Differential Privacy in the Shuffle Model: A Survey of Separations

Albert Cheu
June 1, 2020

Abstract

Classical work in differential privacy operates in extremes of trust assumptions: either all users give their data to a single party or they have no trust in any party. The shuffle model posits an intermediary level of trust in hopes of gaining an intermediary level of accuracy. This survey gives an overview of results in the shuffle model which validate that trade-off.

1 Introduction

Many differentially private algorithms operate in the central model, also known as the trusted curator model. Here, a single analyzer has data on all users in question and the objective is to compute in a manner that is distributionally insensitive to any one user’s datum. But the fact that all users give their data to one party means that there is a single point of failure: if the analyzer is corrupt or otherwise breached, the privacy of all users will be violated.

There are a number of ways to model weaker trust in the analyzer, chief among them being the local model. Here, the dataset is a distributed object where each user holds a single row. To preserve their own privacy, each user executes a randomizing function on their data point and submits the resulting outputs to the analyzer. Due to the severely weakened signal, there are a number of lower bounds on the error of locally private protocols that strongly separate the local model from the central model [11, 14, 10, 1, 27]. Additionally, locally private protocols are particularly susceptible to manipulation attacks: by sending carefully distributed messages, malicious users can skew tests and estimates of distributions beyond simply changing the input of the protocol [15].

These negative results motivate the exploration of alternative models. One tactic is to apply secure multi-party computation (MPC) to simulate arbitrary central model algorithms (see e.g. Dwork et al. [17]). However, current methods of simulating generic circuits impose large costs in terms of computation and communication and are difficult to scale and maintain. Another tactic is to identify a common operation in central model algorithms and then implement it with a cryptographic primitive, possibly with MPC.

Because statistical computations tend to be commutative (e.g. mean), it is without loss of accuracy to apply a uniformly random permutation. On the other hand, if the values being permuted are sent by users, privacy can only be enhanced because an adversary cannot identify the source of any value. To be more specific, users in the shuffle model send randomized messages to a shuffler which uniformly permutes the messages before sending to the analyzer. It is this shuffled collection of messages that needs to satisfy differential privacy: altering one user’s data point must not greatly change the distribution of the shuffled messages.

To have differential privacy, a user needs only to trust that the shuffler acts as intended and sufficiently many of their peers follow the protocol. A more detailed description of a shuffler and its implementation details appears in the work of Bittau et al. [12].
The goal of this survey is to summarize the surge of work in the shuffle model, placing emphasis on separation results. To this end, the survey is structured as follows. We first establish the requisite privacy and model definitions. Next we contrast local model lower bounds with shuffle model upper bounds: there are four problems for which additive error and sample complexity are much lower in the shuffle model. We close the main body with techniques to show that the shuffle model (under natural constraints) is weaker than the central model. In the Appendix, we present results by Balle et al. [7, 8] and Ghazi et al. [23, 22, 21] concerning communication-efficient shuffle protocols. These inspire confidence in the practicality of the model.

2 Preliminaries

We will use the notation \([k] = \{1, 2, \ldots, k\}, \mathbb{N} = \{1, 2, \ldots\}\). We define a dataset \(\vec{x} \in \mathcal{X}^n\) to be an ordered tuple of \(n\) rows where each row is drawn from a data universe \(\mathcal{X}\) and corresponds to the data of one user. Two datasets \(\vec{x}, \vec{x}' \in \mathcal{X}^n\) are considered neighbors (denoted as \(\vec{x} \sim \vec{x}'\)) if they differ in at most one row.

**Definition 1 (Differential Privacy [18]).** An algorithm \(M : \mathcal{X}^n \rightarrow \mathcal{Z}\) satisfies \((\varepsilon, \delta)\)-differential privacy if, for every pair of neighboring datasets \(\vec{x}\) and \(\vec{x}'\), and every subset \(T \subseteq \mathcal{Z}\),

\[
P[M(\vec{x}) \in T] \leq e^\varepsilon \cdot P[M(\vec{x}') \in T] + \delta.
\]

When \(\delta > 0\), we say \(M\) satisfies approximate differential privacy. When \(\delta = 0\), \(M\) satisfies pure differential privacy and we omit the \(\delta\) parameter.

Because this definition assumes that the algorithm \(M\) has “central” access to compute on the entire raw dataset, we sometimes call this central differential privacy. Two common facts about differentially privacy will be useful; proofs appear in Chapter 2 of the survey of Dwork and Roth [19]. First, privacy is preserved under post-processing.

**Fact 2.** For \((\varepsilon, \delta)\)-differentially private algorithm \(M : \mathcal{X}^n \rightarrow \mathcal{Z}\) and arbitrary random function \(f : \mathcal{Z} \rightarrow \mathcal{Z}'\), \(f \circ M\) is \((\varepsilon, \delta)\)-differentially private.

This means that any computation based solely on the output of a differentially private function does not affect the privacy guarantee. Second, privacy composes neatly.

**Fact 3.** For \((\varepsilon_1, \delta_1)\)-differentially private \(M_1\) and \((\varepsilon_2, \delta_2)\)-differentially private \(M_2\), \(M_3\) defined by \(M_3(D) = (M_1(D), M_2(D))\) is \((\varepsilon_1 + \varepsilon_2, \delta_1 + \delta_2)\)-differentially private.

One useful centrally private algorithm is the binomial mechanism.

