Lossless Compression of Efficient Private Local Randomizers

Vitaly Feldman
Apple

Kunal Talwar
Apple

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 large domain or learning a high-dimensional model). This has led to significant efforts on reducing the communication cost of LDP algorithms.

At the same time LDP reports are known to have relatively little information about the user's data due to randomization. Several schemes are known that exploit this fact to design low-communication versions of LDP algorithm but all of them do so at the expense of a significant loss in utility. 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 the message can be compressed to the size of the server's pseudo-random generator seed. More generally, we relate the properties of an LDP randomizer to the power of a pseudo-random generator that suffices for compressing the LDP randomizer. 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 $\mathcal{R}$).

The concept of a local randomizer dates back to the work of Warner [War65] where it was used to encourage truthfulness in surveys. In the context of modern data analysis it was introduced by Evfimievski et al. [EGS03] and then related to differential privacy in the seminal work of Dwork et al. [DMNS06]. Local randomizers are also used for collection of statistics and machine learning in several industrial applications [EPK14; App17; DKY17]. Practical applications such as building a histogram over a large domain or training a model with millions of parameters [MRTZ18], 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 trade-off 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 compressing outputs of LDP mechanisms. We focus on the $\varepsilon > 1$ regime, that has been motivated by recent privacy amplification techniques based on anonymization and shuffling of LDP reports [BEMMRLRKTS17; EFMRTT19; CSUZZ19; BBGN19; FMT20]. 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 compressible using standard tools from information theory. However, standard compression algorithms do not necessarily preserve privacy. Bassily and Smith [BS15] and Bun et al. [BNS19] 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 which makes the approach impractical in the $\varepsilon > 1$ regime. We remark that it is crucial to use a compression technique that preserves the privacy guarantee since in some problems accuracy scales as $e^{-\varepsilon/2}$ when $\varepsilon > 1$. In addition, the central privacy guarantees resulting from amplification by shuffling also scale as $e^{-\varepsilon/2}$.

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 [EFMRSTT20] 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 and Sandler [MS06] 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 3.5 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

[Nis92], 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 [EPK14] (which has two versions depending on whether it is used with replacement or deletion privacy) and also by the closely related Subset Selection algorithm [WHWNXYLQ16; YB18]. 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 [LLW06]. 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. [CKÖ20]. It is however less accurate empirically and more involved than the algorithm of Bhowmick et al. [BDFKR19] that communicates a d dimensional vector. Using our general result we can losslessly compress the algorithm from [BDFKR19] 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 $\varepsilon < 1$ [DJW18; EFMRSTT20]. 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 and Sandler [MS06]. 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 [BS15; BNS19] as well as earlier work on private compression in the two-party setting [MMPRTV10]. 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 [BS15], 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 [BS15; BNS19] 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) [MPRV09] 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 [EPK14] and also by a closely-related Subset Selection algorithm [WHWNXYLQ16; YB18]. 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 [HKR12; EPK14; BS15; KBR16; WHWNXYLQ16; WBLJ17; YB18; ASZ19; AS19; BNS19; BNST20; CKÖ20].

A number of low-communication algorithms that achieve asymptotically optimal bounds in the $\varepsilon < 1$ regime are known [BS15; WBLJ17; ASZ19; AS19; BNST20; CKÖ20]. The first low-communication algorithm that achieves asymptotically optimal bounds in the $\varepsilon > 1$ regime is given in [WBLJ17]. 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 and Sun [AS19] and Chen et al. [CKÖ20] 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 [CKÖ20] show that these algorithms are ≈15-20% worse for $\varepsilon = 5$ than Subset Selection[1]). The constructions in [AS19; CKÖ20] 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) [BS15; BNST20; BNS19]. Frequency estimation is also closely 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 [YB18; ASZ19; AS19]. 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 [BST14; ACGMMTZ16] used in private federated learning [Kai+19]. 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 [ASYKM18; GDDKS20; CKÖ20; GKMM19] or without privacy [AGLTV17; FTMARRK20; SYKM17; GKMM19; MT20].

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 $\Theta(\frac{d}{n \min(\varepsilon,\varepsilon^2)})$ [DJW18; DR19]. When $\varepsilon \leq 1$, the randomizer of Duchi et al. [DJW18] also achieves the optimal $O(\frac{d}{n\varepsilon^2})$ bound. Recent work of Erlingsson et al. [EFMRSTT20] gives a low-communication version of this algorithm. Building on the approach in [DJW18], Bhowmick et al. [BDFKR19] 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. [FGV15] 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 $[-1, 1]^{O(d)}$ [LV10]. 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. [CKÖ20] 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 [DJW18; BDFKR19]. In addition, as we demonstrate empirically[2], the variance of the estimate resulting from this approach is nearly a factor of 5× larger for typical parameters of interest.

[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.

[2] Plots in [CKÖ20] also compare their algorithm with `PrivUnit` yet as their code at [Kas] shows and was confirmed by the authors, they implemented the algorithm from [DJW18] 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.

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 $|a|_+ := \max\{a, 0\}$ [DR14]. 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 [EFMRSTT20].

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 ability 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).

