15-316: Software Foundations of Security and Privacy **Spring 2017**

Lecture 22: Provable Privacy II

Lecturer: Matt Fredrikson

22.1 Recap

In the previous lecture, we introduced **differential privacy**, which relaxed an unnatainable notion of absolute privacy we'd previously discussed.

Definition 22.1 ϵ-Differential Privacy. *Let $\epsilon > 0$. A randomized function* San *satisfies ϵ-differential privacy if for all possible databases X_1 and X_2 that* differ in exactly one row*, and all $s \in \mathsf{Range}(\mathsf{San})$, the following inequality holds:*

$$\Pr[\mathsf{San}(X_1) = s] \leq e^{\epsilon} \times \Pr[\mathsf{San}(X_2) = s] \tag{22.1}$$

The probabilities in this expression are taken over the randomness of San*'s outputs.*

Question *Why do you think that Definition 22.1 uses a multiplicative bound rather than an additive one? What types of behaviors between neighboring databases would be allowed, that are not in the current definition, if the guarantee was instead:*

$$\Pr[\mathsf{San}(X_1) = s] \leq \Pr[\mathsf{San}(X_2) = s] + \epsilon$$

Recall we'd said that this is a **relativized** notion of privacy. Rather than making absolute guarantees about what can be inferred from access to a database, differential privacy guarantees that for any individual in the database, the likelihood that some fact can be inferred about them is not much greater than would have been possible were their data not in the database. The degree to which "not much" refers is controlled by the **privacy budget** ϵ: small budgets are good for privacy and generally negate utility, and large values enable favorable utility but offer little privacy.

What does differential privacy's relativized guarantee mean for privacy? Importantly, it is usually not possible to conclude that the information provided by a differentially-private mechanism will not enable someone to infer a harmful fact about an individual in the database. As we concluded when we discussed absolute privacy, preventing such harm is impossible in the general case. So if a query looks over a large database of individuals' health records for evidence of correlation between a certain genetic marker and mental illness, and the results aren't likely to change much given a single invidiual's data, then any harm that comes from applying this correlation to individuals will not be (and is not intended to be) prevented by differential privacy. The *only* guarantee offered by Definition 22.1 is that any individual's genetic marker, and mental health status, will not be much more inferrable if their data is used in the computation.

We then talked about how to build differentially-private computations, by converting certain types of non-private computations into randomized variants. We discussed the **Laplace mechanism**, which works by adding noise from the Laplace distribution to the output of a computation. The "amount" of noise is dictated by the computation's global sensitivity, which corresponds to the maximum amount that the output can change between neighboring inputs.

Definition 22.2 Global Sensitivity. *Assume that* San *is a function* $\mathsf{San} : \mathbf{X} \mapsto \mathbb{R}$*, where* $\mathbf{X}$ *is the set of all databases up to a particular size accepted by* San*. Then the* global sensitivity *of* San*, written* $\Delta\mathsf{San}$*, is defined as:*

$$\Delta\mathsf{San} = \max_{X_1, X_2} |\mathsf{San}(X_1) - \mathsf{San}(X_2)|$$

where X_1, X_2 *are neighboring databases.*

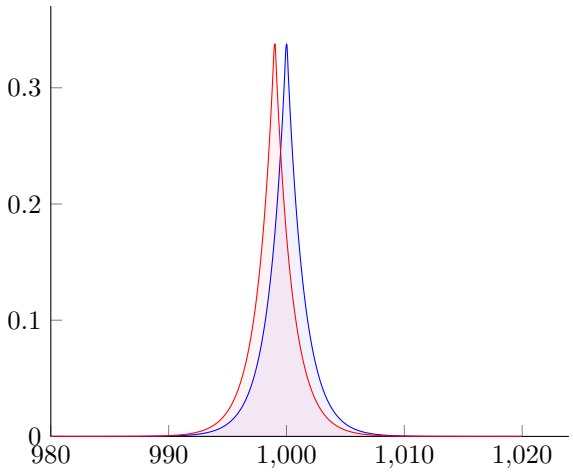
Once we know the sensitivity of San, applying the Laplace mechanism is as simple as sampling from $\mathsf{Lap}(\Delta\mathsf{San}/\epsilon)$ and adding two numbers.

Theorem 22.3 *(Dwork et al., 2006) Assume that* San *is a function* $\mathsf{San} : \mathbf{X} \mapsto \mathbb{R}$*. Then the function*

$$\widehat{\mathsf{San}}(X) = \mathsf{San}(X) + \mathsf{Lap}\left(\frac{\Delta\mathsf{San}}{\epsilon}\right)$$

satisfies ϵ*-differential privacy.*

Depicted graphically, the output distribution of a San function that counts the number of rows satisfying a particular query are depicted as follows. The blue curve depicts the probability of producing a particular output when the true number of rows matching the query is 1,000, and the red curve when the answer is 999.