**Lemma 4 (Binomial Mechanism [17, 22]).** Let \(f : \mathcal{X}^n \rightarrow \mathcal{Z}\) be a 1-sensitive function, i.e. \(|f(\vec{x}) - f(\vec{x}')| \leq 1\) for all neighboring datasets \(\vec{x}, \vec{x}' \in \mathcal{X}^n\). Fix any \(\ell \in \mathbb{N}\), \(p \in (0, 1)\), \(\varepsilon > 0\), and \(\delta < (0, 1)\) such that

\[
\ell \cdot \min(p, 1 - p) \geq 10 \cdot \left(\frac{e^{\varepsilon + 1}}{e^{\varepsilon + 1} - 1}\right)^2 \cdot \ln\left(\frac{1}{\delta}\right).
\]

The algorithm that samples \(\eta \sim \text{Bin}(\ell, p)\) and outputs \(f(\vec{x}) + \eta\) is \((\varepsilon, \delta)\)-differentially private. The error is \(O(\frac{1}{\ell} \cdot \sqrt{\log \frac{1}{\delta}})\).

We will also use the geometric mechanism. Let \(\text{SG}(\varepsilon)\) denote the symmetric geometric distribution with parameter \(\varepsilon\). For every \(\nu \in \mathbb{Z}\), it has mass \(e^{-\nu} \cdot \frac{\varepsilon^{\nu - 1}}{e^{\nu + 1}}\). It is a discrete analogue of the Laplace distribution with mean 0.

**Lemma 5 (Geometric Mechanism [25]).** Let \(f : \mathcal{X}^n \rightarrow \mathcal{Z}\) be a 1-sensitive function, i.e. \(|f(\vec{x}) - f(\vec{x}')| \leq 1\) for all neighboring datasets \(\vec{x}, \vec{x}' \in \mathcal{X}^n\). Then for \(\varepsilon > 0\), the algorithm that samples \(\eta \sim \text{SG}(\varepsilon)\) then outputs \(f(\vec{x}) + \eta\) is \(\varepsilon\)-differentially private. The error is \(O(\frac{1}{\ell})\).
2.1 The Shuffle Model

**Definition 6** (Shuffle Model [12, 16]). A protocol $P$ in the shuffle model consists of three randomized algorithms:

- A *randomizer* $R : X \rightarrow Y^*$ mapping data to (possibly variable-length) vectors. The length of the vector is the number of messages sent. If, on all inputs, the probability of sending $m$ messages is 1, then we have an $m$-message protocol.

- A *shuffler* $S : Y^* \rightarrow Y^*$ that concatenates message vectors and then applies a uniformly random permutation to the messages.

- An *analyzer* $A : Y^* \rightarrow Z$ that computes on a permutation of messages.

As $S$ is the same in every protocol, we identify each shuffle protocol by $P = (R, A)$. We define its execution on input $\vec{x} \in X^n$ as $P(\vec{x}) := A(S(R(x_1), \ldots, R(x_n)))$.

We assume that $R$ and $A$ have access to $n$ and an arbitrary amount of public randomness.

With this setup, we use the following definition of shuffle differential privacy.

**Definition 7** (Shuffle Differential Privacy [16]). A protocol $P = (R, A)$ is $(\epsilon, \delta)$-shuffle differentially private if, for all $n \in \mathbb{N}$, the algorithm $(S \circ R^n)(\vec{x}) := S(R(x_1), \ldots, R(x_n))$ is $(\epsilon, \delta)$-differentially private. The privacy guarantee is over the internal randomness of the users’ randomizers and not the public randomness of the shuffle protocol.

For brevity, we typically call these protocols “shuffle private.”

Note that Definition 7 assumes all users follow the protocol. Ideally, the definition should account for malicious users that aim to make the protocol less private: the parameters should degrade smoothly with the number of malicious users. A simple attack is for such users to drop out: for $\gamma \leq 1$, let $S \circ R^n(\gamma)$ denote the case where only $\gamma n$ out of $n$ users execute $R$. Because the behavior of the randomizer may depend on $n$, $S \circ R^n$ may satisfy a particular level of differential privacy but $S \circ R^n(\gamma)$ may not.\(^1\) This motivates a robust variant of shuffle privacy.

**Definition 8** (Robust Shuffle Differential Privacy [6]). Fix any $\gamma \in (0, 1]$. A protocol $P = (R, A)$ is $(\epsilon, \delta, \gamma)$-robustly shuffle differentially private if, for all $n \in \mathbb{N}$ and $\gamma' \in \{1/n, 2/n, \ldots, 1\} \cap \{\gamma, 1\}$, the algorithm $S \circ R^{\gamma n}$ is $(\epsilon, \delta)$-differentially private. In other words, $P$ guarantees $(\epsilon, \delta)$-shuffle privacy whenever at least a $\gamma$ fraction of users follow the protocol.

As with generic shuffle differential privacy, we often shorthand this as “robust shuffle privacy.” Note that we define robustness with regard to privacy rather than accuracy. A robustly shuffle private protocol promises its users that their privacy will not suffer much from a limited fraction of malicious users. But it does not make any guarantees about the accuracy of the protocol; we will state our accuracy guarantees under the assumption that all users follow the protocol.

3 Separations between Local & Robust Shuffle Privacy

In this section, we will introduce four problems. For each problem, we will state a lower bound in the local model and then describe a protocol in the shuffle model that breaks through that bound. To simplify the presentation, we will assume $\epsilon < 1$ and $\delta = \Theta(1/n)$.

