CS3 325: Private Data Analysis **Fall 2022**

Lecture 9: Local and Shuffle Differential Privacy

Lecturer: Di Wang *Scribes: Di Wang*

Note: *LaTeX template courtesy of UC Berkeley EECS dept.*

9.1 Local Differential Privacy

For most of this course, we have focused on the central model of differential privacy. In this model, there is a trusted curator, who may view all the user data in the clear without any privacy concern. We require that anything that the curator outputs based on the data is appropriately privatized via differential privacy. In these cases, each individual should ensure that their own disclosures are differentially private. In some sense, the trust barrier is moved closer to the user. While this has a benefit of providing a stronger privacy guarantee, it also comes at a cost in terms of accuracy.

In the local model of differential privacy, each individual user randomizes her data herself using a randomizer Q_i to obtain a report z_i which she sends to an untrusted server to be aggregated in to a summary s that can be used to answer queries about the data. The server may provide public coins visible to all parties, but privacy guarantees depend only on the randomness of the users local coins. The local model has been studied extensively because control of private data remains in users hands.

Definition 9.1 ([1]) *We say that an algorithm $Q : \mathcal{X} \mapsto \mathcal{Z}$ is (ϵ, δ)-Locally Differnetially Private (LDP) if for any pairs $x, x' \in \mathcal{X}$ and any (measurable) subset $S \subseteq Z$ we have*

$$\mathbb{P}(Q(x) \in S) \leq e^{\epsilon}\mathbb{P}(Q(x') \in S) + \delta.$$

The special case with $\delta = 0$ is called pure ϵ-LDP.

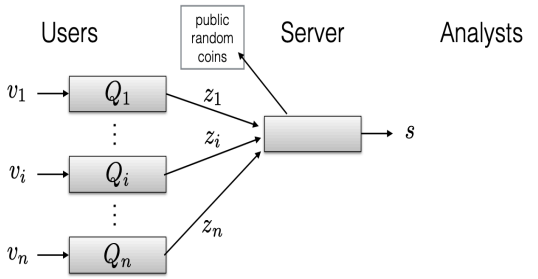

Figure 9.1: Local Differential Privacy Model

Unlike Differential Privacy, there are many interesting technical properties and questions involving local differential privacy which have been studied in recent years. Here we list some of its directions:

1. **The role of interactivity in this setting.** In the non-interactive model, there is a single round in which each user sends (privatized) messages to the curator. In the sequentially interactive model, based on messages sent by users, the curator can ask new questions to other users, but each user may only send messages one time. Finally, in the fully interactive model, there are allowed to be many back and forth interactions between the users and the curator. There are known separations between all these models, and understanding the full power of each for various problems is still a mystery which needs to be understood.

2. **Separation between pure and approximate differential privacy.** There is currently no known separation between pure and approximate differential privacy in the local model. Indeed, it can be formalized that they are equivalent in certain settings (non-interactive LDP model).

3. **Symmetric or Asymmetric Mechanisms.** In the most of the work we assume all the mechanisms Q_i are the same. For example, we can assume Q_i is the Gaussian mechanism. However, it has been showed that for some statistical estimation problem, we can get improved bounds if we use asymmetric mechanisms, *i.e.,* Q_i could be different.

4. **Public or Private Random Coin.** Whether there are public coins or private coins is a very interesting question in LDP. For example, it has been showed that when we use public coin protocols, every ϵ-LDP algorithm can be transformed so that each user sends a single bit to the server. Moreover, the transformation is efficient under the assumption that one can efficiently compute conditional probability $Q(z|x)$ for the randomizer in the protocol. However, it has been showed that for some statistical estimation problem, we can get improved bounds if we use private coins.

All of these technical questions are fascinating, but unfortunately we do not have time to discuss them during this course. In this lecture, we just focus on the average query in the ϵ-LDP model. And we do not consider the public or private random coin and we only focus on the symmetric mechanisms in the non-interactive LDP protocol.

Before that, let us first look at the average of $X_i \in \{0, 1\}$. In the central model of differential privacy, we would use the Laplace mechanism:

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} X_i + \text{Lap}(\frac{1}{n\epsilon}).$$

The noise introduced due to privacy is of magnitude $\frac{1}{n\epsilon}$. Thus we have

$$|\hat{\mu} - \mu| \leq \frac{1}{n\epsilon}. \tag{9.1}$$

That is, to achieve the error of α, the number of samples need $n \geq O(\frac{1}{\alpha\epsilon})$.