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 and Sandler [MS06] (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 $\varepsilon$-DP randomizer $\mathcal{R}$.
$J = e^\varepsilon \ln(1/\gamma)$
for $j = 1, \ldots, J$ do
    Sample a random seed $s \in \{0,1\}^\ell$.
    $y = \mathcal{R}_\emptyset(G(s))$
    Sample $b$ from Bernoulli $\left(\frac{\mathbf{Pr}[\mathcal{R}(x)=y]}{e^\varepsilon\,\mathbf{Pr}_{r\sim\{0,1\}^t}[\mathcal{R}_\emptyset(r)=y]}\right)$
    if $b == 1$ then
        BREAK
Send $s$
```

Naturally, the output of this randomizer can be decompressed by applying $G \circ \mathcal{R}_\emptyset$ 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:

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

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.*

Proof. We demonstrate that if $\mathcal{R}[G]$ is not a deletion $(\varepsilon + 2e^\varepsilon\beta)$-DP randomizer then there exists a test in $\mathcal{D}$ that distinguishes the output of G from true randomness that succeeds with probability at least β. To analyze the privacy guarantees of $\mathcal{R}[G]$ we let the reference distribution ρ_G be the uniform distribution over $\{0,1\}^\ell$. For brevity, for $y \in Y$ we denote the density ratio of $\mathcal{R}(x)$ to ρ at y by

$$\pi_x(y) := \frac{\mathbf{Pr}[\mathcal{R}(x) = y]}{\mathbf{Pr}_{r\sim\{0,1\}^t}[\mathcal{R}_\emptyset(r) = y]}.$$

Then, $\mathcal{R}[G]$ outputs a seed s with probability:

$$\mu_x(s) := \frac{\pi_x(\mathcal{R}_\emptyset(G(s)))}{\sum_{s'\in\{0,1\}^\ell} \pi_x(\mathcal{R}_\emptyset(G(s')))}.$$

By definition of our reference distribution, $\rho_G(s) = 2^{-\ell}$ for all s. Therefore

$$\frac{\mu_x(s)}{\rho_G(s)} = \frac{\mu_x(s)}{2^{-\ell}} = \frac{\pi_x(\mathcal{R}_\emptyset(G(s)))}{\mathbf{E}_{s'\sim\{0,1\}^\ell}[\pi_x(\mathcal{R}_\emptyset(G(s')))]}.$$

We observe that, by the fact that $\mathcal{R}(x)$ is ε-DP we have that $\pi_x(\mathcal{R}_\emptyset(G(s))) \in [e^{-\varepsilon}, e^\varepsilon]$. Therefore, to show that $\mathcal{R}[G]$ is $(\varepsilon + 2e^\varepsilon\beta)$-DP, it suffices to show that the denominator is in the range $[e^{-2e^\varepsilon\beta}, e^{2e^\varepsilon\beta}]$. To show this, we assume for the sake of contradiction that it is not true. Namely, that either

$$\mathop{\mathbf{E}}_{s'\sim\{0,1\}^\ell}[\pi_x(\mathcal{R}_\emptyset(G(s')))] > e^{2e^\varepsilon\beta} > 1 + e^\varepsilon\beta$$

or

$$\mathop{\mathbf{E}}_{s'\sim\{0,1\}^\ell}[\pi_x(\mathcal{R}_\emptyset(G(s')))] < e^{-2e^\varepsilon\beta} < 1 - e^\varepsilon\beta,$$

where we used the assumption that $\beta < 1/(2e^\varepsilon)$ in the second inequality.

We first deal with the case when $\mathbf{E}_{s'\sim\{0,1\}^\ell}[\pi_x(\mathcal{R}_\emptyset(G(s')))] > 1 + e^\varepsilon\beta$ (as the other case will be essentially identical). Observe that for true randomness we have:

$$\mathop{\mathbf{E}}_{r'\sim\{0,1\}^t}[\pi_x(\mathcal{R}_\emptyset(r'))] = \mathop{\mathbf{E}}_{r'\sim\{0,1\}^t}\left[\frac{\mathbf{Pr}[\mathcal{R}(x) = \mathcal{R}_\emptyset(r')]}{\mathbf{Pr}_{r\sim\{0,1\}^t}[\mathcal{R}_\emptyset(r) = \mathcal{R}_\emptyset(r')]}\right] = 1.$$

Using the fact that $\pi_x(y) \in [0, e^\varepsilon]$ we have that

$$\mathop{\mathbf{E}}_{s'\sim\{0,1\}^\ell}[\pi_x(\mathcal{R}_\emptyset(G(s')))] = \int_0^{e^\varepsilon} \mathop{\mathbf{Pr}}_{s'\sim\{0,1\}^\ell}[\pi_x(\mathcal{R}_\emptyset(G(s'))) \geq \theta] d\theta$$

and, similarly,

$$\mathop{\mathbf{E}}_{r'\sim\{0,1\}^t}[\pi_x(\mathcal{R}_\emptyset(r'))] = \int_0^{e^\varepsilon} \mathop{\mathbf{Pr}}_{r'\sim\{0,1\}^t}[\pi_x(\mathcal{R}_\emptyset(r')) \geq \theta]\, d\theta.$$

Thus, by our assumption,

$$\int_0^{e^\varepsilon} \left(\mathop{\mathbf{Pr}}_{s'\sim\{0,1\}^\ell}[\pi_x(\mathcal{R}_\emptyset(G(s'))) \geq \theta] - \mathop{\mathbf{Pr}}_{r'\sim\{0,1\}^t}[\pi_x(\mathcal{R}_\emptyset(r')) \geq \theta] \right) d\theta > e^\varepsilon\beta.$$

This implies that there exists $\theta \in [0, e^\varepsilon]$ such that

$$\mathop{\mathbf{Pr}}_{s' \sim \{0,1\}^\ell}[\pi_x(\mathcal{R}_\emptyset(G(s'))) \geq \theta] - \mathop{\mathbf{Pr}}_{r' \sim \{0,1\}^t}[\pi_x(\mathcal{R}_\emptyset(r')) \geq \theta] > \beta.$$

Note that $\mathtt{ind}(\pi_x(\mathcal{R}_\emptyset(r')) \geq \theta) \in \mathcal{D}$, for all $x \in X$ and $\theta \in [0, e^\varepsilon]$ contradicting our assumption on G. Thus we obtain that $\mathbf{E}_{s' \sim \{0,1\}^\ell}[\pi_x(\mathcal{R}_\emptyset(G(s')))] \leq 1 + e^\varepsilon \beta$. We can arrive at a contradiction in the case when $\mathbf{E}_{s' \sim \{0,1\}^\ell}[\pi_x(\mathcal{R}_\emptyset(G(s')))] < 1 - e^\varepsilon \beta$ in exactly the same way.

To show that $\mathcal{R}[G, \gamma]$ is a deletion $(\varepsilon + 2e^\varepsilon\beta)$-DP local randomizer we observe that for every x, conditioned on accepting one of the samples, $\mathcal{R}[G, \gamma](x)$ outputs a sample distributed exactly according to $\mathcal{R}[G](x)$. If $\mathcal{R}[G, \gamma](x)$ does not accept any samples than it samples from the reference distribution ρ_G. Thus given that $\mathcal{R}[G](x)$ is $(\varepsilon + 2e^\varepsilon\beta)$-close to ρ_G we have that the output distribution $\mathcal{R}[G, \gamma](x)$ is also $(\varepsilon + 2e^\varepsilon\beta)$-close to ρ_G. □

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.

As the first step, we show that when used with the identity G, the resulting randomizer is γ-close in total variation distance to $\mathcal{R}$.

Lemma 3.3. *Let $\mathcal{R}$ be a deletion ε-DP t-samplable local randomizer. Then for the identity function $\mathrm{ID}_t \colon \{0,1\}^t \to \{0,1\}^t$ and any $\gamma > 0$ we have that $\mathcal{R}[\mathrm{ID}_t, \gamma]$ is a deletion ε-DP local randomizer and for every $x \in \mathcal{X}$, $\mathrm{TV}(\mathcal{R}_\emptyset(\mathcal{R}[\mathrm{ID}_t, \gamma](x)), \mathcal{R}(x)) \leq \gamma$.*

Proof. When applied with $G = \mathrm{ID}_t$, y is distributed according to the reference distribution of $\mathcal{R}$. Thus the algorithm performs standard rejection sampling until it accepts a sample or exceeds the bound J on the number of steps. Note that deletion DP implies that $\frac{\mathbf{Pr}[\mathcal{R}(x)=y]}{e^\varepsilon \, \mathbf{Pr}_{r \sim \{0,1\}^t}[\mathcal{R}_\emptyset(r)=y]} \leq 1$. At each step, conditioned on success the algorithm samples s such that $\mathcal{R}_\emptyset(s)$ is distributed identically to $\mathcal{R}(x)$. Further, the acceptance probability at each step is

$$\mathop{\mathbf{E}}_{y \sim \rho}\left[\frac{\mathbf{Pr}[\mathcal{R}(x) = y]}{e^\varepsilon \, \mathbf{Pr}_{r \sim \{0,1\}^t}[\mathcal{R}_\emptyset(r) = y]}\right] = \sum_{y \in Y} \frac{\mathbf{Pr}[\mathcal{R}(x) = y]}{e^\varepsilon} = \frac{1}{e^\varepsilon}.$$

Thus the probability that all steps reject is $\leq (1 - e^{-\varepsilon})^J \leq \gamma$. It follows that $\mathrm{TV}(\mathcal{R}_\emptyset(\mathcal{R}[\mathrm{ID}_t, \gamma](x)), \mathcal{R}(x))$ is bounded by γ as claimed. □

We can now state the implications of using a sufficiently strong PRG on the output of the randomizer.

Lemma 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. 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 $T' = T - T(\mathcal{R}, G, \gamma)$.*

Proof. By Lemma 3.3, $\mathrm{TV}(\mathcal{R}_\emptyset(\mathcal{R}[\mathrm{ID}_t, \gamma](x)), \mathcal{R}(x)) \leq \gamma$ and thus it suffices to prove that $\mathcal{R}_\emptyset(G(\mathcal{R}[G, \gamma](x)))$ is $(T', e^\varepsilon \ln(1/\gamma)\beta)$-computationally indistinguishable from $\mathcal{R}_\emptyset(\mathcal{R}[\mathrm{ID}_t, \gamma](x))$. Towards a contradiction, suppose that there exists a test D' running in time T' such that for some x,

$$\begin{aligned}\big|\, &\mathbf{Pr}[D'(\mathcal{R}_\emptyset(G(\mathcal{R}[G, \gamma](x)))) = 1] - \\ &\quad \mathbf{Pr}[D'(\mathcal{R}_\emptyset(\mathcal{R}[\mathrm{ID}_t, \gamma](x))) = 1]\big| \geq e^\varepsilon \ln(1/\gamma)\beta.\end{aligned}$$

We claim that there exists a test for distinguishing $G(s)$ for $s \sim \{0,1\}^\ell$ from a truly random seed $r \sim \{0,1\}^t$. Note that $\mathcal{R}_\emptyset(G(\mathcal{R}[G, \gamma]))$ can be seen as $\mathcal{R}[G, \gamma]$ that outputs directly $y = \mathcal{R}_\emptyset(G(s))$ instead of s itself. Next we observe that $\mathcal{R}_\emptyset(G(\mathcal{R}[G, \gamma]))$ uses the output of G at most $J = e^\varepsilon \ln(1/\gamma)$ times in place of truly random

t-bit string used by $\mathcal{R}_\emptyset(\mathcal{R}[\mathrm{ID}_t,\gamma])$. Thus, by the standard hybrid argument, one of those applications can be used to test G with success probability at least $e^\varepsilon \ln(1/\gamma)\beta/J = \beta$. This test requires running a hybrid between $\mathcal{R}_\emptyset(G(\mathcal{R}[G,\gamma]))$ and $\mathcal{R}_\emptyset(\mathcal{R}[\mathrm{ID}_t,\gamma])$ in addition to D' itself. The resulting test runs in time $T' + T(\mathcal{R},G,\gamma) = T$.Thus we obtain a contradiction to G being a (T,β)-PRG. □

As a direct corollary of Lemmas 3.2 and 3.4 we obtain a general way to compress efficient LDP randomizers.

Theorem 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 (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 $T' = T - T(\mathcal{R},G,\gamma)$.*

Proof. The second part of the claim is exactly Lemma 3.4. To see the first part of the claim note that by our assumption $T > T(\mathcal{R},G,\gamma)$ and computation of the ratio of densities $\frac{\mathbf{Pr}[\mathcal{R}(x)=\mathcal{R}_\emptyset(r')]}{e^\varepsilon\, \mathbf{Pr}_{r\sim\{0,1\}^t}[\mathcal{R}_\emptyset(r)=\mathcal{R}_\emptyset(r')]}$ for any $r' \in \{0,1\}^t$ is part of $\mathcal{R}[G,\gamma]$. This implies that the test family $\mathcal{D}$ defined in Lemma 3.2 can be computed in time T. Now applying Lemma 3.2 gives us the privacy claim. □

By plugging an exponentially strong PRG G into Theorem 3.5 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 3.6. *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.*

Proof. As in the proof of Lemma 3.2 we observe that if we take the reference distribution to be uniform over $\{0,1\}^\ell$ we will get that, conditioned on accepting a sample, the seed s is output with probability $\mu_x(s)$ such that

$$\frac{\mu_x(s)}{\rho_G(s)} = \frac{\mu_x(s)}{2^{-\ell}} = \frac{\pi_x(\mathcal{R}_\emptyset(G(s)))}{\mathbf{E}_{s'\sim\{0,1\}^\ell}[\pi_x(\mathcal{R}_\emptyset(G(s')))]}.$$

By the fact that $\mathcal{R}(x)$ is ε-DP we have that for every $s' \in \{0,1\}^\ell$, $\pi_x(\mathcal{R}_\emptyset(G(s'))) \in [e^{-\varepsilon}, e^\varepsilon]$ and thus $\frac{\mu_x(s)}{\rho_G(s)} \in [e^{-2\varepsilon}, e^{2\varepsilon}]$. □

3.1 Replacement DP

We now show that the same approach can be used to compress a replacement ε_r-DP randomizer $\mathcal{R}$. To do this we first let ρ be some reference distribution relative to which $\mathcal{R}$ is deletion ε-DP for some $\varepsilon \le \varepsilon_r$. One possible way to define ρ is to pick some fixed $x_0 \in X$ and let ρ be the distribution of $\mathcal{R}(x_0)$. In this case $\varepsilon = \varepsilon_r$. But other choices of ρ are possible that give an easy to sample distribution and $\varepsilon < \varepsilon_r$. In fact, for some standard randomizers such as addition of Laplace noise we will get $\varepsilon = \varepsilon_r/2$.

Now assuming that ρ is t-samplable and given a PRG $G\colon \{0,1\}^\ell \to \{0,1\}^t$ we define $\mathcal{R}[G]$ as in Def. 3.1 and $\mathcal{R}[G,\gamma]$ us in Algorithm 1. The randomizer $\mathcal{R}$ is deletion ε-DP so all the results we proved apply to it as well (with the deletion ε and not the replacement ε_r). In addition we show that replacement privacy is preserved as well.

Lemma 3.7. *For a t-samplable deletion ε-DP and replacement ε_r-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:*

$$\mathcal{D} := \left\{ \mathtt{ind}\left(\frac{\mathbf{Pr}[\mathcal{R}(x) = \mathcal{R}_\emptyset(r')]}{\mathbf{Pr}_{r\sim\{0,1\}^t}[\mathcal{R}_\emptyset(r) = \mathcal{R}_\emptyset(r')]} \ge \theta \right) \;\middle|\; x \in X, \theta \in [0, e^\varepsilon] \right\}.$$

If G β-fools $\mathcal{D}$ for $\beta < 1/(2e^\varepsilon)$ then $R[G]$ is a replacement $(\varepsilon_r + 4e^\varepsilon\beta)$-DP local randomizer. Furthermore, for every $\gamma > 0$, $\mathcal{R}[G,\gamma]$ is a replacement $(\varepsilon_r + 4e^\varepsilon\beta)$-DP local randomizer.

Proof. As in the proof of Lemma 3.2, for $y \in Y$, we denote the density ratio of $\mathcal{R}(x)$ to ρ at y by

$$\pi_x(y) := \frac{\mathbf{Pr}[\mathcal{R}(x) = y]}{\mathbf{Pr}_{r\sim\{0,1\}^t}[\mathcal{R}_\emptyset(r) = y]}$$

and note that $\mathcal{R}[G]$ outputs a seed s with probability:

$$\mu_x(s) := \frac{\pi(\mathcal{R}_\emptyset(G(s)))}{\sum_{s'\in\{0,1\}^\ell} \pi(\mathcal{R}_\emptyset(G(s')))}.$$

Thus for two inputs $x, x' \in X$ and any $s \in \{0,1\}^\ell$ we have that

$$\begin{aligned}\frac{\mu_x(s)}{\mu_{x'}(s)} &= \frac{\pi_x(\mathcal{R}_\emptyset(G(s)))}{\pi_{x'}(\mathcal{R}_\emptyset(G(s)))} \cdot \frac{\sum_{s'\in\{0,1\}^\ell} \pi_{x'}(\mathcal{R}_\emptyset(G(s')))}{\sum_{s'\in\{0,1\}^\ell} \pi_x(\mathcal{R}_\emptyset(G(s')))} \\ &= \frac{\mathbf{Pr}[\mathcal{R}(x) = \mathcal{R}_\emptyset(G(s'))]}{\mathbf{Pr}[\mathcal{R}(x') = \mathcal{R}_\emptyset(G(s'))]} \cdot \frac{\mathbf{E}_{s'\sim\{0,1\}^\ell}[\pi_{x'}(\mathcal{R}_\emptyset(G(s')))]}{\mathbf{E}_{s'\sim\{0,1\}^\ell}[\pi_x(\mathcal{R}_\emptyset(G(s')))]}\end{aligned}$$

Now $\mathcal{R}$ is ε_r-replacement-DP and therefore the first term satisfies:

$$\frac{\mathbf{Pr}[\mathcal{R}(x) = \mathcal{R}_\emptyset(G(s'))]}{\mathbf{Pr}[\mathcal{R}(x') = \mathcal{R}_\emptyset(G(s'))]} \in \left[e^{-\varepsilon_r}, e^{\varepsilon_r}\right].$$

At the same time, we showed in Lemma 3.2 that $\mathbf{E}_{s'\sim\{0,1\}^\ell}[\pi_x(\mathcal{R}_\emptyset(G(s')))] \in [e^{-2e^\varepsilon\beta}, e^{2e^\varepsilon\beta}]$ and also $\mathbf{E}_{s'\sim\{0,1\}^\ell}[\pi_{x'}(\mathcal{R}_\emptyset(G(s')))] \in [e^{-2e^\varepsilon\beta}, e^{2e^\varepsilon\beta}]$. Therefore $\frac{\mu_x(s)}{\mu_{x'}(s)} \in [e^{-\varepsilon_r-4e^\varepsilon\beta}, e^{\varepsilon_r+4e^\varepsilon\beta}]$.

To show that $\mathcal{R}[G,\gamma]$ is a replacement $(\varepsilon_r + 4e^\varepsilon\beta)$-DP local randomizer we observe that for every x, $\mathcal{R}[G,\gamma](x)$ is a mixture of $\mathcal{R}[G](x)$ and ρ_G. As we showed, $\mathcal{R}[G](x)$ is $(\varepsilon_r + 4e^\varepsilon\beta)$-close to $\mathcal{R}[G](x')$ and we also know from Lemma 3.2 that ρ_G is $(\varepsilon + 2e^\varepsilon\beta)$-close to $\mathcal{R}[G](x')$. By quasi-convexity we obtain that $\mathcal{R}[G,\gamma](x)$ is $(\varepsilon_r+4e^\varepsilon\beta)$-close to $\mathcal{R}[G](x')$. We also know that $\mathcal{R}[G,\gamma](x)$ is $(\varepsilon+2e^\varepsilon\beta)$-close to ρ_G. Appealing to quasi-convexity again, we obtain that $\mathcal{R}[G,\gamma](x)$ is $(\varepsilon_r + 4e^\varepsilon\beta)$-close to $\mathcal{R}[G,\gamma](x')$. □

3.2 Extension to (ε,δ)-DP

We next extend our approach to (ε,δ)-DP randomizers. The approach here is similar, except that we for some outputs $y = \mathcal{R}_\emptyset(G(s))$, the prescribed "rejection probability" in the original approach would be larger than one. To handle this, we simply truncate this ratio at 1 to get a probability. Algorithm 2 is identical to Algorithm 1 except for this truncation in the step where we sample b.

Algorithm 2 $\mathcal{R}[G,\gamma]$: PRG compression of deletion (ε,δ)-DP $\mathcal{R}$