\(^1\)Note that, with respect to differential privacy, dropping out is “the worst” malicious users can do. This is because adding messages from malicious users to those from honest users is a post-processing of $S \circ R^n$. If $S \circ R^n$ is already differentially private for the outputs of the $\gamma n$ users alone, then differential privacy’s resilience to post-processing (Fact 2) ensures that adding other messages does not affect this guarantee. Hence, it is without loss of generality to focus on drop-out attacks.
3.1 Binary Sums

In this setting, each user \( i \) has a bit \( x_i \in \{0, 1\} \) and the objective is to compute the sum. Dating back to Warner [29], randomized response is the canonical local protocol for this problem. The randomizer is below:

\[
R_{RR}(x_i) := \begin{cases} 
\text{Ber}(1/2) & \text{with probability } p \\
x_i & \text{otherwise}
\end{cases}
\]

Let \( y_i \) be the message sent by user \( i \). Due to subsampling and noise addition, the expected value of \( \sum y_i \) is \((1 - p) \cdot \sum x_i + np/2\). The analyzer will re-center and re-scale to obtain an unbiased estimator:

\[
A_{RR}(\bar{y}) := \frac{1}{1 - p} \left( \sum y_i - np/2 \right)
\]

\[
\mathbb{E}[A_{RR}(\bar{y})] = \frac{1}{1 - p} \cdot \left( \mathbb{E}[\sum y_i] - np/2 \right)
\]

\[= \sum x_i\]

Setting \( p \leftarrow 2/(e^\epsilon + 1) \) suffices for \( \epsilon \)-local privacy but incurs an additive error of \( O(\frac{1}{\epsilon} \sqrt{n}) \). This is optimal.

**Theorem 9** (Beimel et al. [11] & Chan et al. [14]). Let \( P \) be a \( \epsilon \)-locally private protocol. If \( P \) computes binary sums up to additive error \( \alpha \) with constant probability, then \( \alpha = \Omega(\frac{1}{\epsilon} \sqrt{n}) \).

Note that \( P_{RR} := (R_{RR}, A_{RR}) \) can also be interpreted as a one-message shuffle protocol. Cheu et al. [16] show that the parameter \( p \) can be chosen such that RR achieves \((\epsilon, \delta)\) shuffle privacy while also avoiding error that scales polynomially with \( n \).

**Theorem 10** (Cheu et al. [16]). Randomized response \( P_{RR} = (R_{RR}, A_{RR}) \) is a one-message protocol that is \((\epsilon, 3\delta', \gamma)\)-robustly shuffle private for any \( \gamma \in (0, 1) \). It computes binary sums up to additive error \( O(\frac{1}{\epsilon} \sqrt{\log \frac{1}{\delta'}}) \) with constant probability.

**Proof.** If \( n > 80 \cdot (\frac{e^{-1} + 1}{e^{-1} - 1})^2 \ln \frac{1}{\delta'} \), we set \( p \) to \( \frac{40 \cdot (\frac{e^{-1} + 1}{e^{-1} - 1})^2 \ln \frac{1}{\delta'}}{n} \). If \( n \) is smaller, \( p \) must take a different form and the analysis will naturally change; we omit this technicality for neatness (refer to [16] for more details).

We begin with the robust privacy analysis. The shuffler’s output is \( \bar{y} \leftarrow (S \circ R_{RR}^\gamma) (\vec{x}) \), a vector of bits whose order is chosen uniformly at random. Observe that the permutation does not contain information about the data: given oracle access to the sum \( \Sigma y_i \), a privacy adversary can exactly simulate the distribution of \( \bar{y} \). Hence it suffices to ensure that \( \Sigma y_i \) is differentially private.

Let \( H \) denote the set of users who report messages \( y_i \) from \( \text{Ber}(1/2) \). Notice that \( \sum y_i = \sum_{i \in H} x_i + \sum_{i \notin H} y_i \) so that \( \sum y_i \sim \sum_{i \in H} x_i + \text{Bin}(|H|, 1/2) \). If \(|H| \geq 20 \cdot (\frac{e^{-1} + 1}{e^{-1} - 1})^2 \ln \frac{1}{\gamma} \), then we have \((\epsilon, 2\delta')\) privacy by the binomial mechanism (Lemma 4).

We will argue that this bound on \(|H|\) holds with probability \( 1 - \delta' \) for our choice of \( p \), so we have \((\epsilon, 3\delta')\) privacy overall. The expected value of \(|H|\) is \( \mathbb{E}[|H|] = \gamma n \cdot p = 40 \cdot (\frac{e^{-1} + 1}{e^{-1} - 1})^2 \ln \frac{1}{\gamma} \) which is sufficiently large to invoke a Chernoff bound: \(|H| \geq \frac{1}{2} \cdot \mathbb{E}[|H|] = 20 \cdot (\frac{e^{-1} + 1}{e^{-1} - 1})^2 \ln \frac{1}{\gamma} \) with probability \( \geq 1 - \delta' \).

