil$. This general reduction allows us to reduce the running time to $O(\lceil \varepsilon \rceil d)$ and also use a simple and low-communication randomizer that is (asymptotically) optimal only when $\varepsilon < 1$ (Duchi et al., 2018; Erlingsson et al., 2020). The details of these results and empirical comparisons appear in Section 5.

1.2 Related Work

As mentioned, the closest in spirit to our work is the use of rejection sampling in the work of Mishra & Sandler (2006). Their analysis can be seen as a special case of ours but they only prove that the resulting algorithm satisfies 2ε-DP. Rejection sampling on a sample from the reference distribution is also used in existing compression schemes (Bassily & Smith, 2015; Bun et al., 2019) as well as earlier work on private compression in the two-party setting (McGregor et al., 2010). These approaches assume that the sample is shared between the client and the server, namely, it requires shared randomness. Shared randomness is incompatible with the setting where the report is anonymized and is not directly linked to the user that generated it. As pointed out in (Bassily & Smith, 2015), a simple way to overcome this problem is to include a seed to a PRG in the output of the randomizer and have the server generate the same sample from the reference distribution as the client. While superficially this approach seems similar to ours, its analysis and properties are different. For example, in our setting only the seed for a single sample that passes rejection sampling is revealed to the server, whereas in (Bassily & Smith, 2015; Bun et al., 2019) all samples from the reference distribution are known to the server and privacy analysis does not depend on the strength of the PRG. More importantly, unlike previous approaches our compression scheme is essentially lossless (although at the cost of requiring assumptions for the privacy analysis).

Computational Differential Privacy (CDP) (Mironov et al., 2009) is a notion of privacy that defends against computationally bounded adversaries. Our compression algorithm can be easily shown to satisfy the strongest SIM-CDP definition. At the same time, our privacy bounds also hold for computationally unbounded adversaries as long as the LDP algorithm itself does not lead to a distinguisher. This distinction allows us to remove computational assumptions for specific LDP randomizers.

For both deletion and replacement privacy the best results for frequency estimation are achieved by variants of the RAPPOR algorithm (Erlingsson et al., 2014) and also by a closely-related Subset Selection algorithm (Wang et al., 2016; Ye & Barg, 2018). Unfortunately, both RAPPOR and Subset Selection have very high communication cost of $\approx kH(1/(e^{\varepsilon}+1))$, where H is the binary entropy function. This has led to numerous and still ongoing efforts to design low-communication protocols for the problem (Hsu et al., 2012; Erlingsson et al., 2014; Bassily & Smith, 2015; Kairouz et al., 2016; Wang et al., 2016; 2017; Ye & Barg, 2018; Acharya et al., 2019; Acharya & Sun, 2019; Bun et al., 2019; Bassily et al., 2020; Chen et al., 2020).

A number of low-communication algorithms that achieve asymptotically optimal bounds in the $\varepsilon < 1$ regime are known (Bassily & Smith, 2015; Wang et al., 2017; Acharya et al., 2019; Acharya & Sun, 2019; Bassily et al., 2020; Chen et al., 2020). The first low-communication algorithm that achieves asymptotically optimal bounds in the $\varepsilon > 1$ regime is given in (Wang et al., 2017). It communicates $O(\varepsilon)$ bits and relies on shared randomness. However, it matches the bounds achieved by RAPPOR only when e^{ε} is an integer. Acharya & Sun (2019) and Chen et al. (2020) give closely related approaches that are asymptotically optimal and use $\log_2 k$ bits of communication (without shared randomness). However both the theoretical bounds and empirical results for these algorithms are noticeably worse than those of (asymmetric) RAPPOR and Subset Selection (e.g. plots in (Chen et al., 2020) show that these algorithms are ≈15-20% worse for $\varepsilon = 5$ than Subset Selection[1]). The constructions in (Acharya & Sun, 2019; Chen et al., 2020) and their analysis are also substantially more involved than RAPPOR.

A closely related problem is finding "heavy hitters", namely all elements $j \in [k]$ with counts higher than some given threshold. In this problem the goal is to avoid linear runtime dependence on k that would result from doing frequency estimation and then checking all the estimates. This problem is typically solved using a "frequency oracle" which is an algorithm that for a given $j \in [k]$ returns an estimate of the number of j's held by users (typically without computing the entire histogram) (Bassily & Smith, 2015; Bassily et al., 2020; Bun et al., 2019). Frequency estimation is also closely

[1] The error of asymmetric RAPPOR (namely 0 and 1 are flipped with different probabilities) is essentially identical to that of the Subset Selection randomizer. Comparisons with RAPPOR often use the symmetric RAPPOR which is substantially worse than the asymmetric version for the replacement notion of differential privacy. See Section 4 for details.

related to the discrete distribution estimation problem in which inputs are sampled from some distribution over $[k]$ and the goal is to estimate the distribution (Ye & Barg, 2018; Acharya et al., 2019; Acharya & Sun, 2019). Indeed, bounds for frequency estimation can be translated directly to bounds on distribution estimation by adding the sampling error.

Mean estimation has attracted a lot of attention in recent years as it is an important subroutine in differentially private (stochastic) gradient descent algorithms (Bassily et al., 2014; Abadi et al., 2016) used in private federated learning (Kairouz et al., 2019). Indeed, private federated optimization algorithms aggregate updates to the model coming from each client in a batch of clients by getting a private estimate of the average update. When the models are large, the dimensionality of the update d leads to significant communication cost. Thus reducing the communication cost of mean estimation has been studied in many works with (Agarwal et al., 2018; Girgis et al., 2020; Chen et al., 2020; Gandikota et al., 2019) or without privacy (Alistarh et al., 2017; Faghri et al., 2020; Suresh et al., 2017; Gandikota et al., 2019; Mayekar & Tyagi, 2020).

In the absence of communication constraints and $\varepsilon < d$, the optimal ε-LDP protocols for this problem achieve an expected squared ℓ_2 error of $\Theta(\frac{d}{n\min(\varepsilon,\varepsilon^2)})$ (Duchi et al., 2018; Duchi & Rogers, 2019). When $\varepsilon \leq 1$, the randomizer of Duchi et al. (2018) also achieves the optimal $O(\frac{d}{n\varepsilon^2})$ bound. Recent work of Erlingsson et al. (2020) gives a low-communication version of this algorithm. Building on the approach in (Duchi et al., 2018), Bhowmick et al. (2019) describe the `PrivUnit` algorithm that achieves the asymptotically optimal accuracy also when $\varepsilon > 1$ but has communication cost of $\Omega(d)$.