```
Input: $x \in X$, $\varepsilon, \gamma > 0$; seeded PRG $G\colon \{0,1\}^\ell \to \{0,1\}^t$; $t$-samplable $\varepsilon$-DP randomizer $\mathcal{R}$.
$J = e^\varepsilon \ln(1/\gamma)/(1-\delta)$
for $j = 1, \ldots, J$ do
    Sample a random seed $s \in \{0,1\}^\ell$.
    $y = \mathcal{R}_\emptyset(G(s))$
    Sample $b$ from Bernoulli $\left(\min\left(1, \frac{\mathbf{Pr}[\mathcal{R}(x)=y]}{e^\varepsilon\,\mathbf{Pr}_{r\sim\{0,1\}^t}[\mathcal{R}_\emptyset(r)=y]}\right)\right)$
    if $b == 1$ then
        BREAK
Send $s$
```

The proof is fairly similar to that for the pure DP randomizer. We start with a lemma that relates the properties of the PRG to the properties of the randomizer that need to be preserved in order to ensure that it satisfies deletion (ε',δ')-LDP.

Lemma 3.8. *For a t-samplable deletion (ε, δ)-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:*

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

Suppose that G β-fools $\mathcal{D}$ and let $\pi_x(y) := \frac{\min(\mathbf{Pr}[\mathcal{R}(x)=y], e^\varepsilon \mathbf{Pr}[\mathcal{R}(\emptyset)=y])}{\mathbf{Pr}_{r\sim\{0,1\}^t}[\mathcal{R}_\emptyset(r)=y]}$. Then

$$\mathop{\mathbf{E}}_{s'\sim\{0,1\}^\ell}[\pi_x(\mathcal{R}_\emptyset(G(s')))] \in [1-\delta-e^\varepsilon\beta, 1+e^\varepsilon\beta]$$

and

$$\mathop{\mathbf{E}}_{s'\sim\{0,1\}^\ell}\left[|1-e^\varepsilon\pi_x(\mathcal{R}_\emptyset(G(s')))|_+\right] \leq \delta + \beta.$$

Proof. Let ρ_G be the uniform distribution over $\{0,1\}^\ell$. Let $\nu_x(y) := \mathbf{Pr}[\mathcal{R}(x) = y]$ and let $\tilde{\nu}_x(y) := \min(\nu_x(y), e^\varepsilon \mathbf{Pr}[\mathcal{R}(\emptyset) = y])$. Note that $\tilde{\nu}_x(\cdot)$ does not necessarily define a probability distribution. For $S = \{y : \tilde{\nu}_x(y) < \nu_x(y)\}$, we have

$$\begin{aligned}
\nu_x(S) &= \sum_{y\in S} \nu_x(y) \\
&= \sum_{y\in S} \tilde{\nu}_x(y) + \sum_{y\in S}(\nu_x(y) - \tilde{\nu}_x(y)) \\
&= \sum_{y\in S} e^\varepsilon\rho(y) + \sum_{y}(\nu_x(y) - \tilde{\nu}_x(y)) \\
&= e^\varepsilon\rho(S) + (1 - \sum_{y}\tilde{\nu}_x(y)).
\end{aligned}$$

Then deletion (ε, δ)-DP of $\mathcal{R}$ implies that $\sum_y \tilde{\nu}_x(y) \geq 1 - \delta$. Observe that this implies that for true randomness we have:

$$\begin{aligned}
\mathop{\mathbf{E}}_{r'\sim\{0,1\}^t}[\pi_x(\mathcal{R}_\emptyset(r'))] &= \mathop{\mathbf{E}}_{r'\sim\{0,1\}^t}\left[\frac{\tilde{\nu}_x(\mathcal{R}_\emptyset(r'))}{\mathbf{Pr}_{r\sim\{0,1\}^t}[\mathcal{R}_\emptyset(r) = \mathcal{R}_\emptyset(r')]}\right] \\
&= \mathop{\mathbf{E}}_{r'\sim\{0,1\}^t}\left[\frac{\tilde{\nu}_x(\mathcal{R}_\emptyset(r'))}{\rho(\mathcal{R}_\emptyset(r'))}\right] \\
&= \mathop{\mathbf{E}}_{y\sim\rho}\left[\frac{\tilde{\nu}_x(y)}{\rho(y)}\right] \\
&= \sum_{y\in Y}\rho(y)\cdot\frac{\tilde{\nu}_x(y)}{\rho(y)} \\
&= \sum_{y\in Y}\tilde{\nu}_x(y) \quad \in \quad [1-\delta, 1].
\end{aligned}$$

Using the fact that $\pi_x(y) \in [0, e^\varepsilon]$ we have that

$$\mathop{\mathbf{E}}_{s'\sim\{0,1\}^\ell}[\pi_x(\mathcal{R}_\emptyset(G(s')))] = \int_0^{e^\varepsilon} \mathop{\mathbf{Pr}}_{s'\sim\{0,1\}^\ell}[\pi_x(\mathcal{R}_\emptyset(G(s'))) \geq \theta]d\theta$$

and, similarly,

$$\mathop{\mathbf{E}}_{r'\sim\{0,1\}^t}[\pi_x(\mathcal{R}_\emptyset(r'))] = \int_0^{e^\varepsilon} \mathop{\mathbf{Pr}}_{r'\sim\{0,1\}^t}[\pi_x(\mathcal{R}_\emptyset(r')) \geq \theta]\,d\theta.$$

Thus, it follows that

$$\begin{aligned}
&\left| \mathop{\mathbf{E}}_{s' \sim \{0,1\}^\ell} [\pi_x(\mathcal{R}_\emptyset(G(s')))] - \mathop{\mathbf{E}}_{r' \sim \{0,1\}^t} [\pi_x(\mathcal{R}_\emptyset(r'))] \right| \\
&\qquad = \left| \int_0^{e^\varepsilon} \left(\mathop{\mathbf{Pr}}_{s' \sim \{0,1\}^\ell} [\pi_x(\mathcal{R}_\emptyset(G(s'))) \geq \theta] - \mathop{\mathbf{Pr}}_{r' \sim \{0,1\}^t} [\pi_x(\mathcal{R}_\emptyset(r')) \geq \theta] \right) d\theta \right| \\
&\qquad \leq \int_0^{e^\varepsilon} \left| \mathop{\mathbf{Pr}}_{s' \sim \{0,1\}^\ell} [\pi_x(\mathcal{R}_\emptyset(G(s'))) \geq \theta] - \mathop{\mathbf{Pr}}_{r' \sim \{0,1\}^t} [\pi_x(\mathcal{R}_\emptyset(r')) \geq \theta] \right| d\theta \\
&\qquad \leq e^\varepsilon \beta,
\end{aligned}$$

where in the last step, we have used the property of the pseudorandom generator that it fools $\mathcal{D}$, and the fact that for $\theta \in [0, e^\varepsilon)$, $\frac{\tilde{\nu}_x(y)}{\mathbf{Pr}[\mathcal{R}(\emptyset)=y]} < \theta$ if and only if $\frac{\nu_x(y)}{\mathbf{Pr}[\mathcal{R}(\emptyset)=y]} < \theta$. The first part of the claim follows.

For the second part of the claim we first note that deletion (ε, δ)-DP of $\mathcal{R}$ implies that

$$\begin{aligned}
\mathop{\mathbf{E}}_{r' \sim \{0,1\}^t} [|1 - e^\varepsilon \pi_x(\mathcal{R}_\emptyset(r'))|_+] &= \mathop{\mathbf{E}}_{y \sim \rho} [|1 - e^\varepsilon \pi_x(y)|_+] \\
&= \mathop{\mathbf{E}}_{y \sim \rho} [|1 - e^\varepsilon \pi_x(y)|_+] \\
&= \sum_{y \in Y} \rho(y) |1 - e^\varepsilon \pi_x(y)|_+ \\
&= \sum_{y \in Y} |\rho(y) - e^\varepsilon \tilde{\nu}_x(y)|_+ \\
&= \sum_{y \in Y} |\rho(y) - e^\varepsilon \nu_x(y)|_+ \leq \delta.
\end{aligned}$$

Also note that

$$\mathop{\mathbf{E}}_{r' \sim \{0,1\}^t} [|1 - e^\varepsilon \pi_x(\mathcal{R}_\emptyset(r'))|_+] = \int_0^1 \mathop{\mathbf{Pr}}_{r' \sim \{0,1\}^t} [1 - e^\varepsilon \pi_x(\mathcal{R}_\emptyset(r')) \geq \theta] \, d\theta = 1 - \int_0^1 \mathop{\mathbf{Pr}}_{r' \sim \{0,1\}^t} \left[\pi_x(\mathcal{R}_\emptyset(r')) \geq \frac{\theta}{e^\varepsilon} \right] d\theta.$$

Similarly,

$$\mathop{\mathbf{E}}_{s' \sim \{0,1\}^\ell} [|1 - e^\varepsilon \pi_x(\mathcal{R}_\emptyset(G(s')))|_+] = 1 - \int_0^1 \mathop{\mathbf{Pr}}_{s' \sim \{0,1\}^\ell} \left[\pi_x(\mathcal{R}_\emptyset(G(s'))) \geq \frac{\theta}{e^\varepsilon} \right] d\theta.$$

Thus by the same argument as before, the fact that G, β-fools $\mathcal{D}$ implies that

$$\mathop{\mathbf{E}}_{s' \sim \{0,1\}^\ell} \left[|1 - e^\varepsilon \pi_x(\mathcal{R}_\emptyset(G(s')))|_+ \right] \leq \mathop{\mathbf{E}}_{r' \sim \{0,1\}^t} [|1 - e^\varepsilon \pi_x(\mathcal{R}_\emptyset(r'))|_+] + \beta \leq \delta + \beta.$$

□

We can now give an analogue of Lemma 3.2 for deletion (ε, δ)-DP randomizers.

Lemma 3.9. *For a t-samplable deletion (ε, δ)-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:*

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

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

Proof. As before, we let the reference distribution ρ_G be the uniform distribution over $\{0,1\}^\ell$. Using the definitions in the proof of Lemma 3.8 we observe that $\mathcal{R}[G](x)$ outputs s with probability:

$$\mu_x(s) := \frac{\pi(\mathcal{R}_\emptyset(G(s)))}{\sum_{s'\in\{0,1\}^\ell} \pi(\mathcal{R}_\emptyset(G(s')))} = \frac{\frac{\tilde{\nu}_x(\mathcal{R}_\emptyset(G(s)))}{\rho(\mathcal{R}_\emptyset(G(s)))}}{2^\ell \cdot \mathbf{E}_{s'\sim\{0,1\}^\ell}[\pi_x(\mathcal{R}_\emptyset(G(s')))]} = \rho_G(s) \cdot \frac{\frac{\tilde{\nu}_x(\mathcal{R}_\emptyset(G(s)))}{\rho(\mathcal{R}_\emptyset(G(s)))}}{\mathbf{E}_{s'\sim\{0,1\}^\ell}[\pi_x(\mathcal{R}_\emptyset(G(s')))]}.$$

By the definition of $\tilde{\nu}_x$ we have that the numerator satisfies $\frac{\tilde{\nu}_x(\mathcal{R}_\emptyset(G(s)))}{\rho(\mathcal{R}_\emptyset(G(s)))} \leq e^\varepsilon$. In addition, by Lemma 3.8 the denominator $\mathbf{E}_{s'\sim\{0,1\}^\ell}[\pi_x(\mathcal{R}_\emptyset(G(s')))] \geq 1-\delta-e^\varepsilon\beta$. Therefore

$$\mu_x(s) \leq \rho_G(s) \cdot \frac{e^\varepsilon}{1-\delta-e^\varepsilon\beta} \leq e^{\varepsilon+2\delta+e^\varepsilon\beta}\rho_G(s).$$

For the other side of (ε,δ)-closeness we simply observe that by the Lemma 3.8,

$$\begin{aligned}
\sum_{s\in\{0,1\}^\ell} \left|\rho_G(s) - e^{\varepsilon+e^\varepsilon\beta}\mu_x(s)\right|_+ &= \sum_{s\in\{0,1\}^\ell} \left|\rho_G(s) - e^{\varepsilon+e^\varepsilon\beta}\rho_G(s)\cdot\frac{\pi_x(\mathcal{R}_\emptyset(G(s)))}{\mathbf{E}_{s'\sim\{0,1\}^\ell}[\pi_x(\mathcal{R}_\emptyset(G(s')))]}\right|_+ \\
&\leq \sum_{s\in\{0,1\}^\ell} |\rho_G(s) - e^\varepsilon\rho_G(s)\cdot\pi_x(\mathcal{R}_\emptyset(G(s)))|_+ \\
&= \mathop{\mathbf{E}}_{s\sim\{0,1\}^\ell}\left[|1-e^\varepsilon\pi_x(\mathcal{R}_\emptyset(G(s)))|_+\right] \leq \delta+\beta.
\end{aligned}$$

Finally to establish that $R[G,\gamma]$ is $(\varepsilon+2\delta+2e^\varepsilon\beta, \delta+\beta)$ we, as before, appeal to quasi-convexity. □

To establish the utility guarantees for $\mathcal{R}[G,\gamma]$ we follow the same approach by establishing the utility guarantees for $\mathcal{R}[\mathrm{ID}_t,\gamma]$ and then using the properties of G.

Lemma 3.10. *Let $\mathcal{R}$ be a deletion ε-DP t-samplable local randomizer. Then for the identity function $\mathrm{ID}_t\colon \{0,1\}^t \to \{0,1\}^t$ and any $\gamma > 0$ we have that $\mathcal{R}[\mathrm{ID}_t,\gamma]$ is a deletion ε-DP local randomizer and for every $x \in \mathcal{X}$, $\mathrm{TV}(\mathcal{R}_\emptyset(\mathcal{R}[\mathrm{ID}_t,\gamma](x)), \mathcal{R}(x)) \leq \delta+\gamma$.*