On the other hand, let us consider randomized response, where user i outputs a bit Y_i, which is equal to X_i with probability $\frac{e^\epsilon}{e^\epsilon+1}$, and $1 - X_i$ with probability $\frac{1}{1+e^\epsilon}$. It is easy to see that the users disclosure is ϵ-DP, just by taking the ratio of these probabilities. The curator (no longer trusted) can aggregate these responses as follows:

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n}(\frac{e^\epsilon + 1}{e^\epsilon - 1}Y_i - \frac{1}{e^\epsilon - 1}).$$

In the previous lectures we showed that the variance of $\hat{\mu}$ is $O(\frac{1}{n\epsilon^2})$, thus the error will be $O(\frac{1}{\sqrt{n}\epsilon})$. That is, to achieve the error of α, the number of samples need $n \geq O(\frac{1}{\alpha^2\epsilon^2})$.

We can already see a dramatic difference in these two sample bounds. The algorithm in the local setting requires quadratically more data than the algorithm in the central setting. Unfortunately, this is intrinsic: it is possible to show that both of these sample complexity bounds are tight up to constant factors. This

translates into significantly worse accuracy in practice: typically, most deployments require much more data in order to achieve non-trivial accuracy guarantees in the local model. Indeed, this is the reason why all the deployments at companies have access to massive datasets.

Theorem 9.2 ([3]) *For every 1-LDP algorithm A, there is a dataset $D = \{x_1, x_2, \cdots, x_n\} \in \{0,1\}^n$, such that the error of $A(D)$ to the average $\frac{1}{n}\sum_{i=1}^n x_i$ is $\Omega(\frac{1}{\sqrt{n}})$ with high probability.*

Proof: Consider a uniformly random input dataset $D \in \{0,1\}^n$, let $R = (R_1, R_2, \cdots, R_n)$ denote the randomness of the n parties, and let $A = A(D, R)$ be the random variable denoting the transcript of the protocol. Let $a \in \text{supp}(A)$ be any out of A. We claim that conditioned on $A = a$:

1. The n random variables $(X_1, R_1), \cdots, (X_n, R_n)$ are independent, and in particular $X_1, \cdots, x_n$ are independent.
2. Each $\mathbb{P}(X_i = 1) \in [\frac{1}{4}, \frac{3}{4}]$.

Item 1 is a general fact about interactive protocols if the parties inputs start independent, they remain independent conditioned on the transcript and can be proven by induction on the number of rounds of the protocol. Item 2 uses 1-LDP and and Bayes Rule:

$$\frac{\mathbb{P}[X_i = 1|A = a]}{\mathbb{P}[X_i = 0|A = a]} = \frac{\mathbb{P}[A = a|X_i = 1]\mathbb{P}[X_i = 1]}{\mathbb{P}[A = a|X_i = 0]\mathbb{P}[X_i = 0]} = \frac{\mathbb{P}[A = a|X_i = 1]}{\mathbb{P}[A = a|X_i = 0]} \in [e^{-\epsilon}, e^{\epsilon}].$$

This implies that

$$\mathbb{P}[A = a|X_i = 1] \in [\frac{1}{e^{\epsilon} + 1}, \frac{e^{\epsilon}}{e^{\epsilon} + 1}].$$

Consequently, conditioned on $A = a$, $\frac{1}{n}\sum_{i=1}^n X_i$ is the average of n independent $\{0,1\}$ random variables with bounded bias. In particular, the standard deviation of $\frac{1}{n}\sum_{i=1}^n X_i$ is $\Omega(\frac{1}{\sqrt{n}})$, and by anti-concentration bounds, with high probability we will have

$$|\frac{1}{n}\sum_{i=1}^n X_i - a| = \Omega(\frac{1}{\sqrt{n}})$$

Since the protocol has error $\Omega(\frac{1}{\sqrt{n}})$ on a random dataset with high probability, there is some fixed dataset on which it has error $\Omega(\frac{1}{\sqrt{n}})$ with high probability. ■

Now we focus on the general average query where each $x_i \in [0, m]$. While we have not discussed it yet in the local setting, the Laplace mechanism is still applicable in this setting. Specifically, if a user has a point $x_i \in [0, m]$, they can transmit x_i plus Laplace noise proportional to $\frac{m}{\epsilon}$, where m is the sensitivity of each users datapoint. The curator then averages these results to obtain an estimate of the mean:

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^n (X_i + \text{Lap}(\frac{m}{\epsilon}).$$

Thus we have $|\hat{\mu} - \mu| \leq O(\frac{m}{\sqrt{n}\epsilon})$. Actually, this bound is already tight. However, the downside is that is somewhat wasteful when it comes to the amount of communication most values transmitted by users will be of order $\Omega(m)$ (due to the magnitude of the noise), thus requiring $\Omega(\log m)$ bits of communication. We now present another algorithm, which is proposed by [7]. In this paper, the authors provide an algorithm for the same problem which requires only one bit of communication per user.

Each user takes their value x_i, and transmits a message Y_i which is equal to 1 with probability $\frac{1}{e^\epsilon+1} + \frac{x_i}{m}\frac{e^\epsilon-1}{e^\epsilon+1}$, and 0 otherwise. Observe that this only requires a user to commit a single bit, rather than the $\Omega(\log m)$ bits before. The curator then compute the following estimate for the mean:

$$\hat{\mu} = \frac{m}{n}\sum_{i=1}^{n}\frac{Y_i(e^\epsilon+1)-1}{e^\epsilon-1}. \tag{9.2}$$

Theorem 9.3 *The above algorithm is ϵ-LDP, and $\hat{\mu}$ is an unbiased estimator of the average. And the variance of $\hat{\mu}$ is $O(\frac{m^2}{n\epsilon^2})$.*

This is the exact same as the Laplace mechanism, but requires much less communication. This is a common resource in LDP algorithm analysis: beyond just trying to optimize the error or amount of data, one can also try to simultaneously minimize the communication.

9.1.1 Differential Privacy at Google

We now discuss a deployment of LDP by Google, which appeared at [9].

Let us first describe one of the problems that arises in their use case. Suppose were in the same situation as before, where we are trying to answer the sum (average) query for bits privately. If we simply ask everyone for their bit once, then simple randomized response works fine. If you simply want to collect data from an individual once, then simple randomized response works fine. However, what if you asked the same question to everyone every day? This would be susceptible to what is known as an averaging attack. For concreteness, lets fix $\epsilon = \ln 3$ so that a user responds with the truth in randomized response with probability 3/4. Suppose the output of randomized response was yes: then this would come up with probability 1/4 if the users value was no, which is fairly high plausible deniability. On the other hand, suppose you asked the same question for 100 days, and 75 of the responses were yes: this would occur with probability only 1.39×10^{-24} if the users value was no. This is essentially composition of differential privacy in action: while we started with $\epsilon \approx 1.1$. 100 rounds would result in an overall value of $\epsilon = 110$. And clearly e^{110} is a large number.

To get around this, a strategy known as memoization is employed. That is, the user simply re-uses a privatized response every time the same question is asked. Instead of introducing new randomness each day, they only compute a randomized response on day 1, and then repeat that value every subsequent day the adversary can not learn anything new from these subsequent disclosures. This is not a foolproof system: for example, an adversary could ask equivalent queries which are phrased in different ways in order to elicit a new randomized response. But it can work in situations when a system honestly trying to preserve user privacy is simply asking the exact same query in subsequent days.

The RAPPOR system is illustrated in Figure 9.2. It works in a more general setting than randomized response, which only works on a single bit. Instead, a user can have an arbitrary value v in some domain $\mathcal{X}$, which it then maps to a length-k binary vector B using what is known as a Bloom filter. More precisely, we have a set of h different hash functions, each of which maps from the domain $\mathcal{X}$ to the set $[k]$ (ideally, uniformly at random). B is then the vector which has a 1 in exactly the h entries which it maps to. For simplicity, you can think of $h = 1$, in which B will be a one-hot encoding of the string v. This would be effective when $|\mathcal{X}| \leq k$. If $|\mathcal{X}|$ is larger, then you would have hash collisions in which multiple v map to a single B, so in these cases we require multiple hash functions.