An alternative approach in the $\varepsilon < 1$ regime was given by Feldman et al. (2015) who show that the mean estimation problem can be solved by having each client answer a single counting query. This approach is based on Kashin's representation that maps vectors in the unit d-dimensional ball to vectors in $[-1,1]^{O(d)}$ (Lyubarskii & Vershynin, 2010). Their work does not explicitly discuss the communication cost and assumes that the server can pick the randomizer used at each client. However it is easy to see that a single bit suffices to answer a counting query and therefore an equivalent randomizer can be implemented using $\lceil \log_2 d \rceil + 1$ bits of communication (or just 1 bit if shared randomness is used). Chen et al. (2020) give a randomizer based on the same idea that also achieves the asymptotically optimal bound in the $d > \varepsilon > 1$ regime. Their approach uses $\lceil \varepsilon \rceil \log_2 d$ bits of communication. Computing Kashin's representation is more involved than algorithms in (Duchi et al., 2018; Bhowmick et al., 2019). In addition, as we demonstrate empirically[2], the variance of the estimate resulting from this approach is nearly a factor of $5\times$ larger for typical parameters of interest.

2 Preliminaries

For a positive integer k we denote $[k] = \{1, 2 \ldots, k\}$. For an arbitrary set S we use $x \sim S$ to mean that x is chosen randomly and uniformly from S.

Differential privacy (DP) is a measure of stability of a randomized algorithm. It bounds the change in the distribution on the outputs when one of the inputs is either removed or replaced with an arbitrary other element. The most common way to measure the change in the output distribution is via approximate infinity divergence. More formally, we say that two probability distributions μ and ν over (finite) domain Y are (ε, δ)-*close* if for all $E \subset Y$,

$$e^{-\varepsilon}(\mu(E) - \delta) \leq \nu(E) \leq e^{\varepsilon}\mu(E) + \delta.$$

This condition is equivalent to $\sum_{y\in Y} |\mu(y) - e^{\varepsilon}\nu(y)|_+ \leq \delta$ and $\sum_{y\in Y} |\nu(y) - e^{\varepsilon}\mu(y)|_+ \leq \delta$, where $|a|_+ := \max\{a, 0\}$ (Dwork & Roth, 2014). We also say that two random variables P and Q are (ε, δ)-close if their probability distributions are (ε, δ)-close. We abbreviate $(\varepsilon, 0)$-close to ε-close.

Algorithms in the local model of differential privacy and federated data analysis rely on the notion of *local randomizer*.

Definition 2.1. *An algorithm $\mathcal{R}\colon X \to Y$ is an (ε, δ)-DP* local randomizer *if for all pairs $x, x' \in \mathcal{D}$, $\mathcal{R}(x)$ and $\mathcal{R}(x')$ are (ε, δ)-close.*

We will also use the add/delete variant of differential privacy which was defined for local randomizers in (Erlingsson et al., 2020).

Definition 2.2. *An algorithm $\mathcal{R}\colon X \to Y$ is a* deletion (ε, δ)-DP local randomizer *if there exists a reference distribution ρ such that for all data points $x \in X$, $\mathcal{R}(x)$ and ρ are (ε, δ)-close.*

It is easy to see that a replacement (ε, δ)-DP algorithm is also a deletion (ε, δ)-DP algorithm, and that a deletion (ε, δ)-DP algorithm is also a replacement $(2\varepsilon, 2\delta)$-DP algorithm.

Fooling and Pseudorandomness: The notion of pseudorandomness relies on the inability to distinguish between the output of the generator and true randomness using a family of tests, where a test is a boolean function (or algorithm).

[2] Plots in (Chen et al., 2020) also compare their algorithm with `PrivUnit` yet as their code at (Kas) shows and was confirmed by the authors, they implemented the algorithm from (Duchi et al., 2018) instead of `PrivUnit` which is much worse than `PrivUnit` for $\varepsilon = 5$. The authors also confirmed that parameters stated in their figures are incorrect so cannot be directly compared to our results.

Definition 2.3. *Let $\mathcal{D}$ be a family of boolean functions over some domain Y. We say that two random variables P and Q over Y are $(\mathcal{D}, \beta)$-indistinguishable if for all $D \in \mathcal{D}$,*

$$|\mathbf{Pr}[D(P) = 1] - \mathbf{Pr}[D(Q) = 1]| \leq \beta.$$

We say that P and Q are (T, β)-computationally indistinguishable if P and Q are $(\mathcal{D}, \beta)$-indistinguishable with $\mathcal{D}$ being all tests that can be computed in time T (for some fixed computational model such as boolean circuits).

We now give a definition of a pseudo-random number generator.

Definition 2.4 (Pseudo-random generator)**.** *We say that an algorithm $G\colon \{0,1\}^n \to \{0,1\}^m$ where $m \gg n$, β-fools a family of tests $\mathcal{D}$ if $G(s)$ for $s \sim \{0,1\}^n$ is $(\mathcal{D}, \beta)$-indistinguishable from r for $r \sim \{0,1\}^m$. We refer to such an algorithm as $(\mathcal{D}, \beta)$-PRG and also use (T, β)-PRG to refer to G that β-fools all tests running in time T.*

Standard cryptographic assumptions (namely that one-way functions exist) imply that for any m and T that are polynomial in n there exists an efficiently computable (β, T)-PRG G, for negligible β (namely, $\beta = 1/n^{\omega(1)}$). For a number of standard approaches to cryptographically-secure PRGs, no tests are known that can distinguish the output of the PRG from true randomness with $\beta = 2^{-o(n)}$ in time $T = 2^{o(n)}$. For example finding such a test for a PRG based on SHA-3 would be a major breakthrough. To make the assumption that such a test does not exist we refer to a (β, T)-PRG for $\beta = 2^{-\Omega(n)}$ and $T = 2^{\Omega(n)}$ as an *exponentially strong* PRG.

3 Local Pseudo-Randomizers

In this section we describe a general way to compress LDP randomizers that relies on the complexity of the randomizer and subsequent processing. We will first describe the result for deletion ε-DP and then give the versions for replacement DP and (ε, δ)-DP.

For the purpose of this result we first need to quantify how much randomness a local randomizer needs. We will say that a randomizer $\mathcal{R}\colon X \to Y$ is t-samplable if there exists a deterministic algorithm $\mathcal{R}_\emptyset\colon \{0,1\}^t \to Y$ such that for r chosen randomly and uniformly from $\{0,1\}^t$, $\mathcal{R}_\emptyset(r)$ is distributed according to the reference distribution of $\mathcal{R}$ (denoted by ρ). Typically, for efficiently computable randomizers, t is polynomial in the size of the output $\log(|Y|)$ and ε. Note that every value y in the support of ρ is equal to $\mathcal{R}_\emptyset(r)$ for some r. Thus every element that can be output by $\mathcal{R}$ can be represented by some $r \in \{0,1\}^t$.

Our goal is to compress the communication by restricting the output from all those that can be represented by $r \in \{0,1\}^t$ to all those values in Y that can be represented by a t-bit string generated from a seed of length $\ell \ll t$ using some PRG $G\colon \{0,1\}^\ell \to \{0,1\}^t$. We could then send the seed to the server and let the server first generate the full t-bit string using G and then run $\mathcal{R}_\emptyset$ on it. The challenge is to do this efficiently while preserving the privacy and utility guarantees of $\mathcal{R}$.