Proof. Conditioned on accepting a sample, $\mathcal{R}[\mathrm{ID}_t,\gamma]$ outputs a sample from the truncated version of the distribution of $\mathcal{R}(x)$. Specifically, y is output with probability $\bar{\nu}_x(y) := \frac{\tilde{\nu}(y)}{\sum_{y\in Y}\tilde{\nu}(y)}$, where $\nu_x(y) := \mathbf{Pr}[\mathcal{R}(x) = y]$ and $\tilde{\nu}_x(y) := \min(\nu_x(y), e^\varepsilon\,\mathbf{Pr}[\mathcal{R}(\emptyset) = y])$. From the proof of Lemma 3.8, we know that $\sum_{y\in Y}\tilde{\nu}_x(y) \geq 1-\delta$. Thus

$$\begin{aligned}
\mathrm{TV}(\nu_x,\bar{\nu}_x) &= \frac{1}{2}\sum_{y\in Y}|\nu_x(y)-\bar{\nu}_x(y)| \\
&\leq \frac{1}{2}\sum_{y\in Y}(|\nu_x(y)-\tilde{\nu}_x(y)| + |\tilde{\nu}_x(y)-\bar{\nu}_x(y)|) \\
&= \frac{1}{2}\sum_{y\in Y}(\nu_x(y)-\tilde{\nu}_x(y)+\bar{\nu}_x(y)-\tilde{\nu}_x(y)) \leq \delta.
\end{aligned}$$

Truncation of the distribution also reduces the probability that a sample is accepted. Specifically,

$$\mathop{\mathbf{E}}_{y\sim\rho}\left[\frac{\tilde{\nu}_x(y)}{e^\varepsilon\rho(y)}\right] = \sum_{y\in Y}\frac{\tilde{\nu}_x(y)}{e^\varepsilon} \geq \frac{1-\delta}{e^\varepsilon}.$$

$\mathcal{R}[G,\gamma]$ tries at least $e^\varepsilon\ln(1/\gamma)/(1-\delta)$ samples and therefore, as in the proof of Lemma 3.3, failure to accept any samples adds at most γ to the total variation distance. □

From here we can directly obtain the analogues of Lemma 3.4 and Theorem3.5.

Finally, to deal with the replacement version of (ε, δ)-DP we combine the ideas we used in Lemmas 3.7 and 3.9. The main distinction is a somewhat stronger test that we need to fool in this case.

Lemma 3.11. *For a t-samplable replacement $(\varepsilon_r, \delta_r)$-DP and deletion (ε, δ)-DP local randomizer $\mathcal{R}\colon X \to Y$ and $G\colon \{0,1\}^\ell \to \{0,1\}^t$, let $\mathcal{D}$ and $\mathcal{D}_r$ denote the following families of tests which take $r' \in \{0,1\}^t$ as an input:*

$$\mathcal{D} := \left\{ \mathtt{ind}\left(\frac{\mathbf{Pr}[\mathcal{R}(x) = \mathcal{R}_\emptyset(r')]}{\rho(\mathcal{R}_\emptyset(r'))} \geq \theta \right) \;\middle|\; x \in X, \theta \in [0, e^\varepsilon] \right\};$$

$$\mathcal{D}_r := \left\{ \mathtt{ind}\left(\frac{\tilde{\nu}_x(\mathcal{R}_\emptyset(r'))}{\rho(\mathcal{R}_\emptyset(r'))} - e^\varepsilon \frac{\tilde{\nu}_{x'}(\mathcal{R}_\emptyset(r'))}{\rho(\mathcal{R}_\emptyset(r'))} \geq \theta \right) \;\middle|\; x, x' \in X, \theta \in [0, e^\varepsilon] \right\},$$

where ρ is the reference distribution of $\mathcal{R}$ and $\tilde{\nu}_x(y) := \min(\mathbf{Pr}[\mathcal{R}(x) = y], e^\varepsilon \rho(y))$. If G β-fools $\mathcal{D} \cup \mathcal{D}_r$ where $\delta + e^\varepsilon\beta < 1/2$ then $R[G]$ is a replacement $(\varepsilon_r + 2\delta + 3e^\varepsilon\beta, 2\delta_r + 2e^\varepsilon\beta)$-DP local randomizer. Furthermore, for every $\gamma > 0$, $\mathcal{R}[G, \gamma]$ is a $(\varepsilon_r + 2\delta + 3e^\varepsilon\beta, 2\delta_r + 2e^\varepsilon\beta)$-DP local randomizer.

Proof. First we observe that $\mathcal{R}$ being $(\varepsilon_r, \delta_r)$ replacement DP implies that $\tilde{\nu}_x$ and $\tilde{\nu}_{x'}$ are $(\varepsilon_r, \delta_r)$ close in the following sense:

$$\begin{aligned}
\mathop{\mathbf{E}}_{r' \sim \{0,1\}^t}\left[\left| \frac{\tilde{\nu}_x(\mathcal{R}_\emptyset(r'))}{\rho(\mathcal{R}_\emptyset(r'))} - e^{\varepsilon_r} \frac{\tilde{\nu}_{x'}(\mathcal{R}_\emptyset(r'))}{\rho(\mathcal{R}_\emptyset(r'))} \right|_+ \right] &= \mathop{\mathbf{E}}_{y \sim \rho}\left[\left| \frac{\tilde{\nu}_x(y)}{\rho(y)} - e^{\varepsilon_r} \frac{\tilde{\nu}_{x'}(y)}{\rho(y)} \right|_+ \right] \\
&= \sum_{y \in Y} |\tilde{\nu}_x(y) - e^{\varepsilon_r} \tilde{\nu}_{x'}(y)|_+ \\
&\leq \sum_{y \in Y} |\nu_x(y) - e^{\varepsilon_r} \nu_{x'}(y)|_+ \leq \delta_r,
\end{aligned}$$

where we used the fact that if $\nu_{x'}(y) > \tilde{\nu}_{x'}(y)$ then $\tilde{\nu}_{x'}(y) = e^\varepsilon \rho(y) \geq \tilde{\nu}_x(y)$ and so

$$|\tilde{\nu}_x(y) - e^{\varepsilon_r} \tilde{\nu}_{x'}(y)|_+ = |\tilde{\nu}_x(y) - e^{\varepsilon_r} \nu_{x'}(y)|_+ .$$

Using the decomposition

$$\mathop{\mathbf{E}}_{r' \sim \{0,1\}^t}\left[\left| \frac{\tilde{\nu}_x(\mathcal{R}_\emptyset(r'))}{\rho(\mathcal{R}_\emptyset(r'))} - e^{\varepsilon_r} \frac{\tilde{\nu}_{x'}(\mathcal{R}_\emptyset(r'))}{\rho(\mathcal{R}_\emptyset(r'))} \right|_+ \right] = \int_0^{e^\varepsilon} \mathop{\mathbf{Pr}}_{r' \sim \{0,1\}^t}\left[\frac{\tilde{\nu}_x(\mathcal{R}_\emptyset(r'))}{\rho(\mathcal{R}_\emptyset(r'))} - e^{\varepsilon_r} \frac{\tilde{\nu}_{x'}(\mathcal{R}_\emptyset(r'))}{\rho(\mathcal{R}_\emptyset(r'))} \geq \theta \right] d\theta$$

and the fact that G β fools $\mathcal{D}_r$ we obtain that

$$\mathop{\mathbf{E}}_{s' \sim \{0,1\}^\ell}\left[\left| \frac{\tilde{\nu}_x(\mathcal{R}_\emptyset(G(s')))}{\rho(\mathcal{R}_\emptyset(G(s')))} - e^{\varepsilon_r} \frac{\tilde{\nu}_{x'}(\mathcal{R}_\emptyset(G(s')))}{\rho(\mathcal{R}_\emptyset(G(s')))} \right|_+ \right] \leq \delta_r + e^\varepsilon\beta. \tag{1}$$

By Lemma 3.8 we have that for $\pi_x(y) := \frac{\tilde{\nu}_x(y)}{\rho(y)}$ it holds that

$$\zeta_x := \mathop{\mathbf{E}}_{s' \sim \{0,1\}^\ell}[\pi_x(\mathcal{R}_\emptyset(G(s')))] \in [1 - \delta - e^\varepsilon\beta, 1 + e^\varepsilon\beta].$$

Following the notation in Lemma 3.9, we know that the distribution of $\mathcal{R}[G](x)$ is

$$\mu_x(s) = \rho_G(s) \cdot \frac{\frac{\tilde{\nu}_x(\mathcal{R}_\emptyset(G(s)))}{\rho(\mathcal{R}_\emptyset(G(s)))}}{\mathbf{E}_{s' \sim \{0,1\}^\ell}[\pi_x(\mathcal{R}_\emptyset(G(s')))]} = \frac{\rho_G(s) \cdot \tilde{\nu}_x(\mathcal{R}_\emptyset(G(s)))}{\zeta_x \cdot \rho(\mathcal{R}_\emptyset(G(s)))}.$$

Thus setting $\varepsilon' = \varepsilon_r + 2\delta + 3e^{\varepsilon}\beta$ we obtain:

$$\begin{aligned}
\sum_{s'\in\{0,1\}^\ell} |\mu_x(s') - e^{\varepsilon'}\mu_{x'}(s')|_+ &= \mathop{\mathbf{E}}_{s'\sim\{0,1\}^\ell}\left[\left|\frac{\mu_x(s')}{\rho_G(s')} - e^{\varepsilon'}\frac{\mu_{x'}(s')}{\rho_G(s')}\right|_+\right] \\
&= \mathop{\mathbf{E}}_{s'\sim\{0,1\}^\ell}\left[\left|\frac{\tilde{\nu}_x(\mathcal{R}_\emptyset(G(s')))}{\zeta_x \cdot \rho(\mathcal{R}_\emptyset(G(s')))} - e^{\varepsilon'}\frac{\tilde{\nu}_{x'}(\mathcal{R}_\emptyset(G(s')))}{\zeta_{x'} \cdot \rho(\mathcal{R}_\emptyset(G(s')))}\right|_+\right] \\
&= \frac{1}{\zeta_x} \cdot \mathop{\mathbf{E}}_{s'\sim\{0,1\}^\ell}\left[\left|\frac{\tilde{\nu}_x(\mathcal{R}_\emptyset(G(s')))}{\rho(\mathcal{R}_\emptyset(G(s')))} - e^{\varepsilon'}\frac{\zeta_x \cdot \tilde{\nu}_{x'}(\mathcal{R}_\emptyset(G(s')))}{\zeta_{x'} \cdot \rho(\mathcal{R}_\emptyset(G(s')))}\right|_+\right] \\
&\leq \frac{1}{1-\delta-e^{\varepsilon}\beta} \cdot \mathop{\mathbf{E}}_{s'\sim\{0,1\}^\ell}\left[\left|\frac{\tilde{\nu}_x(\mathcal{R}_\emptyset(G(s')))}{\rho(\mathcal{R}_\emptyset(G(s')))} - e^{\varepsilon'}\frac{(1-\delta-e^{\varepsilon}\beta)\tilde{\nu}_{x'}(\mathcal{R}_\emptyset(G(s')))}{(1+e^{\varepsilon}\beta)\rho(\mathcal{R}_\emptyset(G(s')))}\right|_+\right] \\
&\leq \frac{1}{1-\delta-e^{\varepsilon}\beta} \cdot \mathop{\mathbf{E}}_{s'\sim\{0,1\}^\ell}\left[\left|\frac{\tilde{\nu}_x(\mathcal{R}_\emptyset(G(s')))}{\rho(\mathcal{R}_\emptyset(G(s')))} - e^{\varepsilon_r}\frac{\tilde{\nu}_{x'}(\mathcal{R}_\emptyset(G(s')))}{\rho(\mathcal{R}_\emptyset(G(s')))}\right|_+\right] \\
&\leq 2(\delta_r + e^{\varepsilon}\beta),
\end{aligned}$$

where we used that $\frac{1+e^{\varepsilon}\beta}{1-\delta-e^{\varepsilon}\beta} \leq e^{2\delta+3e^{\varepsilon}\beta}$ and $\frac{1}{1-\delta-e^{\varepsilon}\beta} \leq 2$ whenever $\delta + e^{\varepsilon}\beta < 1/2$. □

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 [EPK14] 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 $\alpha_1 = e^{\varepsilon}/(e^{\varepsilon}+1)$ [EFMRSTT20]. 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 $\frac{\sqrt{n}e^{\varepsilon/2}}{e^{\varepsilon}-1}$ [EPK14; WBLJ17]. 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 $\frac{2\sqrt{n}e^{\varepsilon/2}}{e^{\varepsilon}-1}$ [WBLJ17]. (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 [WHWNXYLQ16; YB18] 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. (This algorithm can also be easily adapted to deletion privacy in which case the results will be nearly identical to symmetric 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, the tests needed for privacy are fooled if the marginals of the PRG are correct. Moreover the accuracy is preserved as long as the bits are randomized in a pairwise independent way. Thus we can simply use a standard derandomization technique for pairwise independent random variables. 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. We note that this approach is a special case of a more general approach is which we associate each j with a non-zero element of an inner product space $\mathbb{F}_q^d$, where q is a prime power. This more general approach (that we describe in Section A) allows to reduce some of the computation overheads in decoding.

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 $\Phi_{j,b}$:

$$\Phi_{j,b} := \{\phi \in \Phi \mid \texttt{bool}(\phi(j)) = b\}. \tag{2}$$

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

Algorithm 3 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 Bernoulli($\alpha_1$)
2: Sample randomly $\phi$ from $\Phi_{j,b}$ defined in eq. (2)
3: Send $\phi$
```