The first step is what is known as a permanent randomized response. Randomized response is applied bitwise to B, producing a new length-k binary vector B_0 . The resulting B_0 is locally differentially private with respect to the users value v, and is memoized. In the future, whenever the system queries the value v from the user, they will produce the private representation B_0 . There is one additional precaution introduced in

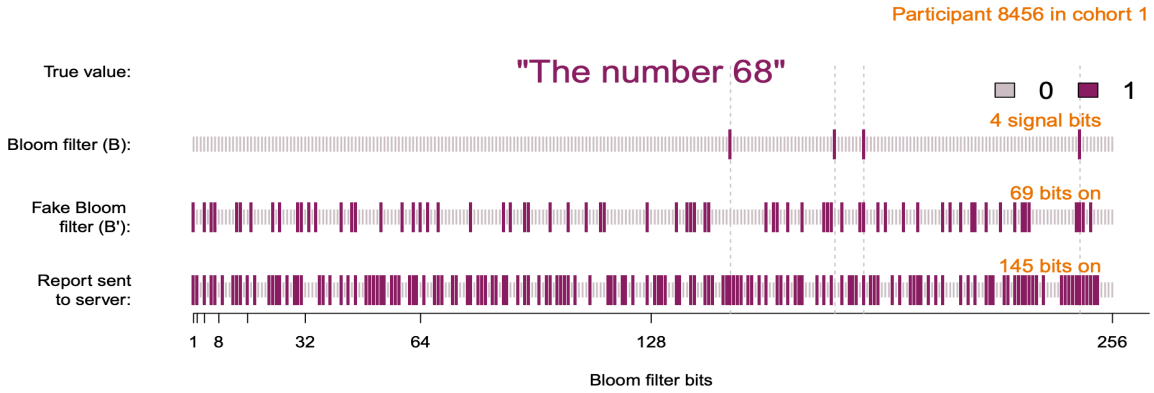

Figure 9.2: RAPPOR Example

RAPPOR. The issue is that a particular B_0 is unlikely to occur for more than one users value, thus resulting in it serving as a type of identifier for the user. Every time the adversary sees a specific value B_0 , they can be fairly certain that this is coming from the same user. Note that this is not a privacy violation within the setting of local differential privacy, which generally assumes that users are associated with the messages they send, but could be a privacy violation in real-world settings. To help mitigate these type of scenarios, in which a B_0 serves as an ad-hoc identifier, there is one last layer of randomization applied by the user to their privatized response B_0 : they apply one more level of bitwise randomized response to this string. In addition to mitigating the effect of the identifier phenomenon, this also increases the strength of the local differential privacy guarantee when an adversary can only view a single response. However, if an adversary is able to view an infinite number of responses, neither of these will be effective we will nonetheless fall back on the local DP guarantees provided by the permanent randomized response.

The RAPPOR system has a way of aggregating these reports, which is somewhat similar to the standard randomized response method but more sophisticated. We dont get into the details in this lecture. The detailed procedure can be found in Figure 9.3.

Figure 9.2 shows a random run of the RAPPOR algorithm. Here, a clients value is $v = 68$, the size of the Bloom filter is $k = 256$, the number of hash functions is $h = 4$, and the tunable randomized response parameters are: $p = 0.25$, $q = 0.75$, and $f = 0.5$. The reported bit array sent to the server is shown at the bottom of the figure. 145 out of 256 bits are set in the report. Of the four Bloom filter bits in B (second row), two are propagated to the noisy Bloom filter B_0. Of these two bits, both are turned on in the final report. The other two bits are never reported on by this client due to the permanent nature of B_0. With multiple collections from this client on the value 68, the most powerful attacker would eventually learn B_0 but would continue to have lim- ited ability to reason about the value of B, as measured by differential privacy guarantee. In practice, learning about the actual clients value v is even harder because multiple values map to the same bits in the Bloom filter.

9.2 Shuffle Differential Privacy

Until now we have talked about the Central DP model and the Local DP model and we can see there are many differences between these models. Especially,

1. For many queries, there is always a gap of $O(\sqrt{n})$ in the estimation error between the central and the local model, where n is the size the dataset. For example, consider the average query over $\{0,1\}^n$. In

The RAPPOR algorithm takes in the client's true value v and parameters of execution k, h, f, p, q, and is executed locally on the client's machine performing the following steps:

1. **Signal.** Hash client's value v onto the Bloom filter B of size k using h hash functions.
2. **Permanent randomized response.** For each client's value v and bit $i, 0 \leq i < k$ in B, create a binary reporting value B'_i which equals to
$$B'_i = \begin{cases} 1, & \text{with probability } \frac{1}{2}f \\ 0, & \text{with probability } \frac{1}{2}f \\ B_i, & \text{with probability } 1-f \end{cases}$$
where f is a user-tunable parameter controlling the level of longitudinal privacy guarantee.
Subsequently, this B' is memoized and reused as the basis for all future reports on this distinct value v.
3. **Instantaneous randomized response.** Allocate a bit array S of size k and initialize to 0. Set each bit i in S with probabilities
$$P(S_i = 1) = \begin{cases} q, & \text{if } B'_i = 1. \\ p, & \text{if } B'_i = 0. \end{cases}$$
4. **Report.** Send the generated report S to the server.