We conclude by bounding the error, assuming all users obey the protocol (\( \gamma = 1 \)). Recall that the output of the protocol is \( \frac{1}{1 - p} (\sum y_i - np/2) \). By a Chernoff bound, we have that \( \sum y_i - np/2 \) is within \( O(\frac{1}{\epsilon} \sqrt{\log \frac{1}{\delta'}}) \) of its expectation. And because \( \frac{1}{1 - p} < 2 \), the error of the unbiased estimator is \( O(\frac{1}{\epsilon} \sqrt{\log \frac{1}{\delta'}}) \). \( \square \)

We remark that there are other shuffle protocols for binary sums with low error; Table 1 presents their most salient features. We also note that the way we proved robust shuffle privacy for RR can be repeated with some modification for SYM2 and ZSUM. This is because they rely upon the binomial mechanism in a similar fashion.
Table 1: Shuffle protocols for binary sums. Each message is one bit. The monikers “SYM2” and “PURE” are original to this survey. “w.c.p.” denotes a bound that holds with constant probability over the randomness of all users.

<table>
<thead>
<tr>
<th>Source</th>
<th>Error w.c.p.</th>
<th>No. Messages per User</th>
<th>Advantage over RR</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>RR</td>
<td>$O(\frac{1}{\sqrt{2\pi n}} \cdot \sqrt{\log \frac{1}{\delta}})$</td>
<td>1</td>
<td>—</td>
<td>[16]</td>
</tr>
<tr>
<td>PURE</td>
<td>$O(\frac{1}{\sqrt{2\pi n}} \cdot \sqrt{\log \frac{1}{\delta}})$</td>
<td>$O(\frac{1}{e} \log u)$</td>
<td>$\delta = 0$</td>
<td>[21]</td>
</tr>
<tr>
<td>SYM</td>
<td>$O(\frac{1}{\sqrt{2\pi n}} \cdot \sqrt{\log \frac{1}{\delta}})$</td>
<td>$O(\frac{1}{e} \log \frac{1}{\delta})$ w.c.p.</td>
<td>$\delta = 0$, noise symmetry</td>
<td>[6]</td>
</tr>
<tr>
<td>SYM2</td>
<td>$O(\frac{1}{\sqrt{2\pi n}} \cdot \sqrt{\log \frac{1}{\delta}})$</td>
<td>$O(\frac{1}{e} \log \frac{1}{\delta})$ w.c.p.</td>
<td>noise symmetry</td>
<td>[6]</td>
</tr>
<tr>
<td>ZSUM</td>
<td>$O(\frac{1}{\sqrt{2\pi n}} \cdot \sqrt{\log \frac{1}{\delta}})$</td>
<td>2</td>
<td>If sum is 0, estimate is 0</td>
<td>[5]</td>
</tr>
</tbody>
</table>

3.2 Histograms

In this setting, each user has one value in the set $[k]$. The objective is to privately estimate the count of each element $j$. To be precise, the output must be a vector $(\hat{c}_1, \ldots, \hat{c}_k)$ minimizing $\ell_\infty$ error with $(c_1, \ldots, c_k)$ where $c_j$ is the count of $j$ in the input dataset. This must grow with $k$ under local privacy:

**Theorem 11** (Bassily & Smith [10]). Let $P$ be an $(\epsilon, \delta)$-locally private protocol. If $P$ reports a histogram that has $\ell_\infty$ error $a$ with constant probability, then $a = \Omega(\frac{1}{\epsilon} \sqrt{n \log k})$.

In contrast, it is possible to have error independent of $k$ under robust shuffle privacy:

**Theorem 12** (Balcer & Cheu [5]). There is a $2k$-message histogram protocol that satisfies $(2\epsilon, 4\delta^2, \gamma)$-robust shuffle privacy for any $\gamma \in (0, 1]$. Its estimate has $\ell_\infty$ error $O(\frac{1}{e} \log \frac{1}{\delta})$ with constant probability.

**Proof.** A simple way to obtain a private histogram is to count each of the $k$ elements with a binary sum protocol, then perform a union bound. The $k$ protocol executions can be done in parallel by having each user label their messages. To be precise, let $R_j$ be a binary sum randomizer that counts the occurrences of $j$. If $R_j(x_i)$ outputs messages $a$ and $b$, user $i$ reports the labeled messages $(j, a)$ and $(j, b)$. The analyzer can feed the messages for counting $j$ into the corresponding function $A_j$.

A side-effect of this reduction approach is that the union bound may create a dependence on $k$. For example, if we use RR for frequency estimation, the $\ell_\infty$ error after the union bound has a $\sqrt{\log k}$ term. But we can avoid this dependence by using ZSUM, a binary sum protocol which guarantees noiseless estimation when the input is $(0, \ldots, 0)$. As such, the elements with nonzero frequency will be the only ones with noisy estimates, so the union bound is over $\leq n$ executions instead of $k$.

We will sketch the protocol ZSUM for target privacy parameters $(\epsilon, 2\delta^2)$ so that simple composition will give our overall privacy guarantee. We begin with the two-message randomizer, which takes parameter $r \in (0, 1)$.

$$R_{\text{ZSUM}}(x_i) := (x_i, \text{Ber}(r))$$

As with RR, the number and order of messages produced by $(S \circ R^n_{\text{ZSUM}})$ is data independent, so it suffices to prove privacy of the message sum $\sum y_{i,1} + y_{i,2}$. But this quantity is exactly $\sum x_i + \eta$, where $\eta$ is drawn from the distribution $\text{Bin}(n, r)$. By Lemma 4, it suffices to choose $r = 1 - \frac{10}{\pi} \left(\frac{\epsilon \eta}{\sqrt{2} - 1}\right)^2 \ln \frac{1}{\delta}$ for $(\epsilon, 2\delta^2)$ privacy.