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.

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. 3) 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.*

Algorithm 4 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.
sum $= 0$
for $i \in [n]$ do
    sum$+= \texttt{bool}(\phi^i(j))$
$\tilde{c}_j = \frac{\text{sum} - \alpha_0 n}{\alpha_1 - \alpha_0}$
Return $\tilde{c}_j$
```

Proof. While it is easy to analyze the privacy guarantees of PI-RAPPOR directly it is instructive to show that these guarantees follow from our general compression technique. Specifically, there is a natural way to sample from the reference distribution of RAPPOR relative to which our pairwise PRG fools the density tests given in Lemma 3.2.

To sample from the reference distribution of RAPPOR we pick k values $z_1, \ldots, z_k$ randomly independently and uniformly from $\mathbb{F}_p$ and then output $\texttt{bool}(z_1), \texttt{bool}(z_2), \ldots, \texttt{bool}(z_k)$ (we note that samplability is defined using uniform distribution over binary strings length t as an input but any other distribution can be used instead). By our choice of parameter p and definition of `bool`, this gives k i.i.d. samples from Bernoulli(α_0), which is the reference distribution for RAPPOR. Let $\mathcal{R}$ denote the RAPPOR randomizer. For any $j \in [k]$ and $z' \in \mathbb{F}_p^k$ the ratio of densities at z' satisfies:

$$\frac{\mathbf{Pr}[\mathcal{R}(j) = \mathcal{R}_\emptyset(z')]}{\mathbf{Pr}_{z \sim \mathbb{F}_p^k}[\mathcal{R}_\emptyset(z) = \mathcal{R}_\emptyset(z')]} = \begin{cases} \frac{\alpha_1}{\alpha_0}, & \text{if } \texttt{bool}(z'_j) = 1 \\ \frac{1-\alpha_1}{1-\alpha_0}, & \text{otherwise.} \end{cases}.$$

With probability α_1, PI-RAPPOR algorithm samples ϕ uniformly from $\Phi_{j,1}$ and with probability $1 - \alpha_1$ PI-RAPPOR algorithm samples ϕ uniformly from $\Phi_{j,0}$. This means that PI-RAPPOR is exactly equal to $\mathcal{R}[G]$, where $G \colon \mathbb{F}_p^2 \to \mathbb{F}_p^k$ is defined as $G(\phi) = \phi(1), \phi(2), \ldots, \phi(k)$.

Now to prove that PI-RAPPOR has the same deletion privacy guarantees as RAPPOR it suffices to prove that G 0-fools the tests based on the ratio of densities above. This follows immediately from the fact that $\texttt{bool}(\phi(j))$ for $\phi \sim \Phi$ is distributed in the same way as $\texttt{bool}(z_j)$ for $z \sim \mathbb{F}_p^k$.

To prove that PI-RAPPOR has the same replacement privacy guarantees as RAPPOR we simply use the same reference distribution and apply Lemma 3.7. □

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. Standard results on concentration of sums of independent random variables imply that the bounds on the variance can be translated easily into high probability bounds and also into bounds on the expectation of ℓ_∞, ℓ_1, ℓ_2 errors.

Lemma 4.2. *For any dataset $S \in [k]^n$, the estimate $\tilde{c}$ computed by PI-RAPPOR algorithm (Algs. 3,4) satisfies:*

- $\mathbf{E}[\tilde{c}] = c(S)$
- *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 particular, the expected ℓ_2 squared error is

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

Proof. We first note that

$$\tilde{c}_j = \sum_{i \in [n]} \frac{\texttt{bool}(\phi^i(j)) - \alpha_0}{\alpha_1 - \alpha_0},$$

where ϕ^i is the output of the PI-RAPPOR randomizer on input x_i. Thus to prove the claim about the expectation it is sufficient to prove that for every i,

$$\mathbf{E}\left[\frac{\texttt{bool}(\phi^i(j)) - \alpha_0}{\alpha_1 - \alpha_0}\right] = \texttt{ind}\,(x_i = j)$$

and to prove the claim for variance it is sufficient to prove that

$$\mathbf{Var}\left[\frac{\texttt{bool}(\phi^i(j)) - \alpha_0}{\alpha_1 - \alpha_0}\right] = \texttt{ind}\,(x_i = j)\,\frac{1 - \alpha_0 - \alpha_1}{\alpha_1 - \alpha_0} + \frac{\alpha_0(1-\alpha_0)}{(\alpha_1 - \alpha_0)^2}.$$

If $x_i = j$ then both of these claims follow directly from the fact that, by definition of PI-RAPPOR randomizer, in this case the distribution of $\texttt{bool}(\phi^i(j))$ is Bernoulli(α_1).

If, on the other hand $x_i \neq j$, we use pairwise independence of $\texttt{bool}(\phi(x_i))$ and $\texttt{bool}(\phi(j))$ for $\phi \sim \Phi$ to infer that conditioning the distribution $\texttt{bool}(\phi(x_i)) = b$ (for any b) does not affect the distribution of $\texttt{bool}(\phi(x_i))$. Thus, if $x_i \neq j$ then $\texttt{bool}(\phi^i(j))$ is distributed as Bernoulli(α_0) and we can verify the desired property directly.

Finally,

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

□

For RAPPOR these bounds are stated in [WBLJ17] who also demonstrate that optimizing α_0 and α_1 subject to the replacement privacy parameter being ε while ignoring the first term in the variance (since it is typically dominated by the second term) leads to the asymmetric version $\alpha_0 = 1/(e^\varepsilon + 1)$ and $\alpha_1 = 1/2$. For deletion privacy the optimal setting of $\alpha_0 = 1 - \alpha_1 = 1/(e^\varepsilon + 1)$ follows from standard optimization of the binary randomized response. Thus we obtain 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 $\mathbf{E}\left[\|\tilde{c} - c(S)\|_2^2\right] = n + nk\frac{4e^\varepsilon}{(e^\varepsilon-1)^2}$.*

Note that in the setting where S is sampled i.i.d. from some distribution over $[k]$ defined by frequencies $f_1, \ldots, f_k$, the term $c(S)_j$ in the variance is comparable to sampling variance. This is true since $c(S)_j \approx nf_j$ and for a sum of n Bernoulli random variable with bias $f_j \ll 1$, the variance is $nf_j(1 - f_j) \approx nf_j$. In most practical regimes of frequency estimation with LDP, sampling error is much lower than error introduced by the local randomizers. This justifies optimization of parameters based on the second term alone.

Finally, we analyze the computational and communication costs of PI-RAPPOR. We first bound these for the client.

Lemma 4.5. *PI-RAPPOR randomizer (Alg. 3) can be implemented in $\tilde{O}(\log p)$ time and uses $2\lceil \log_2 p \rceil$ bits of communication.*

Proof. Any function $\phi \in \Phi$ is represented by two elements from $\mathbb{F}_p$ which implies the claimed bound on the communication cost. The running time of PI-RAPPOR is dominated by the time to pick a random and uniform element in $\Phi_{j,b}$. This can be done by picking $\phi_1 \in \mathbb{F}_p$ randomly and uniformly. We then need to pick ϕ_0 randomly and uniformly from the set $\{\phi_0 \mid \texttt{bool}(\phi(j)) = b\}$. Given the result of multiplication $j\phi_1$ this can be done in $O(\log p)$ time. For example for $b = 1$ this set is equal to $\{-j\phi_1, -j\phi_1 + 1, \ldots, -j\phi_1 + \alpha_0 p - 1\}$ where all arithmetic operations are in $\mathbb{F}_p$. The set consists of at most two contiguous ranges of integers and thus a random and uniform element can be chosen in $O(\log p)$ time. Multiplication in $\mathbb{F}_p$ can be done in $O(\log(p) \cdot (\log\log p)^2)$ (*e.g.* [MVOV18]) but in most practical settings standard Montgomery modular multiplication that takes $O(\log^2(p))$ time would be sufficiently fast. □

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 $\mathbb{F}_p$.

Lemma 4.6. *For every $j \in k$, the server-side of PI-RAPPOR (Alg. 4) computes $\tilde{c}_j$ in time $\tilde{O}(n \log p)$. In particular, the entire histogram is computed in time $\tilde{O}(kn \log p)$.*

Note that the construction of the entire histogram on the server is relatively expensive. In Section A we show an alternative algorithm that runs faster when $k \gg n$. For comparison we note that aggregation in the compression schemes in [ASZ19] and [CKÖ20] can be done in $\tilde{O}(n + k)$. However these schemes require $\Omega(k)$ computation on each client and thus the entire system also performs $\Omega(nk)$ computation. They also do not give a frequency oracle since the decoding time of even a single message is linear in k.

Finally we need to discuss how to pick p. In addition to the condition that is p a prime larger than k, our algorithm requires that $\alpha_0 p$ be an integer. We observe that while, in general, we cannot always guarantee that $\alpha_0 = p/(e^\varepsilon + 1)$, by picking p that is a sufficiently large multiple of $\max\{e^\varepsilon, 1/\varepsilon\}$ we get an ε'-DP PI-RAPPOR algorithm for ε' that is slightly smaller than ε (which also implies that its utility is slightly worse). We make this formal below.

Lemma 4.7. *There exists a constant c_0 such that for any $\varepsilon > 0$, $k \in \mathbb{N}$, $\Delta > 0$ and any prime $p \geq c_0 \max\{e^\varepsilon, 1/\varepsilon\}/\Delta$ we have that symmetric PI-RAPPOR with parameter $\alpha_0 = \lceil p/(e^\varepsilon + 1)\rceil/p$ satisfies deletion ε-DP and outputs an estimate that satisfies: for all $j \in [k]$, $\mathbf{Var}[\tilde{c}_j] \leq n\frac{(1+\Delta)e^\varepsilon}{(e^\varepsilon-1)^2}$. Further, PI-RAPPOR with $\alpha_0 = \lceil p/(e^\varepsilon + 1)\rceil/p$ and $\alpha_1 = 1/2$ satisfies replacement ε-DP and outputs an estimate that satisfies: for all $j \in [k]$, $\mathbf{Var}[\tilde{c}_j] = c(S)_j + n\frac{4(1+\Delta)e^\varepsilon}{(e^\varepsilon-1)^2}$.*

Proof. We first note that by our definition, $\alpha_0 p = \lceil p/(e^\varepsilon + 1)\rceil$ and therefore is an integer (as required by PI-RAPPOR). We denote by $\varepsilon' = \ln(1 - p/\alpha_0)$ (so that $\alpha_0 = 1/(e^{\varepsilon'} + 1)$ and note that $\varepsilon' \leq \varepsilon$. Thus the symmetric PI-RAPPOR satisfies ε-DP. We now note that $|1/(e^{\varepsilon'} + 1) - 1/(e^\varepsilon + 1)| \leq 1/p$. This implies that the bound on variance of PI-RAPPOR satisfies:

$$\mathbf{Var}[\tilde{c}_j] = n\frac{\alpha_0(1-\alpha_0)}{(1-2\alpha_0)^2} = n\frac{\frac{1}{e^{\varepsilon'}+1}(1 - \frac{1}{e^{\varepsilon'}+1})}{(1 - 2\frac{1}{e^{\varepsilon'}+1})^2} \leq n\frac{(\frac{1}{e^\varepsilon+1} + \frac{1}{p})(1 - \frac{1}{e^\varepsilon+1})}{(1 - 2\frac{1}{e^{\varepsilon'}+1} - \frac{2}{p})^2}.$$

If $\varepsilon \leq 1$ then $\frac{1}{e^{\varepsilon'}+1} \geq \frac{1}{e+1}$ and $1 - 2\frac{1}{e^{\varepsilon'}+1} \geq \frac{\varepsilon}{e+1}$. Thus the addition/subtraction of $1/p$ to these quantities for $p \geq c_0/(\varepsilon\Delta)$ increases the bound by at most a multiplicative factor $(1 + \Delta)$ (for a sufficiently large constant c_0).