Our approach is based on the fact that we can easily sample from the pseudo-random version of the reference distribution ρ by outputting $\mathcal{R}_\emptyset(G(s))$ for a random and uniform seed s. This leads to a natural way to define a distribution over seeds on a input x: a seed s is output with probability that is proportional to $\frac{\mathbf{Pr}[\mathcal{R}(x)=\mathcal{R}_\emptyset(G(s))]}{\mathbf{Pr}_{r\sim\{0,1\}^t}[\mathcal{R}_\emptyset(r)=\mathcal{R}_\emptyset(G(s))]}$. Specifically we define the desired randomizer as follows.

Definition 3.1. *For a t-samplable deletion DP local randomizer $\mathcal{R}\colon X \to Y$ and a function $G\colon \{0,1\}^\ell \to \{0,1\}^t$ let $\mathcal{R}[G]$ denote the local randomizer that given $x \in X$, outputs $s \in \{0,1\}^t$ with probability proportional to $\frac{\mathbf{Pr}[\mathcal{R}(x)=\mathcal{R}_\emptyset(G(s))]}{\mathbf{Pr}_{r\sim\{0,1\}^t}[\mathcal{R}_\emptyset(r)=\mathcal{R}_\emptyset(G(s))]}$.*

For some combinations of a randomizer $\mathcal{R}$ and PRG G there is an efficient way to implement $\mathcal{R}[G]$ directly (as we show in one of our applications). In the general case, when such algorithm may not exist we can sample from $R[G](x)$ by applying rejection sampling to uniformly generated seeds. A special case of this approach is implicit in the work of Mishra & Sandler (2006) (albeit with a weaker analysis). Rejection sampling only requires an efficient algorithm for computing the ratio of densities above to sufficiently high accuracy. We describe the resulting algorithm below.

Algorithm 1 $\mathcal{R}[G, \gamma]$: PRG compression of $\mathcal{R}$

Input: $x \in X$, $\varepsilon, \gamma > 0$; seeded PRG $G\colon \{0,1\}^\ell \to \{0,1\}^t$; t-samplable ε-DP randomizer $\mathcal{R}$.

```
J = e^ε ln(1/γ)
for j = 1, ..., J do
    Sample a random seed s ∈ {0,1}^ℓ.
    y = R_∅(G(s))
    Sample b from Bernoulli( Pr[R(x)=y] / (e^ε Pr_{r~{0,1}^t}[R_∅(r)=y]) )
    if b == 1 then
        BREAK
    end if
end for
Send s
```

Naturally, the output of this randomizer can be decompressed by applying $\mathcal{R}_\emptyset \circ G$ to it. It is also clear that the communication cost of the algorithm is ℓ bits.

Next we describe the general condition on the PRG G that suffices for ensuring that the algorithm that outputs a random seed with correct probability is differentially private.

Lemma 3.2. *For a t-samplable deletion ε-DP local randomizer $\mathcal{R}\colon X \to Y$ and $G\colon \{0,1\}^\ell \to \{0,1\}^t$, let $\mathcal{D}$ denote*

the following family of tests which take $r' \in \{0,1\}^t$ as an input:

$$\left\{ \texttt{ind}\left(\frac{\mathbf{Pr}[\mathcal{R}(x) = \mathcal{R}_\emptyset(r')]}{\mathbf{Pr}_{r\sim\{0,1\}^t}[\mathcal{R}_\emptyset(r) = \mathcal{R}_\emptyset(r')]} \geq \theta \right) \;\middle|\; \begin{array}{l} x \in X, \\ \theta \in [0, e^\varepsilon] \end{array} \right\},$$

where ind *denotes the $\{0,1\}$ indicator function of a condition. If G β-fools $\mathcal{D}$ for $\beta < 1/(2e^\varepsilon)$ then $R[G]$ is a deletion $(\varepsilon + 2e^\varepsilon\beta)$-DP local randomizer. Furthermore, for every $\gamma > 0$, $\mathcal{R}[G,\gamma]$ is a deletion $(\varepsilon + 2e^\varepsilon\beta)$-DP local randomizer.*

Unlike the preservation of privacy, conditions on the PRG under which we can ensure that the utility of $\mathcal{R}$ is preserved depend on the application. Here we describe a general result that relies only on the efficiency of the randomizer to establish computational indistinguishability of the output of our compressed randomizer from the output of the original one.

Lemma 3.3. *Let $\mathcal{R}$ be a deletion ε-DP t-samplable local randomizer, let $G\colon \{0,1\}^\ell \to \{0,1\}^t$ be (T,β)-PRG. Let $T(\mathcal{R},G,\gamma)$ denote the running time of $\mathcal{R}[G,\gamma]$ and assume that $T > T(\mathcal{R},G,\gamma)$. Then for all $x \in X$, $\mathcal{R}_\emptyset(G(\mathcal{R}[G,\gamma](x)))$ is (T',β')-computationally indistinguishable from $\mathcal{R}(x)$, where $\beta' = \gamma + e^\varepsilon \ln(1/\gamma)\beta$ and $T' = T - T(\mathcal{R},G,\gamma)$.*

As a direct corollary of Lemmas 3.2 and 3.3 we obtain a general way to compress efficient LDP randomizers.

Theorem 3.4. *Let $\mathcal{R}$ be a deletion ε-DP t-samplable local randomizer, let $G\colon \{0,1\}^\ell \to \{0,1\}^t$ be (T,β)-PRG for $\beta < 1/(2e^\varepsilon)$. Let $T(\mathcal{R},G,\gamma)$ be the running time of $\mathcal{R}[G,\gamma]$ and assume that $T > T(\mathcal{R},G,\gamma)$. Then $\mathcal{R}[G,\gamma]$ is a deletion $(\varepsilon + 2e^\varepsilon\beta)$-DP local randomizer and for all $x \in X$, $\mathcal{R}_\emptyset(G(\mathcal{R}[G,\gamma](x)))$ is (T',β')-computationally indistinguishable from $\mathcal{R}(x)$, where $\beta' = \gamma + e^\varepsilon \ln(1/\gamma)\beta$ and $T' = T - T(\mathcal{R},G,\gamma)$.*

By plugging an exponentially strong PRG G into Theorem 3.4 we obtain that if an LDP protocol based on $\mathcal{R}$ runs in time T then its communication can be compressed to $O(\log(T + T(\mathcal{R},G,\gamma))$ with negligible effect on privacy and utility. We also remark that even without making any assumptions on G, $\mathcal{R}[G,\gamma]$ satisfies 2ε-DP. In other words, failure of the PRG does not lead to a significant privacy violation, beyond the degradation of the privacy parameter ε by a factor of two.

Lemma 3.5. *Let $\mathcal{R}$ be a deletion ε-DP t-samplable local randomizer, let $G\colon \{0,1\}^\ell \to \{0,1\}^t$ be an arbitrary function. Then $\mathcal{R}[G,\gamma]$ is a deletion 2ε-DP local randomizer.*

Our approach extends in a natural way to (ε,δ)-DP as well as to the replacement model of differential privacy. We defer the proof of the following to SM.