The main thing to notice about this plot is that no value is significantly more or less likely when the count is 1,000 than when it is 999, and vice versa. Thinking adversarially, if our goal is to deduce the correct (non-noised) count, how do we proceed?

1. Consider each value on the x-axis, and look at the probability of that value for $\mathsf{count} = 1,000$ and $\mathsf{count} = 999$.
2. Pick the value corresponding to the larger probability.

Where the two graphs intersect, the adversary has a tie that is best broken by flipping a fair coin. The probability of this is fairly high, in fact not far off from the probability of either correct answer. Otherwise, the graph shows us that the probability of inferring the correct value is only slighly greater than not.

22.2 Building Differentially-Private Computations, Continued

22.2.1 Randomized response

Last lecture we briefly mentioned randomized response, which is a privacy technique dating back to the 1960s with roots in the social sciences. Randomized response was motivated by survey collection, in situations where questions asked of respondents touch on sensitive or taboo issues. Randomized response gives respondents plausible deniability, by giving them a structured way of adding random noise to their answer.

We'll take a closer look at the technique and see how it relates to differential privacy. For the following discussion, we'll assume that $\mathsf{San} : \mathbf{X} \mapsto \{\mathsf{true}, \mathsf{false}\}$ is a function that maps a **single row** to Booleans, and $\mathsf{flip}(p)$ is a random function that flips a biased coin with parameter p. In other words,

$$\mathsf{flip}() = \begin{cases} \mathsf{true} & \text{with probability } 1/2 \\ \mathsf{false} & \text{with probability } 1/2 \end{cases} \tag{22.2}$$

Then the randomized response routine for row x_i is as follows.

```
function RandResp(x_i) :
  b_1 := flip()
  if b_2 = true then
    return San(x_i)
  else
    b_2 := flip()
    return b_2
```
(22.3)

In short, randomized response tells individual i to return their true answer with probability $1/2$, and a uniform random answer with probability $1/2$.

Theorem 22.4 *The procedure* RandResp *satisfies* $\ln(3)$*-differential privacy.*

Proof: Recall that we need to show that the following inequality holds over all pairs of neighboring databases and all outputs s:

$$\Pr[\mathsf{RandResp}(X_1) = s] \leq e^{\epsilon} \times \Pr[\mathsf{RandResp}(X_2) = s]$$

There are two possible configurations of neighboring databases: $X_1 = [\mathsf{true}], X_2 = [\mathsf{false}]$ and $X_1 = [\mathsf{false}], X_2 = [\mathsf{true}]$.

Let's consider the first configuration for output true. We see that:

$$\Pr[\mathsf{RandResp}([\mathsf{true}]) = \mathsf{true}] = \Pr[b_1 = \mathsf{true}] + \Pr[b_1 = \mathsf{false} \wedge b_2 = \mathsf{true}] \tag{22.4}$$

$$= 1/2 + 1/4 = 3/4 \tag{22.5}$$

Similarly for the right-hand side of the inequality:

$$\Pr[\mathsf{RandResp}([\mathsf{false}]) = \mathsf{true}] = \Pr[b_1 = \mathsf{false} \wedge b_2 = \mathsf{true}] \tag{22.6}$$

$$= 1/2 \cdot 1/2 = 1/4 \tag{22.7}$$

So we have:

$$\frac{\Pr[\mathsf{RandResp}([\mathsf{true}]) = \mathsf{true}]}{\Pr[\mathsf{RandResp}([\mathsf{false}]) = \mathsf{true}]} = \frac{3/4}{1/4} = 3 \tag{22.8}$$

Moving onto the output false:

$$\Pr[\mathsf{RandResp}([\mathsf{true}]) = \mathsf{false}] = \Pr[b_1 = \mathsf{false} \wedge b_2 = \mathsf{false}] \tag{22.9}$$
$$= 1/2 \cdot 1/2 = 1/4 \tag{22.10}$$

And for the other side:

$$\Pr[\mathsf{RandResp}([\mathsf{false}]) = \mathsf{false}] = \Pr[b_1 = \mathsf{true}] + \Pr[b_1 = \mathsf{false} \wedge b_2 = \mathsf{false}] \tag{22.11}$$
$$= 1/2 + 1/2 \cdot 1/2 = 3/4 \tag{22.12}$$

We see that in this case, the inequality already holds. Identical reasoning would apply for the second configuration of neighboring databases, so we can say that the following holds for $\mathsf{RandResp}$:

$$\forall X_1, X_2, s.\mathsf{Neighbor}(X_1, X_2) \implies \Pr[\mathsf{RandResp}(X_1) = s] \leq 3 \times \Pr[\mathsf{RandResp}(X_2) = s] \tag{22.13}$$

We conclude that $\mathsf{RandResp}$ satisfies $\ln(3)$-differential privacy. ■

One noteworthy aspect of randomized response is that it need not take place in the "centralized" statistical database model. Data processors can obtain information about individuals through this privacy-preserving process before storing it in a database, and using some of the composition properties that we'll discuss later, produce aggregate results later on.