Figure 9.3: RAPPOR Framework [9]

the ϵ central DP, we can just add Laplacian noise, *i.e.,* $M(D) = \frac{1}{n}\sum_{i=1}^{n} x_i + \text{Lap}(\frac{1}{n\epsilon})$. The error will be $O(\frac{1}{n\epsilon})$. In the LDP, model, one can use the Randomized Response to get an unbiased estimator of the sum (see Lecture 3) or the Laplacian mechanism to each data sample. And we can show that the error will be $O(\frac{1}{\sqrt{n}\epsilon})$. The same for the ERM problem where the optimal rate of the excess empirical risk is $O(\frac{d\log 1/\delta}{n^2\epsilon^2})$ and $O(\frac{\sqrt{d\log 1/\delta}}{n\epsilon})$ for strongly convex and general convex loss functions in the central (ϵ, δ)-DP model. While it is $O(\frac{d}{n\epsilon^2})$ and $O(\frac{\sqrt{d}}{\sqrt{n}\epsilon})$ in the ϵ or (ϵ, δ)-LDP model. Note that this $O(\sqrt{n})$ is quite large to make LDP to be useful in practice since the number of efficient samples reduce by a factor of $\sqrt{n}$.

2. For the selection problem, there is a exponential gap. That is in the central ϵ-DP model, we can use the exponential mechanism to do a private selection which only has an error that is logarithmic of the number of candidate, while in the LDP model it is at least polynomial in the number of candidate. See [12, 11] for details.

Current many papers study how to deal with the issue 1. That is, how to design new protocol for differential privacy in the distributed setting while has the same error as in the central model. One issue is based on the crypto technique. However, we will not talk about them in this class. What we will introduce in this lecture is a new model which is called Shuffled Differential Privacy (SDP). SDP is motivated by [2], where the authors propose a principled systems architectureEncode, Shuffle, Analyze (ESA). Generally speaking, the Encode step is to encode the users information, this step could be done by using crypto technique or differential privacy. Between the server and all the users there is a shuffler which could shuffle the information from all the users. You can also think this step as an "Anonymization" step since it obfuscates all the users ID to make the algoithm more private. Then finally, the server will decode the information and perform some analysis. Note that in ESA, all the users and the shuffler are trusted and the server is untrusted, while in the LDP model only the users are trusted, in the central model the server is trusted. [2] mainly focus on the practical issue of the ESA architecture such as how to implement or who is responsible for the shuffling

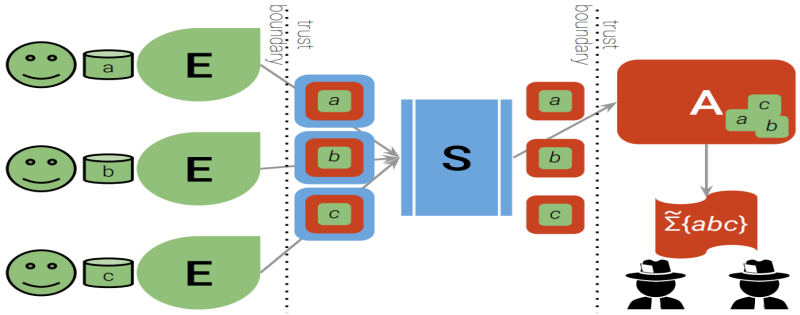

Figure 9.4: ESA architecture: Encode, shuffle, and analyze

procedure. We will not talk about it in this lecture. In the next, we consider the privacy guarantee of the ESA by introducing two work [5, 8] which were published almost at the same period.

