CS 860: Algorithms for Private Data Analysis Fall 2020

Lecture 17 — Deployments of DP: Local Differential Privacy

Prof. Gautam Kamath *Scribe: Gautam Kamath*

Today, we begin a series of lectures on deployments of differential privacy. In the first such lecture, we begin with deployments which operate in the local model of differential privacy.

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.

However, in many common use cases, this may not be an acceptable threat model. In particular, there might not be access to a trusted curator. You could consider the example given at the beginning of this course, where an instructor is trying to estimate what fraction of their class cheated on a test. Naturally, an individual would hesitate to share their data with the instructor. We will see more examples related to IT companies in today's lecture. 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.

To see this in action, let us revisit an example we studied a the beginning of this class. Suppose we want to estimate the average of a dataset of bits $X_1, \ldots, X_n$, that is, $\mu = \frac{1}{n}\sum_i X_i$. In the central model of differential privacy, we would use the Laplace mechanism:

$$\hat{\mu} = \frac{1}{n}\sum_i X_i + \text{Laplace}\left(\frac{1}{\varepsilon n}\right).$$

The noise introduced due to privacy is of magnitude $1/\varepsilon n$, and thus

$$|\hat{\mu} - \mu| \leq \frac{1}{\varepsilon n}.$$

If we wish to bound this error by α, we require $n \geq 1/\alpha\varepsilon$.

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 $\frac{e^\varepsilon}{1+e^\varepsilon}$, and $1 - X_i$ with probability $\frac{1}{1+e^\varepsilon}$. It is easy to see that the user's disclosure is ε-DP, just by taking the ratio of these probabilities. The curator (no longer trusted) can aggregate these responses as follows:

$$\hat{\mu} = \frac{1}{n}\sum_i \left(\frac{e^\varepsilon + 1}{e^\varepsilon - 1} \cdot Y_i - \frac{1}{e^\varepsilon - 1}\right).$$

Simple computations reveal that this is an unbiased estimator for μ, and if $\varepsilon = O(1)$, the variance is of order $O\left(\frac{1}{\varepsilon^2 n}\right)$. By Chebyshev's inequality, this leads to

$$|\hat{\mu} - \mu| \leq \frac{1}{\varepsilon\sqrt{n}}.$$

If we wish to bound this error by α, we require $n \geq 1/\alpha^2\varepsilon^2$.

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 we discuss in today's lecture are at companies which have access to massive datasets.

There are many interesting technical properties and questions involving local differential privacy. For instance, 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. Another oddity is that 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. All of these technical questions are fascinating, but unfortunately we do not have time to discuss them during this course.

Historically, randomized response was introduced by Warner in 1965 [War65]. Evfimievski, Gehrke, and Srikant had a similar definition in 2003 [EGS03]. It was first formulated and explored in the language of differential privacy by Kasiviswanathan, Lee, Nissim, Raskhodnikova, and Smith in 2008 [KLN+11]. A paper by Duchi, Jordan, and Wainwright is known to have popularized the notion in various communities [DJW13].

Differential Privacy at Google

We first discuss RAPPOR [EPK14], which was introduced by Google in 2014 and is among the first major deployments of differential privacy.

Let us first describe one of the problems that arises in their use case. Suppose we're in the same situation as before, where we are trying to find how many people cheated on a test. 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, let's fix $\varepsilon = \ln 3$, 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 user's 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 user's value was "no." This is essentially composition of differential privacy in action: while we started with $\varepsilon = \ln 3 \approx 1.1$, 100 rounds would result in an overall value of $\varepsilon = 110$. And clearly e^{110} is a large number.

To get around this, a strategy known as memoization is employed (not memorization, though it's kind of the same thing). 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 1. 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 $|\mathcal{X}| \ll k$. 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'. The resulting B' is locally differentially private with respect to the user's 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'.

There is one additional precaution introduced in RAPPOR. The issue is that a particular B' is unlikely to occur for more than one user's value, thus resulting in it serving as a type of identifier for the user. Every time the adversary sees a specific value B', 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' serves as an ad-hoc identifier, there is one last layer of randomization applied by the user to their privatized response B': 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 don't get into the details in this lecture.

To provide one case which RAPPOR explicitly can *not* handle, is when the values in subsequent reports do differ, but only by a small amount. As one example, suppose you ask each user their age in days, every day. RAPPOR would have to generate a new B for every such query, and lose the guarantees of the permanent randomized response would be lost. We will see ways of getting around this in the Microsoft solution [DKY17].