Otherwise (if $\varepsilon > 1$), then $\frac{1}{e^{\varepsilon'}+1} \geq \frac{1}{e^\varepsilon}$ and $1 - 2\frac{1}{e^{\varepsilon'}+1} \geq \frac{e-1}{e+1}$. Thus the addition/subtraction of $1/p$ to these quantities for $p \geq c_0 e^\varepsilon/\Delta$ increases the bound by at most a multiplicative factor $(1 + \Delta)$ (for a sufficiently large constant c_0).

The analysis for replacement DP is analogous. □

In practice, setting $\Delta = 1/100$ will make the loss of accuracy insignificant. Thus we can conclude that PI-RAPPOR with $p \geq c_1 \max\{k, e^\varepsilon, 1/\varepsilon\}$ for a sufficiently large constant c_1 achieves 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 $\Theta(\frac{d}{n\min(\varepsilon,\varepsilon^2)})$ [DJW18; DR19]. 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 [DR16; BS16; Mir17] guarantees, then standard Gaussian noise addition achieves the asymptotically optimal bound. When $\varepsilon \leq 1$, the randomizer of Duchi et al. [DJW18] (which we refer to as `PrivHS`) also achieves the optimal $O(\frac{d}{n\varepsilon^2})$ bound. Recent work of Erlingsson et al. [EFMRSTT20] 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. [BDFKR19] 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 3.5 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 $me^{\varepsilon/m}$. Below we describe the reduction and state the resulting guarantees.

Lemma 5.1. *Assume that for some $\varepsilon > 0$ there exists a local ε-DP randomizer $\mathcal{R}_\varepsilon \colon \mathbb{B}^d \to Y$ and a decoding procedure* $\mathtt{decode}\colon Y \to \mathcal{R}^d$ *that for all $\mathbf{x} \in \mathbb{B}^d$, satisfies: $\mathbf{E}[\mathtt{decode}(\mathcal{R}_\varepsilon(\mathbf{x}))] = \mathbf{x}$ and $\mathbf{E}[\|\mathtt{decode}(\mathcal{R}_\varepsilon(\mathbf{x})) - \mathbf{x}\|_2^2] \leq \alpha_\varepsilon$. Further assume that $\mathcal{R}_\varepsilon$ uses ℓ bits of communication and runs in time T. Then for every integer $m \geq 2$ there is a local $(m\varepsilon)$-DP randomizer $\mathcal{R}_\varepsilon^m \colon \mathbb{B}^d \to Y^m$ and decoding procedure $\mathtt{decode}^m \colon Y^m \to \mathcal{R}^d$ that uses $m\ell$ bits of communication, runs in time mT and for every $\mathbf{x} \in \mathbb{B}^d$ satisfies: $\mathbf{E}[\mathtt{decode}^m(\mathcal{R}'_\varepsilon(\mathbf{x}))] = \mathbf{x}$ and $\mathbf{E}[\|\mathtt{decode}^m(\mathcal{R}_\varepsilon^m(\mathbf{x})) - \mathbf{x}\|_2^2] \leq \frac{\alpha_\varepsilon}{m}$.*

In particular, if for every $\varepsilon \in (1/2, 1]$, $\alpha_\varepsilon \leq \frac{cd}{\varepsilon^2}$ for some constant c, then for every $\varepsilon > 0$ there is a local ε-DP randomizer $\mathcal{R}'_\varepsilon$ and decoding procedure $\mathtt{decode}'$ that uses $\lceil\varepsilon\rceil\ell$ bits of communication, runs in time $\lceil\varepsilon\rceil T$ and for every $\mathbf{x} \in \mathbb{B}^d$ satisfies: $\mathbf{E}[\mathtt{decode}'(\mathcal{R}'_\varepsilon(\mathbf{x}))] = \mathbf{x}$ and $\mathbf{E}[\|\mathtt{decode}'(\mathcal{R}'_\varepsilon(\mathbf{x})) - \mathbf{x}\|_2^2] \leq \frac{2cd}{\min\{\varepsilon,\varepsilon^2\}}$.

Proof. The randomizer $\mathcal{R}_\varepsilon^m(\mathbf{x})$ runs $\mathcal{R}_\varepsilon(\mathbf{x})$ m times independently to obtain $y_1, \ldots, y_m$ and outputs these values. To decode we define $\mathtt{decode}^m(y_1, \ldots, y_m) := \frac{1}{m}(\mathtt{decode}(y_1) + \cdots + \mathtt{decode}(y_m))$. By (simple) composition of differential privacy, $\mathcal{R}_\varepsilon^m$ is (εm)-DP. The utility claim follows directly from linearity of expectation and independence of the estimates:

$$\mathbf{E}[\|\mathtt{decode}^m(\mathcal{R}^m_\varepsilon(\mathbf{x})) - \mathbf{x}\|_2^2] = \frac{1}{m} \cdot \mathbf{E}[\|\mathtt{decode}(\mathcal{R}_\varepsilon(\mathbf{x})) - \mathbf{x}\|_2^2] \leq \frac{\alpha_\varepsilon}{m}.$$

For the second part of the claim we define $\mathcal{R}'_\varepsilon$ as follows. For $\varepsilon \leq 1$, $\mathcal{R}'_\varepsilon(\mathbf{x})$ just outputs $\mathcal{R}_\varepsilon(\mathbf{x})$ and in this case $\mathtt{decode}'$ is the same as $\mathtt{decode}$. For $\varepsilon > 1$, we let $m = \lceil \varepsilon \rceil$ and apply the lemma to $\mathcal{R}_{\varepsilon'}$ for $\varepsilon' = \varepsilon / \lceil \varepsilon \rceil$. Note that $\varepsilon' \in (1/2, 1)$ and therefore the resulting bound on variance is

$$\mathbf{E}[\|\mathtt{decode}'(\mathcal{R}'_\varepsilon(\mathbf{x})) - \mathbf{x}\|_2^2] \leq \frac{1}{\lceil \varepsilon \rceil} \frac{cd}{\varepsilon'^2} = \frac{cd}{\varepsilon \varepsilon'} \leq \frac{2cd}{\varepsilon}.$$

□

For example, by using the reduction in Lemma 5.1, we can reduce the computational cost to $\tilde{O}(\lceil \varepsilon \rceil d)$ while increasing the communication to $O(\lceil \varepsilon \rceil \log d)$. The server side reconstruction now requires sampling and averaging $n \lceil \varepsilon \rceil$ d-dimensional vectors. Thus the server running time is $\tilde{O}(nd\varepsilon)$.

This reduction allows one to achieve different trade-offs between computation, communication, and closeness to the accuracy of the original randomizer. As an additional benefit, 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 [EFMRSTT20]. This bypasses the need for our compression algorithm and makes the privacy guarantees unconditional.

Remark 5.2. *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).*

5.1 Empirical Comparison of Mean Estimation Algorithms

While `PrivUnit`, `SQKR` and the repeated version of `PrivHS` (using Lemma 5.1) 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. As an example, for $d = 2000, \varepsilon = 8$, $B^2 \approx 3145$ so that the proxy is accurate up to 3.5% (and the proxy is even more accurate when d is significantly larger which is the typical setting where compression). 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 [BDFKR19] 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

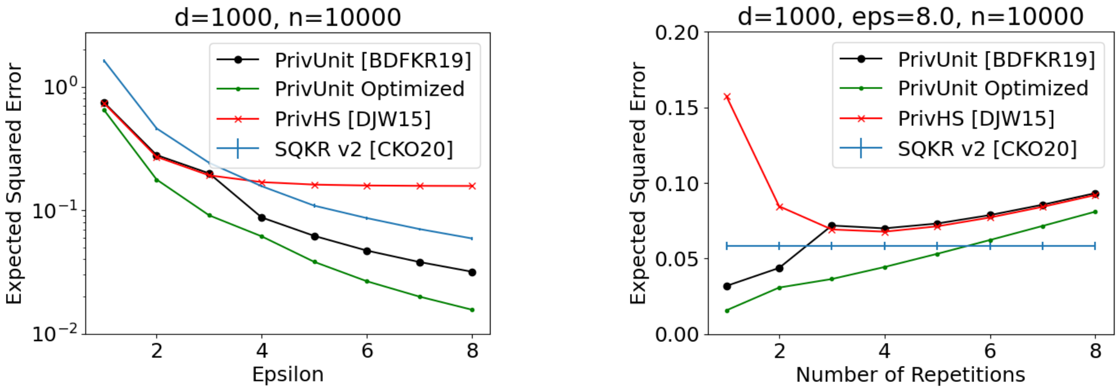

Figure 1: (Left) Expected ℓ_2^2 error of mechanisms `PrivHS`, `PrivUnit`, `PrivUnitOptimized` and `SQKR` for values of ε between 1 and 8. (Right) Expected ℓ_2^2 error of mechanisms `PrivHS`, `PrivUnit` and `PrivUnitOptimized` for a total $\varepsilon = 8$, as a function of the number of repetitions of the mechanism with a proportionately smaller ε. The `SQKR` v2 line is for a single run with $\varepsilon = 8$ without splitting. Both plots use $n = 10,000, d = 1,000$, and 95% error bars for `SQKR` are computed based on 10 trials.

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 (Left) 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 Lemma 5.1. 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. Figure 1 (Right) shows the results for `PrivHS`, `PrivUnit` and `PrivUnitOptimized`. We plot the single repetition version of `SQKR` for comparison. The `SQKR` algorithm does not get more efficient for smaller ε and thus splitting it makes it worse in every aspect. As its error grows quickly with splitting, we do not plot the split version of `SQKR` in these plots. The results demonstrate that splitting does have some cost in terms of expected squared error, and going from $\varepsilon = 8$ to two runs of $\varepsilon = 4$ costs us about $2\times$ in expected squared error, and that the error continues to increase as we split more. These results can inform picking an appropriate point on the computation cost-error tradeoff and suggest that for ε around 8, the choice in most cases will be between not splitting and splitting into two mechanisms. Note that even with two or three repetitions, `PrivUnitOptimized` has $2-3\times$ smaller error compared to `PrivHS` and `SQKR`. For `PrivHS`, the sweet spot seems to be splitting into multiple mechanisms each with $\varepsilon \approx 2$.

References

[ACGMMTZ16] M. Abadi, A. Chu, I. J. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. "Deep Learning with Differential Privacy". In: *Proceedings of the 2016*

ACM SIGSAC Conference on Computer and Communications Security (CCS). 2016, pp. 308–318.

[AGLTV17] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic. "QSGD: Communication-Efficient SGD via Gradient Quantization and Encoding". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017, pp. 1709–1720. URL: `https://proceedings.neurips.cc/paper/2017/file/6c340f25839e6acdc73414517203f5f0-Paper.pdf`.

[App17] Apple's Differential Privacy Team. "Learning with Privacy at Scale". In: *Apple Machine Learning Journal* 1.9 (Dec. 2017).

[AS19] J. Acharya and Z. Sun. "Communication complexity in locally private distribution estimation and heavy hitters". In: *arXiv preprint arXiv:1905.11888* (2019).

[ASYKM18] N. Agarwal, A. T. Suresh, F. X. X. Yu, S. Kumar, and B. McMahan. "cpSGD: Communication-efficient and differentially-private distributed SGD". In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Vol. 31. Curran Associates, Inc., 2018, pp. 7564–7575. URL: `https://proceedings.neurips.cc/paper/2018/file/21ce689121e39821d07d04faab328370-Paper.pdf`.

[ASZ19] J. Acharya, Z. Sun, and H. Zhang. "Hadamard Response: Estimating Distributions Privately, Efficiently, and with Little Communication". In: ed. by K. Chaudhuri and M. Sugiyama. Vol. 89. Proceedings of Machine Learning Research. PMLR, 2019, pp. 1120–1129.

[BBGN19] B. Balle, J. Bell, A. Gascón, and K. Nissim. "The Privacy Blanket of the Shuffle Model". In: *Advances in Cryptology – CRYPTO 2019*. Ed. by A. Boldyreva and D. Micciancio. Cham: Springer International Publishing, 2019, pp. 638–667. ISBN: 978-3-030-26951-7.

[BDFKR19] A. Bhowmick, J. Duchi, J. Freudiger, G. Kapoor, and R. Rogers. *Protection Against Reconstruction and Its Applications in Private Federated Learning*. 2019. arXiv: `1812.00984 [stat.ML]`.

[BEMMRLRKTS17] A. Bittau, U. Erlingsson, P. Maniatis, I. Mironov, A. Raghunathan, D. Lie, M. Rudominer, U. Kode, J. Tinnes, and B. Seefeld. "Prochlo: Strong Privacy for Analytics in the Crowd". In: *Proceedings of the 26th Symposium on Operating Systems Principles*. SOSP '17. 2017, pp. 441–459.