Theorem 3.6. *Let $\mathcal{R}$ be a deletion (replacement) (ε,δ)-DP t-samplable local randomizer, let $G\colon \{0,1\}^\ell \to \{0,1\}^t$ be (T,β)-PRG for $\beta < 1/(2e^\varepsilon)$. Let $T(\mathcal{R},G,\gamma)$ is the running time of $\mathcal{R}[G,\gamma]$ and assume that $T > T(\mathcal{R},G,\gamma)$. Then $\mathcal{R}[G,\gamma]$ is a deletion (resp. replacement) $(\varepsilon+2e^\varepsilon\beta, e^{O(\varepsilon)}\delta)$-DP local randomizer.*

4 Frequency Estimation

In this section we apply our approach to the problem of frequency estimation over a discrete domain. In this problem on domain $X = [k]$, the goal is to estimate the frequency of each element $j \in [k]$ in the dataset. Namely, for $S = (x_1,\ldots,x_n) \in X^n$ we let $c(S) \in \{0,\ldots,n\}^k$ be the vector of the counts of each of the elements in S: $c(S)_j = |\{i \mid x_i = j\}|$. In the frequency estimation problem the goal is to design a local randomizer and a decoding/aggregation algorithm that outputs a vector $\tilde{c}$ that is close to $c(S)$. Commonly studied metrics are (the expected) ℓ_∞, ℓ_1 and ℓ_2 norms of $\tilde{c} - c(S)$. In most regimes of interest, n is large enough and all these errors are essentially determined by the variance of the estimate of each count produced by the randomizer and therefore the choice of the metric does not affect the choice of the algorithm.

The randomizer used in the RAPPOR algorithm (Erlingsson et al., 2014) is defined by two parameters α_0 and α_1. The algorithm first converts the input j to the indicator vector of j (also referred to as one-hot encoding). It then randomizes each bit in this encoding: if the bit is 0 then 1 is output with probability α_0 (and 0 with probability $1-\alpha_0$) and if the bit is 1 then 1 is output with probability α_1.

For deletion privacy the optimal error is achieved by a symmetric setting $\alpha_0 = 1/(e^\varepsilon+1)$ and $\alpha_1 = e^\varepsilon/(e^\varepsilon+1)$ (Erlingsson et al., 2020). This makes the algorithm equivalent to applying the standard binary randomized response to each bit. A simple analysis shows that this results in the standard deviation of each count being $\frac{\sqrt{n}e^{\varepsilon/2}}{e^\varepsilon-1}$ (Erlingsson et al., 2014; Wang et al., 2017). For replacement privacy the optimal error is achieved by an asymmetric version in which $\alpha_0 = 1/(e^\varepsilon+1)$ but $\alpha_1 = 1/2$. The resulting standard deviation for each count is dominated by $\frac{2\sqrt{n}e^{\varepsilon/2}}{e^\varepsilon-1}$ (Wang et al., 2017). (We remark that several works analyze the symmetric RAPPOR algorithm in the replacement privacy. This requires setting $\alpha_0 = (1-\alpha_1) = 1/(e^{\varepsilon/2}+1)$ resulting in a substantially worse algorithm than the asymmetric version).

Note that the resulting encoding has $\approx n/(e^\varepsilon+1)$ ones. A closely-related Subset Selection algorithm (Wang et al., 2016; Ye & Barg, 2018) maps inputs to bit vectors of length k with exactly $\lceil \approx n/(e^\varepsilon+1)\rceil$ ones (that can be thought of as a subset of $[k]$). An input j is mapped with probability $\approx 1/2$ to a random subset that contains j and with probability $\approx 1/2$ to a random subset that does not. This results in

essentially the same marginal distributions over individual bits and variance bounds as asymmetric RAPPOR.

4.1 Pairwise-independent RAPPOR

While we can use our general result to compress communication in RAPPOR, in this section we exploit the specific structure of the randomizer. Specifically, to obtain a Bernoulli random variable with bias α_0 we will use a finite field $X = \mathbb{F}_p$ of size p such that $\alpha_0 p$ is an integer (or, in general, sufficiently close to an integer) and p is a prime larger than k. This allows us to treat inputs in $[k]$ as non-zero elements of $\mathbb{F}_p$. We will associate all elements of the field that are smaller (in the regular order over integers) than $\alpha_0 p$ with 1 and the rest with 0. We denote this indicator function of the event $z < \alpha_0 p$ by $\texttt{bool}(z)$. Now for a randomly and uniformly chosen element $z \in \mathbb{F}_p$, we have that $\texttt{bool}(z)$ is distributed as a Bernoulli random variable with bias α_0.

As mentioned we will, associate each index $j \in [k]$ with the element j in $\mathbb{F}_p$. We can describe an affine function ϕ over $\mathbb{F}_p$ using its 2 coefficients: ϕ_0 and ϕ_1 and for $z \in \mathbb{F}_p$ we define $\phi(z) = \phi_0 + z\phi_1$, where addition and multiplication are in the field $\mathbb{F}_p$. Each such function encodes a vector in $\mathbb{F}_p^k$ as $\phi([k]) := \phi(1), \phi(2), \ldots, \phi(k)$. Let $\Phi := \{\phi \mid \phi \in \mathbb{F}_p^2\}$ be the family of all such functions. For a randomly chosen function from this family the values of the function on two distinct non-zero values are uniformly distributed and pairwise-independent: for any $j_1 \neq j_2 \in [k]$ and $a_1, a_2 \in \mathbb{F}_p$ we have that

$$\mathop{\mathbf{Pr}}_{\phi\sim\Phi}[\phi(j_1) = a_1 \text{ and } \phi(j_2) = a_2] = \mathop{\mathbf{Pr}}_{\phi\sim\Phi}[\phi(j_1) = a_1] \cdot \mathop{\mathbf{Pr}}_{\phi\sim\Phi}[\phi(j_2) = a_2] = \frac{1}{p^2}.$$

In particular, if we use the encoding of ϕ as a boolean vector

$$\texttt{bool}(\phi[k]) := \texttt{bool}(\phi(1)), \texttt{bool}(\phi(2)), \ldots, \texttt{bool}(\phi(k))$$

then we have that for $\phi \sim \Phi$ and any $j_1 \neq j_2 \in [k]$, $\texttt{bool}(\phi(j_1))$ and $\texttt{bool}(\phi(j_2))$ are independent Bernoulli random variables with bias α_0.

Finally, for every index $j \in [k]$ and bit $b \in \{0, 1\}$ we denote the set of functions ϕ whose encoding has bit b in position j by $\Phi_{j,b}$:

$$\Phi_{j,b} := \{\phi \in \Phi \mid \texttt{bool}(\phi(j)) = b\}. \tag{1}$$

We can now describe the randomizer, which we refer to as Pairwise-Independent (PI) RAPPOR for general $\alpha_1 > \alpha_0$.