RAPPOR was deployed as part of Google Chrome, in order to monitor various aggregate statistics of users. One example mentioned in the paper is when malware hijacks and modifies a user's machine, changing settings such as their Internet browser's homepage or search engine. Google has

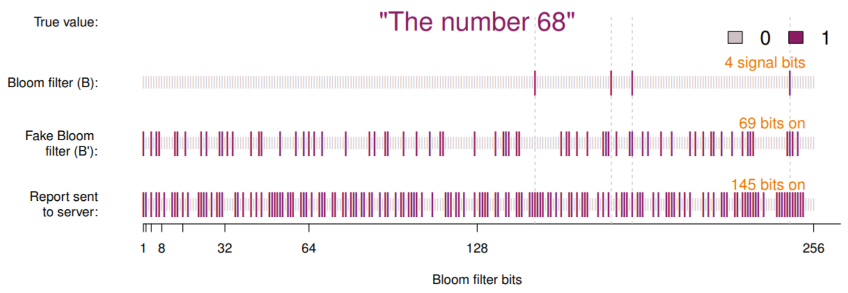


Figure 1: Figure from [EPK14], demonstrating the RAPPOR system.

an opt-in feature whereby users can report these statistics, which are appropriately privatized using RAPPOR. In the paper, they present a histogram of statistics on which Google Chrome homepages are most popular, barring common or expected ones. As another example, they also investigate presence of various processes on Windows computers, some of which may be malicious.

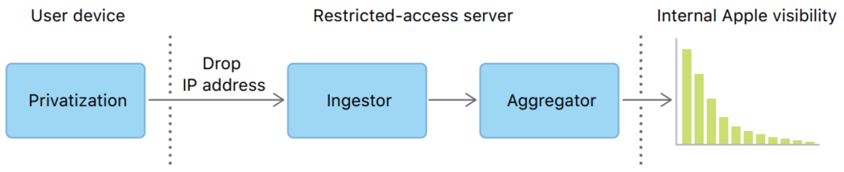
Differential Privacy at Apple

Apple is another high-profile adopter of local differential privacy [Dif17]. We proceed to describe some features of their deployment.

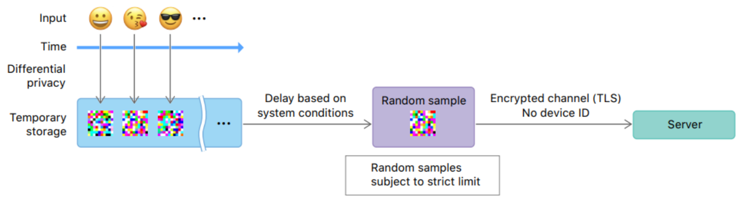
Their architecture is displayed in Figure 2. In addition to their LDP randomization (which we will discuss shortly), they employ other methods to ensure user privacy. Some features include delays in transmission of messages, random subsampling of generated messages, de-identification including removal of IP addresses, and TLS encryption. There is a limit on the amount of privacy budget which can be expended per day. That said, at roughly the same time as Apple described their deployment, some researchers investigated and criticized the parameters of their deployment [TKB+17].

Most of the algorithms described the Apple white paper are based around the celebrated Count Sketch algorithm, by Charikar, Chen, and Farach-Colton [CCFC02]. This is a well-known technique used for counting freqencies of items within a stream, and it is a method that is particularly convenient for privatization. One factor that was not discussed in the RAPPOR paper is the *communication cost* of a procedure. Ideally, to reduce the burden on individual users, we would like to minimize the amount of data that the user must send. The paper presents a version of this algorithm which only requires the user to send a single bit to the server. We will discuss more about communication efficient algorithms in the next deployment we investigate.

One feature of the Apple deployment that RAPPOR does not support is the ability to discover new words, which were not previously in the dictionary. Recall when discussing RAPPOR, each value would be hashed by a number of functions (i.e., passed through a Bloom filter) in order to generate a representation as a k-bit binary string. In order to discover the frequency of a specific value, one would need to discover the corresponding string, which is not possible if the value wasn't known beforehand. To discover, say, new strings of length 10, one would have to enumerate over 26^{10} possibilities, which would be prohibitively expensive and also result in false positives for many strings that weren't in the dataset.



System Overview



Privatization Stage

Figure 2: Figures from [Dif17], illustrating their architecture.