Now we define the analyzer $A_{\text{ZSUM}}$.

$$A_{\text{ZSUM}}(\vec{y}) := \begin{cases} 0 & \text{if } \sum y_{i,1} + y_{i,2} \leq n \\ \sum y_{i,1} + y_{i,2} - nr & \text{otherwise} \end{cases}$$
First consider the case where \( \sum x_i = 0 \). Because \( \eta \sim \text{Bin}(n, r) \) has maximum value \( n \), \( P[\sum y_{i,1} + y_{i,2} \leq n] = 1 \) so there is zero error.

Now consider the case where \( \sum x_i = 0 \). We can use a Chernoff bound to argue that \( |\eta - nr| = O(\sqrt{n(1 - r) \log n}) \) with probability \( 1/10n \). If we do not truncate, subtracting \( nr \) removes bias so that error has magnitude \( O(\sqrt{n(1 - r) \log n}) = O(\frac{1}{r} \log \frac{1}{r}) \). Otherwise, error is exactly \( \sum x_i \). But truncation will not occur when \( \sum x_i = \Omega(\frac{1}{\alpha^2} \log \frac{1}{\alpha}) \); in this case, \( \sum x_i + \eta > \sum x_i + n - O(\frac{1}{\alpha^2} \log \frac{1}{\alpha}) \) so that \( \sum y_{i,1} + y_{i,2} > n \).

When \( n < 10 \cdot \left(\frac{c + 1}{c - 1}\right)^2 \cdot \ln \frac{1}{\delta} \), our choice of \( r \) is not a valid probability. In this case, each user can deterministically send the bits 0, 0 to ensure \( (0, 0) \)-privacy. \( A_{\text{ZSUM}} \) will report 0 with probability 1, which has error \( \sum x_i \leq n = O(\frac{1}{\alpha^2} \log \frac{1}{\alpha}) \).

### 3.3 Uniformity testing

The results so far concern protocols that make no assumption about where user data comes from. We now transition to problems where data is drawn i.i.d. from some probability distribution \( D \) over \([k]\). In the case of \( \alpha \)-uniformity testing, the objective is to report “uniform” with probability 2/3 when \( D = U \) and “not uniform” with probability 2/3 when \( |D - U|_{TV} > \alpha \). The minimum number of users needed to ensure that these conditions hold is the sample complexity of the protocol. Under local privacy, this must scale at least linearly with \( k \).

**Theorem 13** (Acharya et al. [1]). If an \( \epsilon \)-locally private protocol performs \( \alpha \)-uniformity testing, then its sample complexity is \( \Omega(k/\alpha^2 \epsilon^2) \).

But under robust shuffle privacy, the sample complexity is smaller by at least a polynomial factor.

**Theorem 14** (Balcer et al. [6]). There is a multi-message protocol that satisfies \((\epsilon/\gamma, 0, \gamma)\)-robust shuffle privacy for any \( \gamma \in (0, 1) \) and solves \( \alpha \)-uniformity testing with sample complexity \( O\left(\frac{k^{2/3}}{\alpha^{2/3} \epsilon \gamma^2} + \frac{k^{2/3}}{\alpha \epsilon^2} \right) \sqrt{\log k} \).

**Proof.** We note that we will take \( n \sim \text{Pois}(m) \) and upper bound \( m \). This “Poissonization” has the effect of making the random variables \( c_1, \ldots, c_k \) mutually independent, which simplifies the analysis.

Cai et al. [13] give a recipe for private uniformity testing under Poissonization. First, compute a private histogram \( (c_1, \ldots, c_k) \). Then, compute the test statistic

\[
Z'(c_1, \ldots, c_k) := \frac{k}{m} \sum_{j=1}^{k} (\tilde{c}_j - m/k)^2 - \tilde{c}_j
\]

The final step is to prove that this statistic is small when the data distribution is uniform but large when it is \( \alpha \)-far from uniform, which means we can distinguish the two cases with a threshold test.

Amin et al. [4] give the following procedure to analyze \( Z' \). If we let \( \eta_j \) be the noise in \( \tilde{c}_j \) introduced by privacy, then we rewrite \( Z' \) as

\[
(1) = \frac{k}{m} \sum_{j=1}^{k} (c_j + \eta_j - m/k)^2 - c_j - \eta_j
\]

\[
= \frac{k}{m} \sum_{j=1}^{k} (c_j - m/k)^2 - c_j + \sum_{j=1}^{k} \eta_j^2 + 2k \sum_{j=1}^{k} \eta_j \cdot (c_j - m/k) - \frac{k}{m} \sum_{j=1}^{k} \eta_j
\]

Analysis in Acharya et al. [2] imply bounds on term \( Z \) in the two relevant cases: there is a constant \( t \) and a function \( f(\alpha, m) \) such that

\[
(1) \leq Z + A + B + C
\]
• when \( \|D - U\|_{TV} > a, Z > t \cdot f(a, m) \) with constant probability
• when \( D = U, Z \leq f(a, m) \) with constant probability

It will suffice to prove the two statements below:

(i) when \( \|D - U\|_{TV} > a, A + B + C > 0 \) with constant probability
(ii) when \( D = U, A + B + C < (t - 1) \cdot f(a, m) \) with constant probability