Algorithm 2 PI-RAPPOR randomizer

Input: An index $j \in [k]$, $0 < \alpha_0 < \alpha_1 < 1$, prime $p \geq k + 1$ s.t. $\alpha_0 p \in \mathbb{N}$

1: Sample b from $\text{Bernoulli}(\alpha_1)$
2: Sample randomly ϕ from $\Phi_{j,b}$ defined in eq. (1)
3: Send ϕ

The server side of the frequency estimation with pairwise-independent RAPPOR consists of a decoding step that converts ϕ to $\texttt{bool}(\phi[k])$ and then the same debiasing and aggregation as for the standard RAPPOR. We describe it as a frequency oracle to emphasize that each count can be computed individually.

Algorithm 3 Server-side frequency for PI-RAPPOR

Input: $0 < \alpha_0 < \alpha_1 < 1$, k, index $j \in [k]$ and prime $p > k$. Reports $\phi^1, \ldots, \phi^n$ from n users.

1: $\text{sum} = 0$
2: **for** $i \in [n]$ **do**
3: $\quad \text{sum} += \texttt{bool}(\phi^i(j))$
4: **end for**
5: $\tilde{c}_j = \frac{\text{sum} - \alpha_0 n}{\alpha_1 - \alpha_0}$
6: Return $\tilde{c}_j$

We start by establishing several general properties of PI-RAPPOR. First we establish that the privacy guarantees for PI-RAPPOR are identical to those of RAPPOR.

Lemma 4.1. *PI-RAPPOR randomizer (Alg. 2) is deletion* $\max\left\{\frac{\alpha_1}{\alpha_0}, \frac{1-\alpha_0}{1-\alpha_1}\right\}$*-DP and replacement* $\frac{\alpha_1(1-\alpha_0)}{\alpha_0(1-\alpha_1)}$*-DP.*

Second we establish that the utility guarantees of PI-RAPPOR are identical to those of RAPPOR. This follows directly from the fact that the utility is determined by the variance of the estimate of each individual count in each user's contribution. The variance of the estimate of $c(S)_j$ is a sum of $c(S)_j$ variances for randomization of 1 and $n - c(S)_j$ variances of randomization of 0. These variances are identical for RAPPOR and PI-RAPPOR leading to identical *exact* bounds.

Lemma 4.2. *For any dataset $S \in [k]^n$, the estimate $\tilde{c}$ computed by PI-RAPPOR algorithm (Algs. 2,3) satisfies $\mathbf{E}[\tilde{c}] = c(S)$, and for all $j \in [k]$,*

$$\mathbf{Var}[\tilde{c}_j] = c(S)_j \frac{1 - \alpha_0 - \alpha_1}{\alpha_1 - \alpha_0} + n\frac{\alpha_0(1-\alpha_0)}{(\alpha_1 - \alpha_0)^2}$$

For the symmetric case $\alpha_0 = 1 - \alpha_1$ this simplifies to $\mathbf{Var}[\tilde{c}_j] = n\frac{\alpha_0(1-\alpha_0)}{(1-2\alpha_0)^2}$. In addition, the expected ℓ_2 squared error is

$$\mathbf{E}\left[\|\tilde{c} - c(S)\|_2^2\right] = n\frac{1 - \alpha_0 - \alpha_1}{\alpha_1 - \alpha_0} + nk\frac{\alpha_0(1-\alpha_0)}{(\alpha_1 - \alpha_0)^2}.$$

Plugging $\alpha_0 = 1/(e^\varepsilon + 1)$ and $\alpha_1 = 1/2$ for replacement privacy and $\alpha_0 = 1 - \alpha_1 = 1/(e^\varepsilon + 1)$ for deletion privacy gives the following utility bounds for ε-DP versions of PI-RAPPOR.

Corollary 4.3. *For any $\varepsilon > 0$ and a setting of p that ensures that $p/(e^\varepsilon + 1) \in \mathbb{N}$ we have that PI-RAPPOR for $\alpha_0 = 1 - \alpha_1 = 1/(e^\varepsilon + 1)$ satisfies deletion ε-DP and for every dataset $S \in [k]^n$, the estimate $\tilde{c}$ computed by PI-RAPPOR satisfies: $\mathbf{E}[\tilde{c}] = c(S)$, for all $j \in [k]$, $\mathbf{Var}[\tilde{c}_j] = n\frac{e^\varepsilon}{(e^\varepsilon-1)^2}$ and $\mathbf{E}\left[\|\tilde{c} - c(S)\|_2^2\right] = nk\frac{e^\varepsilon}{(e^\varepsilon-1)^2}$.*

Corollary 4.4. *For any $\varepsilon > 0$ and a setting of p that ensures that $p/(e^\varepsilon + 1) \in \mathbb{N}$ we have that PI-RAPPOR for $\alpha_0 = 1/(e^\varepsilon + 1)$ and $\alpha_1 = 1/2$ is replacement ε-DP and for every dataset $S \in [k]^n$, the estimate $\tilde{c}$ computed by PI-RAPPOR satisfies: $\mathbf{E}[\tilde{c}] = c(S)$, for all $j \in [k]$, $\mathbf{Var}[\tilde{c}_j] = c(S)_j + n\frac{4e^\varepsilon}{(e^\varepsilon-1)^2}$ and $\mathbf{E}\left[\|\tilde{c} - c(S)\|_2^2\right] = n + nk\frac{4e^\varepsilon}{(e^\varepsilon-1)^2}$.*

Finally, we analyze the computational and communication cost of PI-RAPPOR. Clearly, the communication cost of PI-RAPPOR is $2\lceil \log_2 p \rceil$ bits. In addition, it is not hard to see that all computations performed by PI-RAPPOR can be implemented in essentially the same time as single multiplication in $\mathbb{F}_p$. The analysis of the running time of decoding and aggregation is similarly straightforward since decoding every bit of message takes time that is dominated by the time of a single multiplication in $\mathbb{F}_p$.

We defer the details to SM. As these complexities depend on $\log p$ we also need to discuss the choice of p. It is not hard to show (see SM for details) that $p \geq c_1 \max\{k, e^\varepsilon, 1/\varepsilon\}$ for a sufficiently large constant c_1 ensures that PI-RAPPOR will have essentially the same guarantees as RAPPOR. This means that the communication cost of PI-RAPPOR is $2\log_2(\max\{k, e^\varepsilon, 1/\varepsilon\}) + O(1)$. Also we are typically interested in compression when $k \gg \max\{e^\varepsilon, 1/\varepsilon\}$ and in such case the communication cost is $2\log_2(k) + O(1)$.

5 Mean Estimation

In this section, we consider the problem of mean estimation in ℓ_2 norm, for ℓ_2-norm bounded vectors. Formally, each client has a vector $\mathbf{x}_i \in \mathbb{B}^d$, where $\mathbb{B}^d := \{\mathbf{x} \in \mathcal{R}^d \mid \|\mathbf{x}\|_2 \leq 1\}$. Our goal is to compute the mean of these vectors privately, and we measure our error in the ℓ_2 norm. In the literature this problem is often studied in the statistical setting where $\mathbf{x}_i$'s are sampled i.i.d. from some distribution supported on $\mathbb{B}^d$ and the goal is to estimate the mean of this distribution. In this setting, the expected squared ℓ_2 distance between the mean of the distribution and the mean of the samples is at most $1/n$ and is dominated by the privacy error in the regime that we are interested in ($\varepsilon < d$).