The Apple DP paper uses a new approach which they term the Sequence Fragment Puzzle, which gets around this by breaking longer words into short sub-strings, which we *can* enumerate over. In some more detail, their approach can be visualized in a two stage process: in the first stage, new words which occur frequently are discovered, and in the second stage, the frequencies of these words are computed. The second stage is standard, so we describe only the first stage. Succinctly speaking: a word is broken up into (short) substrings, and the client sends a random substring concatenated with a hash of the overall string (which they call the "puzzle piece") (privatized), along with the index at which the substring starts (not privatized). Once these are transmitted, the server can discover (for each index) which substring/puzzle piece pairs are common (if we use a hash function which maps to a limited range, then we can enumerate over all possibilities since the substrings are short), and match up substrings which share the same puzzle pieces.

Let's do a quick example. Suppose we wish to privately transmit the word "waterloo," which a function (which maps to the range 0 through 255) hashes to the value 42. We would then send one of the following four messages, chosen uniformly at random: (1,wa,42), (3,te,42), (5,rl,42), (7,oo,42). While the index in the first entry of the tuple is transmitted in the clear (and in fact is not data dependent), the other two elements of the tuple are privatized. If enough people send the word "waterloo," then the server will observe that all four of those messages occur frequently – note that there are only 26^2 possibilities for the substring and 255 values for the hash, which multiply to 172380, which can be enumerated. By matching the strings that have the same puzzle piece 42, it is clear to see that the new phrase is waterloo, which we can then estimate the frequency of using a new query.

This specific method was used to discover new words typed by users of Apple devices which were not previously in Apple's dictionary. Some examples include abbreviations like "wyd," "wbu," and "idc," and phrases such as "bruh" and "bae." Interestingly, they also discovered words such as "lov" and "th," which don't immediately seem to have meaning. However, if one presses the left-most prediction cell when using an iOS device, it will use the current word verbatim, even if it is not in the dictionary. The speculation is that users were trying to type the words “love” and “the,” but frequently missed the letter “e” and hit this prediction (which is right above) instead.

Another interesting application is a discovering frequency of emoji usage. Figure 3 demonstrates emojis used in both English and French, noticing significant differences in usage habits. Other applications include discovering when users did or didn’t want videos to auto-play on various sites in the Safari web browser, determining which websites required high energy or memory (also in Safari), and understanding which health data types are most popular in Apple’s Health app. These applications are all sensitive in nature, but can be used to improve the user experience when revealed in a privacy-preserving manner.

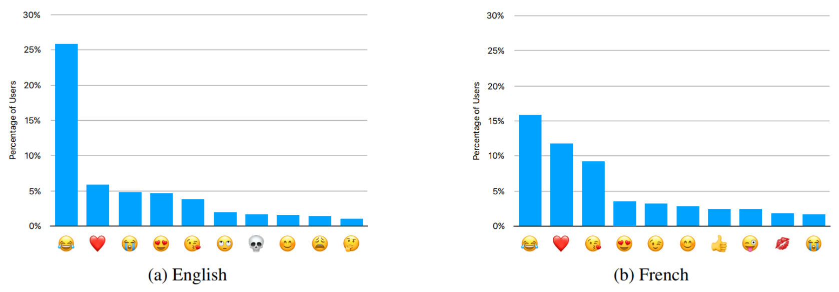


Figure 3: Figures from [Dif17], emoji usage in English versus French.

Differential Privacy at Microsoft

Finally, we discuss a deployment of differential privacy by Microsoft, in a paper by Ding, Kulkarni, and Yekhanin [DKY17]. In particular, we describe their one-bit protocol for mean estimation, and also their alternative strategy for memoization, which is applicable in more situations than RAPPOR [EPK14]. Algorithms from this paper have been implemented in Windows 10. They are used to estimate the amount of time that users spend in various applications.

Low-Communication LDP

Suppose we wish to average a set of numbers in $[0, m]$ under the constraint of ε-LDP. 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 m/ε, where m is the sensitivity of each user’s datapoint. The curator then averages these results to obtain an estimate of the mean:

$$\hat{\mu} = \frac{1}{n}\sum_i \left(X_i + \text{Laplace}(m/\varepsilon)\right).$$

Taking the difference between the estimated mean and the true mean, we get

$$\hat{\mu} - \frac{1}{n}\sum_i X_i = \frac{1}{n}\sum_i \text{Laplace}(m/\varepsilon),$$

and the standard deviation of this error is $O\left(\frac{m}{\varepsilon\sqrt{n}}\right)$. Duchi, Jordan, and Wainwright proved that this error is optimal for this setting [DJW13]. 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. 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 $\frac{1}{e^\varepsilon+1} + \frac{X_i}{m} \cdot \frac{e^\varepsilon-1}{e^\varepsilon+1}$, 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:

$$\hat{\mu} = \frac{m}{n} \sum_i \frac{Y_i \cdot (e^\varepsilon + 1) - 1}{e^\varepsilon - 1}$$