9.2.1 Privacy Amplification via Shuffling

[8] consider the shuffler as a tool for privacy amplification. Recall that in the previous lecture we consider the subsampling procedure to amplify the privacy and this is the second we will talk about. Specifically, the authors consider the case where the Encode step is some ϵ_0-LDP algorithm and show that the whole ESA protocol will be (ϵ, δ)-central DP with $\epsilon = O(\epsilon_0\sqrt{\frac{1}{n}})$. In details:

Theorem 9.4 (Theorem 3.1 in [10]) *For any domain $\mathcal{D}$, let $\mathcal{R}^{(i)} : \mathcal{S}^{(1)} \times \cdots \times \mathcal{S}^{(i-1)} \times \mathcal{D}$ for $i \in [n]$ (where $\mathcal{S}^{(i)}$ is the range space of $\mathcal{R}^{(i)}$) be a sequence of algorithms such that $\mathcal{R}^{(i)}(z_{1:i-1}, \cdot)$ is an ϵ_0-DP local randomizer for all values of auxiliary input $z_{1:i-1} \in \mathcal{S}^{(1)} \times \cdots \times \mathcal{S}^{(i-1)}$. Let $A_s : \mathcal{D}^n \mapsto \mathcal{S}^{(1)} \times \cdots \times \mathcal{S}^{(n)}$ be the algorithm that given a dataset $D = \{x_1, \cdots, x_n\}$, samples a uniform random permutation π over $[n]$, then sequentially computes $z_i = \mathcal{R}^{(i)}(z_{1:i-1}, x_{\pi(i)})$ for $i \in [n]$ and outputs $z_{1:n}$. Then for any $\delta \in (0, 1]$ such that $\epsilon_0 \leq \log(\frac{n}{16\log(2/\delta)})$, A_s is (ϵ, δ)-DP where*

$$\epsilon = O(\frac{\sqrt{e^{\epsilon_0}\log(1/\delta)}}{\sqrt{n}}) \text{ when } \epsilon_0 > 1 \text{ and } \epsilon = O(\epsilon_0\frac{\sqrt{\log(1/\delta)}}{\sqrt{n}}) \text{ when } \epsilon_0 \leq 1. \tag{9.3}$$

The theorem is quite interesting. For example, let us consider the average query over $\{0, 1\}^n$. If we want to achieve (ϵ, δ)-DP with $\epsilon = O(\frac{\sqrt{\log 1/\delta}}{\sqrt{n}})$ in the shuffled model. Then it is sufficient to let each local randomizer be $O(\frac{\epsilon\sqrt{n}}{\sqrt{\log 1/\delta}}) < 1$-LDP. Thus, we can just add $O(\text{Lap}(\frac{\sqrt{\log 1/\delta}}{\epsilon\sqrt{n}})$ to each sample x_i, and aggregate $\frac{1}{n}\sum_{i=1}^n(x_i + O(\text{Lap}(\frac{\sqrt{\log 1/\delta}}{\epsilon\sqrt{n}}))$. The final error is $O(\frac{1}{n\epsilon})$ which is the same as in the central DP model.

While the previous result is quite interesting in theory, it is not very useful since $\epsilon = O(\epsilon_0\sqrt{\frac{1}{n}})$ only when $\epsilon_0 < 1$ which means $\epsilon = O(\frac{1}{\sqrt{n}})$ which is quite small as in practice it is sufficient to let $\epsilon = 0.1 - 1$. At the same time [5] also study the average query in the shuffling model.

9.2.2 Shuffled Differential Privacy

Instead of focusing on the bridge between LDP and DP by using the shuffling, [5] consider a general model which is called the SDP model. Note that in the SDP model the encoder procedure could not be LDP algorithm.

Definition 9.5 *In the model, there are n users, each of whom holds data $x_i \in \mathcal{D}$. We will use $D = \{x_1, \cdots, x_n\} \in \mathcal{D}^n$ to denote the dataset of all n users data. We say two datasets D, D' are neighboring if they differ on at most one users data, and denote this $D \sim D'$. The protocols we consider consist of three algorithms:*

- *$\mathcal{R} : \mathcal{D} \mapsto \mathcal{Y}^m$ is a randomized encoder that takes as input a single users data x_i and outputs a set of m messages $y_{i,1}, y_{i,2}, \cdots, y_{i,m} \in \mathcal{Y}^m$. If $m = 1$, the protocol is the one-message shuffled model.*
- *$S : \mathcal{Y}^* \mapsto \mathcal{Y}^*$ is a shuffler that takes a set of messages and outputs these messages in a uniformly random order.*
- *$A : \mathcal{Y}^* \mapsto \mathcal{Z}$ is some analysis function or analyzer that takes a set of messages $y_1, \cdots, y_N$ and attempts to estimate some function $f(x_1, x_2, \cdots, x_n)$ from these messages.*