In the absence of communication constraints and $\varepsilon < d$, the optimal ε-LDP protocols for this problem achieve an expected squared ℓ_2 error of $\Theta(\frac{d}{n\min(\varepsilon,\varepsilon^2)})$ (Duchi et al., 2018; Duchi & Rogers, 2019). Here and in the rest of the section we focus on the replacement DP both for consistency with existing work and since for this problem the dependence on ε is linear (when $1 < \varepsilon < d$) and thus the difference between replacement and deletion is less important.

If one is willing to relax to (ε, δ) or concentrated differential privacy (Dwork & Rothblum, 2016; Bun & Steinke, 2016; Mironov, 2017) guarantees, then standard Gaussian noise addition achieves the asymptotically optimal bound. When $\varepsilon \leq 1$, the randomizer of Duchi et al. (2018) (which we refer to as `PrivHS`) also achieves the optimal $O(\frac{d}{n\varepsilon^2})$ bound. Recent work of Erlingsson et al. (2020) gives a low-communication version of `PrivHS`. Specifically, in the context of federated optimization they show that `PrivHS` is equivalent to sending a single bit and a randomly and uniformly generated unit vector. This vector can be sent using a seed to a PRG. Bhowmick et al. (2019) describe the `PrivUnit` algorithm that achieves the optimal bound also when $\varepsilon > 1$. Unfortunately, `PrivUnit` has high communication cost of $\Omega(d)$.

By applying Theorem 3.4 to `PrivUnit` or Gaussian noise addition, we can immediately obtain a low communication algorithm with negligible effect on privacy and utility. This gives us an algorithm that communicates a single seed, and has the asymptotically optimal privacy utility trade-off. Implementing `PrivUnit` requires sampling uniformly from a spherical cap $\{\mathbf{v} \mid \|\mathbf{v}\|_2 = 1, \langle \tilde{\mathbf{x}}, \mathbf{v} \rangle \geq \alpha\}$ for $\alpha \approx \sqrt{\varepsilon/d}$. Using standard techniques this can be done with high accuracy using $\tilde{O}(d)$ random bits and $\tilde{O}(d)$ time. Further, for every $\mathbf{x}$ the resulting densities can be computed easily given the surface area of the cap. Overall rejection sampling can be computed in $\tilde{O}(d)$ time. Thus this approach to compression requires time $\tilde{O}(e^\varepsilon d)$. This implies that given an exponentially strong PRG G, we can compress `PrivUnit` to $O(\log(dn) + \varepsilon)$ bits with negligible effects on utility and privacy. In most settings of interest, the computational cost $\tilde{O}(e^\varepsilon d)$ is not much larger than the typical cost of computing the vector itself, e.g. by back propagation in the case of gradients of neural networks (e.g. $\varepsilon = 8$ requires ≈ 3000 trials in expectation).

We can further reduce this computational overhead. We show a simple reduction from the general case of $\varepsilon > 1$ to a protocol for $\varepsilon' = \varepsilon/m$ that preserves asymptotic optimality, where $m \leq 2\varepsilon$ is an integer. The algorithm simply runs m copies of the ε'-DP randomizer and sends all the reports. The estimates produced from these reports are averaged by the server. This reduces the expected number of rejection sampling trials to $me^{\varepsilon/m}$. We describe the formal details of this simple reduction in SM.

This reduction allows one to achieve different trade-offs between computation, communication, and closeness to the accuracy of the original randomizer. As an additional ben-

efit, we no longer need an LDP randomizer that is optimal in the $\varepsilon > 1$ regime. We can simply use $m = \lceil \varepsilon \rceil$ and get an asymptotically optimal algorithm for $\varepsilon > 1$ from any algorithm that is asymptotically optimal for $\varepsilon' \in [1, 1/2]$. In particular, instead of `PrivUnit` we can use the low communication version of `PrivHS` from (Erlingsson et al., 2020). This bypasses the need for our compression algorithm and makes the privacy guarantees unconditional.

Remark 5.1. *We remark that the compression of* `PrivUnit` *can be easily made unconditional. The reference distribution ρ of* `PrivUnit` *is uniform over a sphere of some radius $B(d,\varepsilon) = O(\sqrt{d/\min(\varepsilon,\varepsilon^2)})$. It is not hard to see that both the privacy and utility guarantees of* `PrivUnit` *are preserved by any PRG G which preserves $\mathbf{Pr}_{\mathbf{v}\sim\rho}[\langle \mathbf{x}, \mathbf{v}\rangle \geq \theta]$ for every vector $\mathbf{x}$ sufficiently well (up to some $1/poly(d, n, e^\varepsilon/\varepsilon)$ accuracy). Note that these tests are halfspaces and have VC dimension d. Therefore by the standard ϵ-net argument, a random sample of size $O(dB(d,\varepsilon)/\gamma^2)$ from the reference distribution will, with high probability, give a set of points S that γ-fools the test (for any $\gamma > 0$). By choosing $\gamma = 1/poly(d, n, e^\varepsilon/\varepsilon)$ we can ensure that the effect on privacy and accuracy is negligible (relative to the error introduced due to privacy). Thus one can compress the communication to $\log_2(|S|) = O(\log(dn/\varepsilon) + \varepsilon)$ bits unconditionally (with negligible effect on accuracy and privacy).*

While `PrivUnit`, `SQKR` and the repeated version of `PrivHS` are asymptotically optimal, the accuracy they achieve in practice may be different. Therefore we empirically compare these algorithms. In our first comparison we consider four algorithms. The `PrivHS` algorithm outputs a vector whose norm is fully defined by the parameters d, ε: the output vector has norm $B(d,\varepsilon) = \frac{e^\varepsilon+1}{e^\varepsilon-1}\frac{\sqrt{\pi}}{2}\frac{d\Gamma(\frac{d-1}{2}+1)}{\Gamma(\frac{d}{2}+1)}$. The variance is then easily seen to be $(B^2+1\pm 2B)/n$ when averaging over n samples. For large dimensional settings of interest, $B \gg 1$ so this expression is very well approximated by B^2/n and we use this value as a proxy. For `SQKR`, we use the implementation provided by the authors at (Kas) (specifically, second version of the code that optimizes some of the parameters). We show error bars for the empirical squared error based on 20 trials.