[BNS19] M. Bun, J. Nelson, and U. Stemmer. "Heavy hitters and the structure of local privacy". In: *ACM Transactions on Algorithms (TALG)* 15.4 (2019), pp. 1–40.

[BNST20] R. Bassily, K. Nissim, U. Stemmer, and A. Thakurta. "Practical Locally Private Heavy Hitters." In: *Journal of Machine Learning Research* 21.16 (2020), pp. 1–42.

[BS15] R. Bassily and A. Smith. "Local, private, efficient protocols for succinct histograms". In: *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*. 2015, pp. 127–135.

[BS16] M. Bun and T. Steinke. "Concentrated Differential Privacy: Simplifications, Extensions, and Lower Bounds". In: *Proceedings, Part I, of the 14th International Conference on Theory of Cryptography - Volume 9985*. Berlin, Heidelberg: Springer-Verlag, 2016, 635–658. ISBN: 9783662536407. URL: `https://doi.org/10.1007/978-3-662-53641-4_24`.

[BST14] R. Bassily, A. Smith, and A. Thakurta. "Private Empirical Risk Minimization, Revisited". In: *CoRR* abs/1405.7085 (2014). URL: `http://arxiv.org/abs/1405.7085`.

[CKÖ20] W.-N. Chen, P. Kairouz, and A. Özgür. “Breaking the Communication-Privacy-Accuracy Trilemma”. In: *arXiv preprint arXiv:2007.11707* (2020).

[CSUZZ19] A. Cheu, A. Smith, J. Ullman, D. Zeber, and M. Zhilyaev. “Distributed Differential Privacy via Shuffling”. In: *Advances in Cryptology – EUROCRYPT 2019*. Ed. by Y. Ishai and V. Rijmen. Cham: Springer International Publishing, 2019, pp. 375–403. ISBN: 978-3-030-17653-2.

[DJW18] J. C. Duchi, M. I. Jordan, and M. J. Wainwright. “Minimax optimal procedures for locally private estimation”. In: *Journal of the American Statistical Association* 113.521 (2018), pp. 182–201.

[DKY17] B. Ding, J. Kulkarni, and S. Yekhanin. “Collecting Telemetry Data Privately”. In: *31st Conference on Neural Information Processing Systems (NIPS)*. 2017, pp. 3574–3583.

[DMNS06] C. Dwork, F. McSherry, K. Nissim, and A. Smith. “Calibrating noise to sensitivity in private data analysis”. In: *TCC*. 2006, pp. 265–284.

[DR14] C. Dwork and A. Roth. *The Algorithmic Foundations of Differential Privacy*. Vol. 9. 3-4. 2014, pp. 211–407. URL: `http://dx.doi.org/10.1561/0400000042`.

[DR16] C. Dwork and G. N. Rothblum. “Concentrated Differential Privacy”. In: *ArXiv* abs/1603.01887 (2016).

[DR19] J. Duchi and R. Rogers. “Lower Bounds for Locally Private Estimation via Communication Complexity”. In: *Proceedings of the Thirty-Second Conference on Learning Theory*. Ed. by A. Beygelzimer and D. Hsu. Vol. 99. Proceedings of Machine Learning Research. Phoenix, USA: PMLR, 2019, pp. 1161–1191. URL: `http://proceedings.mlr.press/v99/duchi19a.html`.

[EFMRSTT20] U. Erlingsson, V. Feldman, I. Mironov, A. Raghunathan, S. Song, K. Talwar, and A. Thakurta. “Encode, Shuffle, Analyze Privacy Revisited: Formalizations and Empirical Evaluation”. In: (2020). arXiv: `2001.03618 [cs.CR]`.

[EFMRTT19] U. Erlingsson, V. Feldman, I. Mironov, A. Raghunathan, K. Talwar, and A. Thakurta. “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. San Diego, California: Society for Industrial and Applied Mathematics, 2019, 2468–2479.

[EGS03] A. V. Evfimievski, J. Gehrke, and R. Srikant. “Limiting privacy breaches in privacy preserving data mining”. In: *PODS*. 2003, pp. 211–222.

[EPK14] Ú. Erlingsson, V. Pihur, and A. Korolova. “Rappor: Randomized aggregatable privacy-preserving ordinal response”. In: *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*. 2014, pp. 1054–1067.

[FGV15] V. Feldman, C. Guzman, and S. Vempala. “Statistical Query Algorithms for Mean Vector Estimation and Stochastic Convex Optimization”. In: *CoRR* abs/1512.09170 (2015). Extended abstract in SODA 2017. URL: `http://arxiv.org/abs/1512.09170`.

[FMT20] V. Feldman, A. McMillan, and K. Talwar. “Hiding Among the Clones: A Simple and Nearly Optimal Analysis of Privacy Amplification by Shuffling”. In: *CoRR* abs/2012.12803 (2020). arXiv: `2012.12803`. URL: `https://arxiv.org/abs/2012.12803`.

[FTMARRK20] F. Faghri, I. Tabrizian, I. Markov, D. Alistarh, D. M. Roy, and A. Ramezani-Kebrya. “Adaptive Gradient Quantization for Data-Parallel SGD”. In: *Advances in Neural Information Processing Systems*. Vol. 33. 2020.

[GDDKS20] A. M. Girgis, D. Data, S. Diggavi, P. Kairouz, and A. T. Suresh. *Shuffled Model of Federated Learning: Privacy, Communication and Accuracy Trade-offs*. 2020. arXiv: `2008.07180 [cs.LG]`.

[GKMM19] V. Gandikota, D. Kane, R. K. Maity, and A. Mazumdar. “vqsgd: Vector quantized stochastic gradient descent”. In: *arXiv preprint arXiv:1911.07971* (2019).

[HKR12] J. Hsu, S. Khanna, and A. Roth. “Distributed private heavy hitters”. In: *International Colloquium on Automata, Languages, and Programming*. Springer. 2012, pp. 461–472.

[Kai+19] P. Kairouz et al. *Advances and Open Problems in Federated Learning*. 2019. arXiv: `1912.04977 [cs.LG]`.

[Kas] *An implementation of Kashine based mean estimation scheme*. `https://github.com/WeiNingChen/Kashin-mean-estimation`. Accessed: 2021-02-17.

[KBR16] P. Kairouz, K. Bonawitz, and D. Ramage. “Discrete distribution estimation under local privacy”. In: *arXiv preprint arXiv:1602.07387* (2016).

[LLW06] M. Luby, M. G. Luby, and A. Wigderson. *Pairwise independence and derandomization*. Vol. 4. Now Publishers Inc, 2006.

[LV10] Y. Lyubarskii and R. Vershynin. “Uncertainty principles and vector quantization”. In: *Information Theory, IEEE Transactions on* 56.7 (2010), pp. 3491–3501.

[Mir17] I. Mironov. “Rényi Differential Privacy”. In: *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*. 2017, pp. 263–275.

[MMPRTV10] A. McGregor, I. Mironov, T. Pitassi, O. Reingold, K. Talwar, and S. Vadhan. “The Limits of Two-Party Differential Privacy”. In: *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*. 2010, pp. 81–90.

[MPRV09] I. Mironov, O. Pandey, O. Reingold, and S. Vadhan. “Computational Differential Privacy”. In: *Advances in Cryptology - CRYPTO 2009*. Ed. by S. Halevi. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 126–142. ISBN: 978-3-642-03356-8.

[MRTZ18] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang. “Learning Differentially Private Recurrent Language Models”. In: *6th International Conference on Learning Representations, ICLR 2018*. 2018.

[MS06] N. Mishra and M. Sandler. “Privacy via Pseudorandom Sketches”. In: *Proceedings of the Twenty-Fifth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*. PODS ’06. Chicago, IL, USA: Association for Computing Machinery, 2006, 143–152. ISBN: 1595933182. URL: `https://doi.org/10.1145/1142351.1142373`.

[MT20] P. Mayekar and H. Tyagi. “Limits on Gradient Compression for Stochastic Optimization”. In: *2020 IEEE International Symposium on Information Theory (ISIT)*. 2020, pp. 2658–2663.

[MVOV18] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone. *Handbook of applied cryptography*. CRC press, 2018.

[Nis92] N. Nisan. “Pseudorandom generators for space-bounded computation”. In: *Combinatorica* 12.4 (1992), pp. 449–461.

[SYKM17] A. T. Suresh, F. X. Yu, S. Kumar, and H. B. McMahan. “Distributed Mean Estimation with Limited Communication”. In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by D. Precup and Y. W. Teh. Vol. 70. Proceedings of Machine Learning Research. International Convention Centre, Sydney, Australia: PMLR, 2017, pp. 3329–3337. URL: `http://proceedings.mlr.press/v70/suresh17a.html`.

[War65] S. L. Warner. “Randomized response: A survey technique for eliminating evasive answer bias”. In: *Journal of the American Statistical Association* 60.309 (1965), pp. 63–69.

[WBLJ17] T. Wang, J. Blocki, N. Li, and S. Jha. "Locally Differentially Private Protocols for Frequency Estimation". In: *26th USENIX Security Symposium (USENIX Security 17)*. Vancouver, BC: USENIX Association, Aug. 2017, pp. 729–745. ISBN: 978-1-931971-40-9. URL: `https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/wang-tianhao`.

[WHWNXYLQ16] S. Wang, L. Huang, P. Wang, Y. Nie, H. Xu, W. Yang, X.-Y. Li, and C. Qiao. "Mutual information optimally local private discrete distribution estimation". In: *arXiv preprint arXiv:1607.08025* (2016).

[YB18] M. Ye and A. Barg. "Optimal schemes for discrete distribution estimation under locally differential privacy". In: *IEEE Transactions on Information Theory* 64.8 (2018), pp. 5662–5676.

A A generalization of PI-RAPPOR

In the more general version of RAPPOR we let q by any prime power and assume that $\alpha_0 q$ is an integer. We will rely on some arbitrary order on the elements of $\mathbb{F}_q$ and denote $\alpha_0 q$ smallest elements as F_1. We denote the indicator function of the event $z \in F_1$ by $\texttt{bool}(z)$. As before, for a randomly and uniformly chosen element $z \in F$ we have that $\texttt{bool}(z)$ is distributed as Bernoulli random variable with bias α_0. For efficiency, the order needs to be chosen in a way that allows computing $\texttt{bool}(z)$ and generating a random element of the field in some range in $O(\log q)$ time.

We will associate each index $j \in [k]$ with a distinct *non-zero* element of $z(j) \in \mathbb{F}_q^d$, where $d := \lceil \log_q(k+1) \rceil$ (in particular, $q^d \leq k+1 < q^{d+1}$). We can describe an affine function ϕ over $\mathbb{F}_q^d$ using the vector of its $d+1$ coefficients: $\phi_0, \ldots, \phi_d$ and for $z \in \mathbb{F}_q^d$ we define $\phi(z) = \phi_0 + \sum_{u \in [d]} z_u \phi_u$, where addition and multiplication are in the field $\mathbb{F}_q$. For brevity we will also overload $\phi(j) := \phi(z(j))$. Each such function encodes a vector in $\mathbb{F}_q^k$ as $\phi([k]) := \phi(1), \phi(2), \ldots, \phi(k)$. The family of functions defined by all $d+1$ tuples in $\mathbb{F}_q$ is defined as $\Phi := \{\phi \mid \phi \in \mathbb{F}_q^{d+1}\}$. For a randomly chosen function from this family the values of the function on two distinct non-zero values uniformly distributed and pairwise independent: for any $j_1 \neq j_2 \in [k]$ and $a_1, a_2 \in \mathbb{F}_q$ we have that

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

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 $\Phi_{j,b}$:

$$\Phi_{j,b} := \{\phi \in \Phi \mid \texttt{bool}(\phi(j)) = b\}. \tag{3}$$

The generalization of the PI-RAPPOR randomizer is described below.

Algorithm 5 General PI-RAPPOR randomizer