In Balcer et al. [6], each \( \eta_j \) is an independent sample from zero-mean symmetric noise. This comes from \( k \) independent executions of the binary sum protocol \( SYM \) (see Sec. 3.2). (i) follows from this symmetry. (ii) follows from Chebyshev's inequality and the moments of \( \eta_j \); at a high level, a user in \( SYM \) contributes independent \( SG(\epsilon) \) noise with probability \( \approx 1/e^2n \) which yields a mean of 0 and a variance of \( O(1/e^4) \). We omit the details for brevity.

\[ \square \]

3.4 Pointer-Chasing

The pointer chasing problem is denoted \( PC(d, \ell) \) where \( d, \ell \in \mathbb{N} \). A problem instance is a set \( \{(1, \vec{a}), (2, \vec{b})\} \), where \( \vec{a} \) and \( \vec{b} \) are permutations of \( [\ell] \). A protocol solves \( PC(d, \ell) \) with sample complexity \( n \) if, given \( n \) independent samples drawn uniformly with replacement from any problem instance \( \{(1, \vec{a}), (2, \vec{b})\} \), it outputs the \( d \)-th integer in the sequence \( a_1, b_{a_1}, a_{b_{a_1}}, \ldots \) with constant probability.

The sample complexity of \( PC(d, \ell) \) under local privacy must scale at least linearly with \( \ell \).

**Theorem 15** (Joseph et al. [27]). If an \((\epsilon, \delta)\)-locally private protocol solves \( PC(2, \ell) \) with sample complexity \( n \) then \( n = \Omega(\ell) \).

In stark contrast, the sample complexity under shuffle privacy is independent of \( \ell \):

**Theorem 16** (Balcer & Cheu [5]). There is a \( 8 \cdot (\ell!)^2 \)-message protocol that satisfies \( (2\epsilon, 4\delta, \gamma) \)-robust shuffle privacy for any \( \gamma \in (0, 1) \) and solves \( PC(2, \ell) \) with sample complexity \( O(\frac{1}{\ell^2} \log \frac{1}{\delta}) \).

**Proof.** Let \( \pi(\ell) \) denote all permutations of \( [\ell] \). Observe that the tuples \( (1, \vec{a}), (2, \vec{b}) \) are elements of the universe \( (1, 2) \times \pi(\ell) \) which has size \( 2 \cdot \ell! \). We can solve the problem once we have a protocol that singles out \( (1, \vec{a}) \) and \( (2, \vec{b}) \) from the universe with constant probability.

Balcer & Cheu argue that the task of privately identifying \( (1, \vec{a}) \) and \( (2, \vec{b}) \) with constant probability is \( O(\frac{1}{\ell^2} \log \frac{1}{\delta}) \). By a straightforward concentration argument, it suffices to have \( O(t) \) samples to ensure \( (1, \vec{a}) \) and \( (2, \vec{b}) \) each appear \( \geq t + 1 \) times with constant probability. Taking universe size \( k = 4 \cdot (\ell!)^2 \), we then use the histogram protocol built atop \( ZSUM \) (Theorem 12). When \( t = \Omega(\frac{1}{\ell^2} \log \frac{1}{\delta}) \), it will report nonzero frequencies for \( (1, \vec{a}) \) and \( (2, \vec{b}) \) but zero for every other element in the universe.

\[ \square \]

4 Separations between Central & Shuffle Privacy

There are results that separate the shuffle model from the central model, but they do not hold for general protocols. Instead, they leverage some structural constraint.

4.1 Single-message Protocols

The first class of lower bounds hold for protocols wherein each user sends exactly one message with probability 1. We begin with a negative result for bounded-value sums. Here, a protocol must privately estimate the sum of values in the interval \([0, 1]\).
Theorem 17 (Balle et al. [9]). If a single-message shuffle protocol satisfies \((\varepsilon, \delta)\) differential privacy for \(n\) users and computes bounded-value sums, then the mean-squared error must be \(\Omega(n^{1/3})\).

In the desirable case of unbiased estimators, this implies an additive error of \(\Omega(n^{1/6})\). But in the central model, we can simply use the Laplace mechanism to achieve an error of \(O(1/\varepsilon)\).

The techniques used to prove the above are specific to bounded-value sums. A more general technique is to consider what shuffle privacy entails for an arbitrary single-message randomizer.

Lemma 18 (Balcer & Cheu [5]). If a single-message protocol \(P = (R, A)\) satisfies pure shuffle privacy, then removing the shuffler leaves behind a pure locally private protocol. Specifically, \(R\) must satisfy \(\varepsilon\)-differential privacy on its own whenever \(P\) is \(\varepsilon\)-shuffle private.

This means that under pure differential privacy, the single-message shuffle model is exactly equivalent to the local model. So all separations between the central and local models hold here as well.

But it is clear from RR that this exact equivalence does not hold for approximate shuffle privacy. The following lemma accommodates the relaxation.

Lemma 19 (Cheu et al. [16]). If a single-message protocol \(P = (R, A)\) satisfies \((\varepsilon, \delta)\)-shuffle privacy for \(n\) users, then \(R\) must satisfy \((\varepsilon + \ln n, \delta)\)-differential privacy on its own.

Thus, we can invoke any local model lower bound that holds for \((\varepsilon + \ln n, \delta)\) privacy. This recipe gives the following lower bound.

Theorem 20 (Ghazi et al. [22]). Any single-message protocol that satisfies \((1, o(1/n))\)-shuffle privacy and outputs histograms with \(\ell_\infty\) error \(n/10\) must have \(n = \Omega(\frac{\log k}{\log \log k})\).