The `PrivUnit` algorithm internally splits its privacy budget ε into two parts $\varepsilon_0, \varepsilon_1 = 1 - \varepsilon_0$. As in the case of `PrivHS`, the output of `PrivUnit` (for fixed $d, \varepsilon_0, \varepsilon_1$) has a fixed squared norm which is the proxy we use for variance. We first consider the default split used in the experiments in (Bhowmick et al., 2019) and refer to it as `PrivUnit`. In addition, we optimize the splitting so as to minimize the variance proxy, by evaluating the expression for the variance proxy as a function of the $\theta = \varepsilon_0/\varepsilon$, for 101 values of $\theta = 0.00, 0.01, 0.02, \ldots, 0.99, 1.0$. We call this algorithm `PrivUnitOptimized`. Note that since we are optimizing

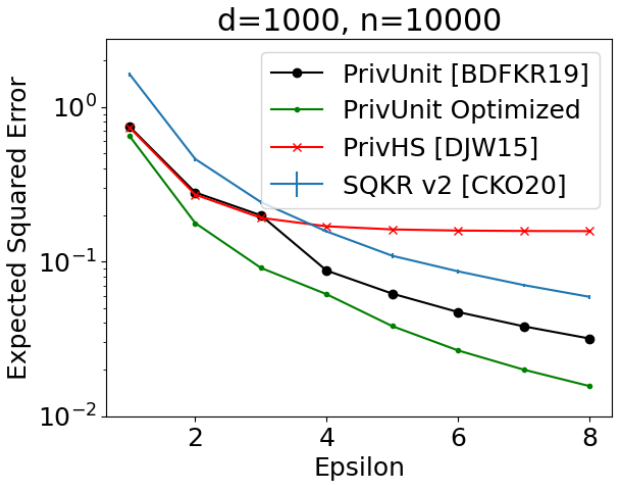

Figure 1. Expected ℓ_2^2 error of mechanisms `PrivHS`, `PrivUnit`, `PrivUnitOptimized` and `SQKR` for values of ε between 1 and 8.

θ to minimize the norm proxy, this optimization is data-independent and need only be done once for a fixed ε. For both variants of `PrivUnit`, we use the norm proxy in our evaluation; as discussed above, in high-dimensional settings of interest, the proxy is nearly exact.

Figure 1 compares the expected squared error of these algorithms for $d = 1,000$, $n = 10,000$ and ε taking integer values from 1 to 8. These plots show both `PrivUnit` and `PrivUnitOptimized` are more accurate than `PrivHS` and `SQKR` in the whole range of parameters While `PrivHS` is competitive for small ε, it does not get better with ε for large ε. `SQKR` consistently has about $5\times$ higher expected squared error than `PrivUnitOptimized` and about $2.5\times$ higher error compared to `PrivUnit`. Thus in the large ε regime, the ability to compress `PrivUnitOptimized` gives a $5\times$ improvement in error compared to previous compressed algorithms. We also observe that `PrivUnitOptimized` is noticeably better than `PrivUnit`. Our technique being completely general, it will apply losslessly to any other better local randomizers that may be discovered in the future.

As discussed earlier, one way to reduce the computational cost of compressed `PrivUnitOptimized` is to use the reduction described above. For instance, instead of running `PrivUnitOptimized` with $\varepsilon = 8$, we may run it twice with $\varepsilon = 4$ and average the results on the server. Asymptotically, this gives the same expected squared error and we empirically evaluate the effect of such splitting on the expected error in SM.

References

An implementation of kashine based mean estimation scheme. https://github.com/WeiNingChen/Kashin-mean-estimation. Accessed: 2021-02-17.

Abadi, M., Chu, A., Goodfellow, I. J., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pp. 308–318, 2016.

Acharya, J. and Sun, Z. Communication complexity in locally private distribution estimation and heavy hitters. *arXiv preprint arXiv:1905.11888*, 2019.

Acharya, J., Sun, Z., and Zhang, H. Hadamard response: Estimating distributions privately, efficiently, and with little communication. volume 89 of *Proceedings of Machine Learning Research*, pp. 1120–1129. PMLR, 16–18 Apr 2019.

Agarwal, N., Suresh, A. T., Yu, F. X. X., Kumar, S., and McMahan, B. cpsgd: Communication-efficient and differentially-private distributed sgd. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 31, pp. 7564–7575. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/21ce689121e39821d07d04faab328370-Paper.pdf.

Alistarh, D., Grubic, D., Li, J., Tomioka, R., and Vojnovic, M. Qsgd: Communication-efficient sgd via gradient quantization and encoding. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30, pp. 1709–1720. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/6c340f25839e6acdc73414517203f5f0-Paper.pdf.

Apple's Differential Privacy Team. Learning with privacy at scale. *Apple Machine Learning Journal*, 1(9), December 2017.

Balle, B., Bell, J., Gascón, A., and Nissim, K. The privacy blanket of the shuffle model. In Boldyreva, A. and Micciancio, D. (eds.), *Advances in Cryptology – CRYPTO 2019*, pp. 638–667, Cham, 2019. Springer International Publishing. ISBN 978-3-030-26951-7.

Bassily, R. and Smith, A. Local, private, efficient protocols for succinct histograms. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pp. 127–135, 2015.

Bassily, R., Smith, A., and Thakurta, A. Private empirical risk minimization, revisited. *CoRR*, abs/1405.7085, 2014. URL http://arxiv.org/abs/1405.7085.

Bassily, R., Nissim, K., Stemmer, U., and Thakurta, A. Practical locally private heavy hitters. *Journal of Machine Learning Research*, 21(16):1–42, 2020.

Bhowmick, A., Duchi, J., Freudiger, J., Kapoor, G., and Rogers, R. Protection against reconstruction and its applications in private federated learning, 2019.

Bittau, A., Erlingsson, U., Maniatis, P., Mironov, I., Raghunathan, A., Lie, D., Rudominer, M., Kode, U., Tinnes, J., and Seefeld, B. Prochlo: Strong privacy for analytics in the crowd. In *Proceedings of the 26th Symposium on Operating Systems Principles*, SOSP '17, pp. 441–459, 2017.

Bun, M. and Steinke, T. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Proceedings, Part I, of the 14th International Conference on Theory of Cryptography - Volume 9985*, pp. 635–658, Berlin, Heidelberg, 2016. Springer-Verlag. ISBN 9783662536407. doi: 10.1007/978-3-662-53641-4_24. URL https://doi.org/10.1007/978-3-662-53641-4_24.

Bun, M., Nelson, J., and Stemmer, U. Heavy hitters and the structure of local privacy. *ACM Transactions on Algorithms (TALG)*, 15(4):1–40, 2019.

Chen, W.-N., Kairouz, P., and Özgür, A. Breaking the communication-privacy-accuracy trilemma. *arXiv preprint arXiv:2007.11707*, 2020.

Cheu, A., Smith, A., Ullman, J., Zeber, D., and Zhilyaev, M. Distributed differential privacy via shuffling. In Ishai, Y. and Rijmen, V. (eds.), *Advances in Cryptology – EUROCRYPT 2019*, pp. 375–403, Cham, 2019. Springer International Publishing. ISBN 978-3-030-17653-2.

Ding, B., Kulkarni, J., and Yekhanin, S. Collecting telemetry data privately. In *31st Conference on Neural Information Processing Systems (NIPS)*, pp. 3574–3583, 2017.