```
Input: An index $j \in [k]$, $0 < \alpha_0 < \alpha_1 < 1$, prime power $q$ s.t. $\alpha_0 q \in \mathbb{N}$
$d = \lceil \log_q(k+1) \rceil$
Sample Bernoulli $b$ with bias $\alpha_1$
Sample randomly $\phi$ from $\Phi_{j,b}$ defined in eq. (3)
Send $\phi$
```

The server side of the frequency estimation can be done exactly as before. We can convert ϕ to $\texttt{bool}(\phi(j))$ and then aggregate the results. In addition, we describe an alternative algorithm that runs in time $\tilde{O}(k\|\Phi\| + n)$. This algorithm is faster than direct computation when $|\Phi| < n$. In this case we can first count the number of times each ϕ is used and then decode each ϕ only once. This approach is based on an idea in [BNST20] which also relies on pairwise independence to upper bound the total number of encodings. We note that in the simpler version of PI-RAPPOR $|\Phi| \geq (k+1)^2$, whereas in the generalized version $|\Phi|$ can be as low $(k+1) \cdot (e^{\varepsilon}+1)$.

Algorithm 6 Private histograms with PI-RAPPOR

```
Input: $0 < \alpha_0 < \alpha_1 < 1$, prime power $q$
Receive $\phi^1, \dots, \phi^n$ from $n$ users.
for $\phi \in \Phi$ do
    $n_\phi = 0$
for $i \in [n]$ do
    $n_{\phi^i} += 1$
sum $= -\alpha_0, \dots, -\alpha_0$
for $\phi \in \Phi$ do
    sum$+= n_\phi \cdot \texttt{bool}(\phi^i)$
$\tilde{c} = \frac{1}{\alpha_1 - \alpha_0}$sum
Return $\tilde{c}$
```

It is easy to see that pairwise-independence implies that privacy and utility guarantees of the generalized PI-RAPPOR are the same as for the simpler version we described before. The primary difference is in the computational costs.

Lemma A.1. *PI-RAPPOR randomizer (Alg. 5) can be implemented in $\tilde{O}(\log k)$ time and uses $\lceil \log_2 |\Phi| \rceil \leq \log_2 k + 2\log_2 q + 1$ bits of communication.*

Proof. The running time of PI-RAPPOR is dominated by the time to pick a random and uniform element in $\Phi_{j,b}$. This can be done by picking $\phi_1, \dots, \phi_d \in \mathbb{F}_p$ randomly and uniformly. We then need to pick ϕ_0 randomly and uniformly from the set $\{\phi_0 \mid \texttt{bool}(\phi(j)) = b\}$. Given the result of inner product $\sum_{i \in [d]} z(j)_i \phi_i$ this can be done in $O(\log p)$ time as explained in the proof of Lemma 4.5. The computation of the inner product can be done in $d \cdot \tilde{O}(\log q) = \tilde{O}(\log k)$. □

The analysis of the running time of the aggregation algorithm Alg. 6 follows from the discussion above.

Lemma A.2. *The server-side of the histogram construction for generalized PI-RAPPOR (Alg. 6) can be done in time $\tilde{O}(n + k|\Phi| \log k)$.*

The running time depends on $|\Phi| \leq kq^2$ and thus also depends on the choice of q. The effect of the choice of q on the utility of Algorithm 6 is the same as the effect of the choice of p on the utility of the simple PI-RAPPOR (given in Lemma 4.7). Thus we can assume that $q = O(\max\{e^{\varepsilon}, 1/\varepsilon\})$.