It is not hard to see this is unbiased:

$$\begin{aligned}
\mathbf{E}\left[\hat{\mu}\right] &= \frac{m}{n} \sum_i \frac{\mathbf{E}[Y_i] \cdot (e^\varepsilon + 1) - 1}{e^\varepsilon - 1} \\
&= \frac{m}{n} \sum_i \frac{\left(\frac{1}{e^\varepsilon+1} + \frac{X_i}{m} \cdot \frac{e^\varepsilon-1}{e^\varepsilon+1}\right) \cdot (e^\varepsilon + 1) - 1}{e^\varepsilon - 1} \\
&= \frac{1}{n} \sum_i X_i
\end{aligned}$$

A little more effort (via a Chernoff bound) shows that the error is $O\left(\frac{m}{\varepsilon\sqrt{n}}\right)$. 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.

Memoization

Recall in the RAPPOR paper [EPK14], they used memoization: if the user is requested to send the same value multiple times, they privatize the value only once and re-send the same result. This may not be realistic in certain settings, as values are unlikely to be the exact same for multiple time periods. For example, if the statistic of interest is the amount of time spent in an application per day, this is highly unlikely to be the exact same value, but it may vary only a small amount – say, if times only differ by 15 minutes daily.

One simple solution is to appropriately discretize the range of values and round to the center of each bin, but this raises a dilemma. If we create too many bins (discretizing too finely), the user would have to re-randomize for small fluctuations in their value. If we create too few bins (discretizing too coarsely), users would incur significant error when they round to the center of each bin. The authors provide a randomized rounding scheme which allows them to eliminate the latter concern, allowing them to use a coarse discretization but avoid the corresponding error.

We describe a specific instantiation of their scheme. At the start of the process, each user i precomputes and memoizes the result of running the above 1-bit mean estimation algorithm, as if their value was 0, and as if their value was m. Denote these results $A_i(0)$ and $A_i(m)$. User i also selects an offset $\alpha_i \in \{0, \dots, m-1\}$ uniformly at random. Then, if user i's value for a request is $x_i \in [0, m]$, and $x_i + \alpha_i \leq m$, then they output $A_i(0)$. Otherwise, that implies $x_i + \alpha_i > m$, and they output $A_i(m)$. The authors show that this algorithm enjoys the same accuracy guarantees as the simple 1-bit mean estimation algorithm described above.

To give one concrete benefit of this approach as compared to RAPPOR: recall the query we mentioned earlier, where a user is asked for their age each day. The RAPPOR paper said that this query could not be supported, as it would trigger many LDP randomizations. However, with the solution in this paper, the output would be appear as a sequence of $A_i(0)$'s followed by a sequence of $A_i(m)$'s – this only reveals when the user's age passes $m - \alpha_i$, which is private based on the user's random choice of α_i.

References

[CCFC02] Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding frequent items in data streams. In *Proceedings of the 29th International Colloquium on Automata, Languages, and Programming*, ICALP '02, pages 693–703, 2002.

[Dif17] Differential Privacy Team, Apple. Learning with privacy at scale. https://machinelearning.apple.com/docs/learning-with-privacy-at-scale/appledifferentialprivacysystem.pdf, December 2017.

[DJW13] John C. Duchi, Michael I. Jordan, and Martin J. Wainwright. Local privacy and statistical minimax rates. In *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '13, pages 429–438, Washington, DC, USA, 2013. IEEE Computer Society.

[DKY17] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. Collecting telemetry data privately. In *Advances in Neural Information Processing Systems 30*, NIPS '17, pages 3571–3580. Curran Associates, Inc., 2017.

[EGS03] Alexandre Evfimievski, Johannes Gehrke, and Ramakrishnan Srikant. Limiting privacy breaches in privacy preserving data mining. In *Proceedings of the 22nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '03, pages 211–222, New York, NY, USA, 2003. ACM.

[EPK14] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. RAPPOR: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM Conference on Computer and Communications Security*, CCS '14, pages 1054–1067, New York, NY, USA, 2014. ACM.

[KLN$^+$11] Shiva Prasad Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. What can we learn privately? *SIAM Journal on Computing*, 40(3):793–826, 2011.

[TKB$^+$17] Jun Tang, Aleksandra Korolova, Xiaolong Bai, Xueqiang Wang, and Xiaofeng Wang. Privacy loss in Apple's implementation of differential privacy on MacOS 10.12. *arXiv preprint arXiv:1709.02753*, 2017.

[War65] Stanley L. Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309):63–69, 1965.