Duchi, J. and Rogers, R. Lower bounds for locally private estimation via communication complexity. In Beygelzimer, A. and Hsu, D. (eds.), *Proceedings of the Thirty-Second Conference on Learning Theory*, volume 99 of *Proceedings of Machine Learning Research*, pp. 1161–1191, Phoenix, USA, 25–28 Jun 2019. PMLR. URL http://proceedings.mlr.press/v99/duchi19a.html.

Duchi, J. C., Jordan, M. I., and Wainwright, M. J. Minimax optimal procedures for locally private estimation. *Journal of the American Statistical Association*, 113(521):182–201, 2018.

Dwork, C. and Roth, A. *The Algorithmic Foundations of Differential Privacy*, volume 9. 2014. URL http://dx.doi.org/10.1561/0400000042.

Dwork, C. and Rothblum, G. N. Concentrated differential privacy. *ArXiv*, abs/1603.01887, 2016.

Dwork, C., McSherry, F., Nissim, K., and Smith, A. Calibrating noise to sensitivity in private data analysis. In *TCC*, pp. 265–284, 2006.

Erlingsson, Ú., Pihur, V., and Korolova, A. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, pp. 1054–1067, 2014.

Erlingsson, U., Feldman, V., Mironov, I., Raghunathan, A., Talwar, K., and Thakurta, A. Amplification by shuffling: From local to central differential privacy via anonymity. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '19, pp. 2468–2479, USA, 2019. Society for Industrial and Applied Mathematics.

Erlingsson, U., Feldman, V., Mironov, I., Raghunathan, A., Song, S., Talwar, K., and Thakurta, A. Encode, shuffle, analyze privacy revisited: Formalizations and empirical evaluation. 2020.

Evfimievski, A. V., Gehrke, J., and Srikant, R. Limiting privacy breaches in privacy preserving data mining. In *PODS*, pp. 211–222, 2003.

Faghri, F., Tabrizian, I., Markov, I., Alistarh, D., Roy, D. M., and Ramezani-Kebrya, A. Adaptive gradient quantization for data-parallel sgd. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Feldman, V., Guzman, C., and Vempala, S. Statistical query algorithms for mean vector estimation and stochastic convex optimization. *CoRR*, abs/1512.09170, 2015. URL http://arxiv.org/abs/1512.09170. Extended abstract in SODA 2017.

Feldman, V., McMillan, A., and Talwar, K. Hiding among the clones: A simple and nearly optimal analysis of privacy amplification by shuffling. *CoRR*, abs/2012.12803, 2020. URL https://arxiv.org/abs/2012.12803.

Gandikota, V., Kane, D., Maity, R. K., and Mazumdar, A. vqsgd: Vector quantized stochastic gradient descent. *arXiv preprint arXiv:1911.07971*, 2019.

Girgis, A. M., Data, D., Diggavi, S., Kairouz, P., and Suresh, A. T. Shuffled model of federated learning: Privacy, communication and accuracy trade-offs, 2020.

Hsu, J., Khanna, S., and Roth, A. Distributed private heavy hitters. In *International Colloquium on Automata, Languages, and Programming*, pp. 461–472. Springer, 2012.

Kairouz, P., Bonawitz, K., and Ramage, D. Discrete distribution estimation under local privacy. *arXiv preprint arXiv:1602.07387*, 2016.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., D'Oliveira, R. G. L., Rouayheb, S. E., Evans, D., Gardner, J., Garrett, Z., Gascón, A., Ghazi, B., Gibbons, P. B., Gruteser, M., Harchaoui, Z., He, C., He, L., Huo, Z., Hutchinson, B., Hsu, J., Jaggi, M., Javidi, T., Joshi, G., Khodak, M., Konečný, J., Korolova, A., Koushanfar, F., Koyejo, S., Lepoint, T., Liu, Y., Mittal, P., Mohri, M., Nock, R., Özgür, A., Pagh, R., Raykova, M., Qi, H., Ramage, D., Raskar, R., Song, D., Song, W., Stich, S. U., Sun, Z., Suresh, A. T., Tramèr, F., Vepakomma, P., Wang, J., Xiong, L., Xu, Z., Yang, Q., Yu, F. X., Yu, H., and Zhao, S. Advances and open problems in federated learning, 2019.

Luby, M., Luby, M. G., and Wigderson, A. *Pairwise independence and derandomization*, volume 4. Now Publishers Inc, 2006.

Lyubarskii, Y. and Vershynin, R. Uncertainty principles and vector quantization. *Information Theory, IEEE Transactions on*, 56(7):3491–3501, 2010.

Mayekar, P. and Tyagi, H. Limits on gradient compression for stochastic optimization. In *2020 IEEE International Symposium on Information Theory (ISIT)*, pp. 2658–2663, 2020. doi: 10.1109/ISIT44484.2020.9174075.

McGregor, A., Mironov, I., Pitassi, T., Reingold, O., Talwar, K., and Vadhan, S. The limits of two-party differential privacy. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pp. 81–90, 2010.

McMahan, H. B., Ramage, D., Talwar, K., and Zhang, L. Learning differentially private recurrent language models. In *6th International Conference on Learning Representations, ICLR 2018*, 2018.

Menezes, A. J., Van Oorschot, P. C., and Vanstone, S. A. *Handbook of applied cryptography*. CRC press, 2018.

Mironov, I. Rényi differential privacy. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, pp. 263–275, 2017. doi: 10.1109/CSF.2017.11.

Mironov, I., Pandey, O., Reingold, O., and Vadhan, S. Computational differential privacy. In Halevi, S. (ed.), *Advances in Cryptology - CRYPTO 2009*, pp. 126–142, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. ISBN 978-3-642-03356-8.

Mishra, N. and Sandler, M. Privacy via pseudorandom sketches. In *Proceedings of the Twenty-Fifth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '06, pp. 143–152, New York, NY, USA, 2006. Association for Computing Machinery. ISBN 1595933182. doi: 10.1145/1142351.1142373. URL https://doi.org/10.1145/1142351.1142373.

Nisan, N. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12(4):449–461, 1992.

Suresh, A. T., Yu, F. X., Kumar, S., and McMahan, H. B. Distributed mean estimation with limited communication. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 3329–3337, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. URL http://proceedings.mlr.press/v70/suresh17a.html.

Wang, S., Huang, L., Wang, P., Nie, Y., Xu, H., Yang, W., Li, X.-Y., and Qiao, C. Mutual information optimally local private discrete distribution estimation. *arXiv preprint arXiv:1607.08025*, 2016.

Wang, T., Blocki, J., Li, N., and Jha, S. Locally differentially private protocols for frequency estimation. In *26th USENIX Security Symposium (USENIX Security 17)*, pp. 729–745, Vancouver, BC, August 2017. USENIX Association. ISBN 978-1-931971-40-9. URL https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/wang-tianhao.

Warner, S. L. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309):63–69, 1965.

Ye, M. and Barg, A. Optimal schemes for discrete distribution estimation under locally differential privacy. *IEEE Transactions on Information Theory*, 64(8):5662–5676, 2018.