We denote the overall protocol $P = (R, S, A)$. The mechanism by which we achieve privacy is

$$\Pi_R(x_1, \cdots, x_n) = S(\cup_{i=1}^n \mathcal{R}(x_i)) = S(y_{1,1}, \cdots, y_{n,m}),$$

where both R and S are randomized. A protocol $P = (R, S, A)$ is (ϵ, δ)-Shuffled Differentially Private (SDP) if the algorithm $\Pi_R(x_1, \cdots, x_n)$ is (ϵ, δ)-DP.

For the average query, [5] showed the following result:

Theorem 9.6 *For every $\epsilon \in (0, 1]$ and $\delta \geq 2^{-\epsilon n}$ and every function $f : \mathcal{D} \mapsto \{0, 1\}$ there is a protocol P that is (ϵ, δ)-SDP and for every n and every $D = \{x_1, \cdots, x_n\}$ we have*

$$\mathbb{E}|P(D) - \frac{1}{n}\sum_{i=1}^n f(x_i)| = O(\frac{1}{n\epsilon}\sqrt{\log 1/\delta}). \tag{9.4}$$

Each user sends a single one-bit message.

Actually, the protocol corresponding is extremely simple:

- For some appropriate choice of $p \in (0, 1)$, each user i with input x_i outputs $y_i = x_i$ with probability $1 - p$ and a uniform bit y_i with probability $1 - p$ with $p \approx \frac{\log 1/\delta}{n\epsilon^2}$.
- The analyzer collects the shuffled messages $y_1, \cdots, y_n$ and outputs

$$\frac{1}{n(1-p)}(\sum_{i=1}^n y_i - \frac{p}{2}). \tag{9.5}$$

Note that SDP current is a hot topic in Differential Privacy as it could achieve the same bound as in the central model in the distributed setting. However, there are still some limitations. The first one is it still cannot deal with the selection problem [6]. That is, in machine learning it cannot deal with the high dimensional case where the dimensional is far greater than the sample size n. Even in the low dimensional case, there is still a separation between SDP and central DP, such as the histogram [4]. Thus, instead of considering the shuffler as a tool for privacy amplification, it is quite interesting to consider the SDP model.

References

[1] Raef Bassily and Adam Smith. Local, private, efficient protocols for succinct histograms. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 127–135, 2015.

[2] Andrea Bittau, Úlfar Erlingsson, Petros Maniatis, Ilya Mironov, Ananth Raghunathan, David Lie, Mitch Rudominer, Ushasree Kode, Julien Tinnes, and Bernhard Seefeld. Prochlo: Strong privacy for analytics in the crowd. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 441–459, 2017.

[3] TH Hubert Chan, Elaine Shi, and Dawn Song. Optimal lower bound for differentially private multi-party aggregation. In *European Symposium on Algorithms*, pages 277–288. Springer, 2012.

[4] Albert Cheu. Differential privacy in the shuffle model: A survey of separations. *arXiv preprint arXiv:2107.11839*, 2021.

[5] Albert Cheu, Adam Smith, Jonathan Ullman, David Zeber, and Maxim Zhilyaev. Distributed differential privacy via shuffling. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 375–403. Springer, 2019.

[6] Albert Cheu and Jonathan Ullman. The limits of pan privacy and shuffle privacy for learning and estimation. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1081–1094, 2021.

[7] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. Collecting telemetry data privately. *arXiv preprint arXiv:1712.01524*, 2017.

[8] Úlfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Abhradeep Thakurta. Amplification by shuffling: From local to central differential privacy via anonymity. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2468–2479. SIAM, 2019.

[9] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, pages 1054–1067, 2014.

[10] Vitaly Feldman, Audra McMillan, and Kunal Talwar. Hiding among the clones: A simple and nearly optimal analysis of privacy amplification by shuffling. *IEEE Symposium on Foundations of Computer Science (FOCS), 2021*, 2021.

[11] Jonathan Ullman. Tight lower bounds for locally differentially private selection. *arXiv preprint arXiv:1802.02638*, 2018.

[12] Di Wang and Jinhui Xu. On sparse linear regression in the local differential privacy model. In *International Conference on Machine Learning*, pages 6628–6637. PMLR, 2019.