In contrast, there is a central model algorithm where \(n = O(1)\) suffices for the same privacy and accuracy regimes.

4.2 Robustly Shuffle Private Protocols

The second class of lower bound applies to robustly shuffle private protocols. These lower bounds exploit reductions to pan-privacy. A pan-private algorithm is an online algorithm such that, for any time \(t\), the joint distribution of the internal state at time \(t\) and the final output is differentially private.

We first consider the \(\alpha\)-distinct elements problem. The objective is to estimate—up to additive error \(\alpha\)—the number of elements in \([k]\) that appear in \(\vec{x}\) at least once. Balcer et al. [6] show that a robustly shuffle private protocol implies a pan-private algorithm and then invoke lower bounds from the pan-privacy literature.

Lemma 21 (Balcer et al. [6]). If an \((\varepsilon, \delta, 1/3)\)-robustly shuffle private protocol solves \(\alpha\)-distinct elements, then there is an \((\varepsilon, \delta)\)-pan-private algorithm that solves \(\alpha + 1\)-distinct elements.

Theorem 22 (Mir et al. [28] and Balcer et al. [6]). If an \((\varepsilon, \delta)\)-pan-private algorithm solves \(\alpha\)-distinct elements, then \(\alpha = \Omega(\sqrt{k/\varepsilon})\).

Theorem 23 (Balcer et al. [6]). If an \((\varepsilon, \delta, 1/3)\)-robustly shuffle private protocol solves \(\alpha\)-distinct elements, then \(\alpha = \Omega(\sqrt{k/\varepsilon})\).

In contrast, the binomial and geometric mechanisms (Lemmas 4 and 5) imply that the error in the central model is independent of \(k\).

Proof Sketch of Lemma 21. The pan-private algorithm will first generate \(n/3\) copies of 1 and execute \((S \circ R^{n/3})\) on that vector. Then the algorithm will read user datum \(x_1\) from the stream, execute \(R(x_1)\), and
add the messages to the set. The updates repeat until data from all \( n/3 \) users are read. The algorithm concludes by passing in \( n/3 \) more copies of 1 and then executing \( A \) on all messages.

The number of distinct elements is increased by at most 1, so the error is at most \( \alpha + 1 \). Pan-privacy is ensured because the internal state at any point is a post-processing of \( (S \circ R^u)^{\omega/3} \) which is differentially private by robustness.

We can obtain a lower bound for uniformity testing by using essentially identical steps. This time, the pan-private lower bound comes from Amin et al. [4].

**Theorem 24** (Balcer et al. [6]). If an \((\varepsilon, 0, 1/3)\)-robustly shuffle private protocol solves \( \alpha \)-uniformity testing, then its sample complexity is
\[
n = \Omega\left(\frac{k^{2/3}}{\varepsilon^{2/3}d} + \frac{\sqrt{k}}{\alpha} + \frac{1}{\varepsilon^2}\right).
\]

In contrast, Acharya et al. [3] show that the sample complexity in the central model scales with \( \sqrt{k} \).

5 **Open Questions**

**How robust are single-message shuffle protocols?** If all \( n-1 \) of Alice’s peers drop out of a \((\varepsilon, \delta)\)-shuffle private protocol, then she is left with \((\varepsilon'(n), \delta'(n))\) privacy, where we currently know that \( \varepsilon'(n) \leq \varepsilon + \ln n \) and \( \delta'(n) \leq \delta \) (Lemma 19). It is conceivable that these bounds can be improved. On the other hand, we may tighten the amplification-by-shuffling lemmas of Erlingsson et al. [20] and Balle et al. [9]. These lemmas state that there are functions \( \text{amp}_1, \text{amp}_2 \) such that an \((\varepsilon, \delta)\)-locally private protocol implies an \((\text{amp}_1(\varepsilon, \delta, n), \text{amp}_2(\varepsilon, \delta, n))\)-shuffle private protocol. So when \( \gamma n - 1 \) of Alice’s peers are honest, she has \((\text{amp}_1(\varepsilon'(n), \delta'(n), \gamma n), \text{amp}_2(\varepsilon'(n), \delta'(n), \gamma n))\) privacy.

**How robust are multi-message shuffle protocols?** The fact that each user is performing the same randomization algorithm implies an equal division of labor. Intuitively, this should mean that drop-outs cause a graceful degradation of privacy. Unfortunately, we cannot extend the arguments for single-message protocols. For example, there exists a multi-message randomizer \( R \) where \((S \circ R^n)\) does not satisfy any level of pure differential privacy and yet \((S \circ R^1)\) does, for some large but finite \( \varepsilon \) parameter.

**Are there other techniques to obtain lower bounds for robust shuffle privacy?** The current technique is bottlenecked by pan-privacy. For example, there is no known lower bound for uniformity testing under approximate pan-privacy.

**References**


A Communication-efficient Protocols

A.1 Bounded-value Sums

In this setting, users have values in the interval \([0, 1]\) and the objective is to privately compute their sum. When each user is willing to send multiple messages, results from Balle et al. \([7, 8]\) and Ghazi et al. \([24, 23]\) show that it is possible to simulate the geometric mechanism in the shuffle model.
**Theorem 25.** There is an \((\epsilon, \delta)\)-shuffle private protocol for bounded-value sums with error \(O(\frac{1}{\epsilon^2})\) where each user sends \(O(\frac{\log(1/\delta)}{\log(n)})\) messages of \(O(\log n)\) bits.

**Proof Sketch.** The first step is to equate a sample from \(SG(\epsilon)\) with the sum of \(n\) samples from another distribution \(D_\epsilon\). Then we recall a modular arithmetic protocol \(P_{\text{MOD}}\) by Ishai et al. [26] which has the following property: two datasets with the same sum induce two distributions of \((S \circ R_{\text{MOD}}^n)\) that are \(\delta\)-close in statistical distance. Now define \(R\) to be the execution of \(R_{\text{MOD}}\) on \(y_i \leftarrow x_i + \eta\) where \(\eta \sim D_\epsilon\).

If an adversary can only recover \(\sum y_i\), then we will have \(\epsilon\)-differential privacy. And due to the way we use \(\text{MOD}\), the output of the shuffler \((S \circ R_{\text{MOD}})^n(x_1, \ldots, x_n)\) is \(\delta\)-close to \((S \circ R_{\text{MOD}})^n(\sum y_i, 0, \ldots, 0)\). This is enough to ensure approximate differential privacy. The error bound \(O(\frac{1}{\epsilon^2})\) follows from the fact that we are simulating the geometric mechanism (Lemma 5), as well as the fact that sums exceed the modulus with very low probability.

Refer to Balle et al. [8] and Ghazi et al. [23] for analyses of the message complexity of \(\text{MOD}\).

---

**A.2 Histograms & Range queries**

Unlike Balcer & Cheu [5], Ghazi et al [22] strive to minimize communication complexity of histograms rather than error. This is achieved with the count-min and Hadamard response techniques that found success in the local model. Refer to Table 2 for a summary of their results. [22] also explain how to use their protocols in a black-box way to solve the range-query problem. In this setting, data is drawn from \([k]^d\) and the objective is to estimate the number of points in a given rectangle. Refer to Table 3 a summary of the results.

<table>
<thead>
<tr>
<th>Technique</th>
<th>(\ell_\infty) Error w.c.p.</th>
<th>No. Messages per User</th>
<th>Bits per Message</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>(k) parallel counts</td>
<td>(O\left(\frac{1}{\epsilon^2}\log \frac{1}{\delta}\right))</td>
<td>(O(k))</td>
<td>(O(\log k))</td>
<td>[5]</td>
</tr>
<tr>
<td>Count-Min</td>
<td>(\tilde{O}\left(\frac{1}{\epsilon^2}\cdot \log^3 k \cdot \log \frac{1}{\delta}\right))</td>
<td>(\tilde{O}\left(\frac{1}{\epsilon^2}\log^3 k \cdot \log \frac{1}{\delta}\right)) w.c.p.</td>
<td>(O(\log n + \log k))</td>
<td></td>
</tr>
<tr>
<td>Hadamard</td>
<td>(O\left(\log k + \frac{1}{\epsilon^2}\cdot \sqrt{\log k \cdot \log \frac{1}{\epsilon^2}}\right))</td>
<td>(O\left(\frac{1}{\epsilon^2}\log \frac{1}{\epsilon^2}\right))</td>
<td>(O(\log n \log k))</td>
<td>[22]</td>
</tr>
</tbody>
</table>

Table 3: Shuffle protocols for range queries. All take \(\delta > 0\). We assume \(\delta < 1/\log k\) for results from [22] and \(\delta \geq e^{-O(n \epsilon^2)}\) for the result from [5]. The notation \(\tilde{O}(\ldots)\) suppresses nested logarithms.

<table>
<thead>
<tr>
<th>Technique</th>
<th>Error</th>
<th>Messages per User</th>
<th>Bits per Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count-Min</td>
<td>(O\left(\frac{1}{\epsilon^2}\log^2 d + \frac{3}{2}(k^d) \log \frac{1}{\delta}\right))</td>
<td>(O\left(\frac{1}{\epsilon^2}\log^3 d + 3(k^d) \log \frac{1}{\delta}\right)) w.c.p.</td>
<td>(O(\log n + \log(d \log k)))</td>
</tr>
<tr>
<td>Hadamard</td>
<td>(O\left(\frac{1}{\epsilon^2}\log^{2d+1/2} (k^d) \log \frac{1}{\delta^2}\right))</td>
<td>(O\left(\frac{1}{\epsilon^2}\log^{2d} (k^d) \log \frac{1}{\delta^2}\right))</td>
<td>(O(\log(n) \cdot d \log k))</td>
</tr>
</tbody>
</table>

---

**A.3 Lower Bounds**

We close this section with a result by Ghazi et al. [21] which states that every communication-bounded shuffle protocol must imply some local protocol with a nontrivial privacy guarantee:
Lemma 26 (Ghazi et al. [21]). Suppose \( \mathcal{P} = (\mathcal{R}, A) \) satisfies \( O(1) \)-shuffle privacy and each user sends \( m \) messages of \( \ell \) bits. Then the local randomizer \( (S \circ R_1) \) satisfies \( (0, 1 - 2^{-O(m^2 \ell)}) \)-differential privacy.

By way of the local model, this implies a lower bound for binary sums:

Corollary 27 (Ghazi et al. [21]). If an \( m \)-message shuffle protocol satisfies \( O(1) \)-differential privacy and computes binary sums up to error \( o(\sqrt{n}) \), then \( m^2 \ell = \Omega(\log n) \).