Question. *Suppose that we wanted to generalize randomized response with a parameter p that controls the probability of truthful response. We can do so by adding this parameter to* flip*, so that* $\mathsf{flip}(p)$ *returns* true *with probability p and* false *with probability $1 - p$. What conditions does p need to satisfy to ensure ϵ-differential privacy?*

22.2.2 Exponential Mechanism

We've seen how to achieve differential privacy by adding noise. What do we do if the result of our computation doesn't allow additive noise from an appropriate distribution? For example, if the result is a predicted label such as "dog", "cat", "other"? One approach for answering such queries relies on the so-called **exponential mechanism**.

The exponential mechanism is based on a **scoring function**. Suppose that $\mathsf{San} : \mathbf{X} \mapsto \mathbf{O}$. The a scoring function $q : \mathbf{X} \times \mathbf{O} \mapsto \mathbb{R}$ maps pairs of inputs and outputs of San to the reals. Intuitively, given input X and output O, we want $q(X, O)$ to return a large value whenever O is close to $\mathsf{San}(X)$, and a low value otherwise.

Example 22.5 *Suppose that we wish to answer the query "what is the most common class of students in 15316?". Suppose without loss of generality that there are four answers,* $\{\mathrm{Fr}, \mathrm{So}, \mathrm{Ju}, \mathrm{Se}\}$*, and so X consists of sequences from this set. Then a natural choice for our scoring function is:*

$$q(X, O) = |\{x_i \mid x_i = O\}| \tag{22.14}$$

In other words, q is just the number of students in X whose class matches a particular output value. Notice that the correct answer has the highest score according to this choice of q.

Example 22.6 *Notice that we can create a scoring function for any deterministic function* San *as follows:*

$$q(X, O) = \begin{cases} 1 & \text{if } \mathsf{San}(X) = 0 \\ 0 & \text{otherwise} \end{cases} \tag{22.15}$$

However, the only information that such a scoring function provides is whether O is the output that is mapped by X. Importantly, it does not provide any information about how "close" a candidate output O is, or how well it approximates the correct output. A scoring function is more useful if it provides such information. Whenever $\mathbf{O}$ has a useful measure of distance, it can be exploited to achieve this structure, e.g.,

$$q(X, O) = |O - \mathsf{San}(X)| \tag{22.16}$$

Notice that this generalizes all of the functions that we could apply the Laplace mechanism to.

Now that we're comfortable with scoring functions and the generality that they afford, we can introduce the exponential mechanism.

Definition 22.7 *Exponential Mechanism Let $\epsilon > 0$, and $q : \mathbf{X} \times \mathbf{O} \mapsto \mathbb{R}$ be any scoring function. Then given an input $X \in \mathbf{X}$, produce an output according to the following rule:*

$$\mathsf{Expmech}(X) = \text{output } O \text{ with probability proportional to } \exp\left(\frac{\epsilon q(X, O)}{2\Delta q}\right) \tag{22.17}$$

Theorem 22.8 *The exponential mechanism (Definition 22.7) satisfies ϵ-differential privacy (Definition 22.1).*

Proof: Let X and X' be neighboring databases. The probability that the mechanism outputs a given O on X and X is given in Equations 22.18 and 22.19, respectively.

$$\Pr[\mathsf{Expmech}(X) = O] = \frac{\exp\left(\epsilon q(X, O)/2\Delta q\right)}{\sum_{O' \in \mathbf{O}} \exp\left(\epsilon q(X, O')/2\Delta q\right)} \tag{22.18}$$

$$\Pr[\mathsf{Expmech}(X') = O] = \frac{\exp\left(\epsilon q(X', O)/2\Delta q\right)}{\sum_{O' \in \mathbf{O}} \exp\left(\epsilon q(X', O')/2\Delta q\right)} \tag{22.19}$$

Then we can proceed by calculation:

$$\begin{aligned}
\frac{\Pr[\mathsf{Expmech}(X) = O]}{\Pr[\mathsf{Expmech}(X') = O]} &= \left(\frac{\exp\left(\epsilon q(X, O)/2\Delta q\right)}{\exp\left(\epsilon q(X', O)/2\Delta q\right)}\right)\left(\frac{\sum_{O' \in \mathbf{O}} \exp\left(\epsilon q(X', O')/2\Delta q\right)}{\sum_{O' \in \mathbf{O}} \exp\left(\epsilon q(X, O')/2\Delta q\right)}\right) \\
&= \exp\left(\frac{\epsilon(q(X, O) - q(X', O))}{2\Delta q}\right)\left(\frac{\sum_{O' \in \mathbf{O}} \exp\left(\epsilon q(X', O')/2\Delta q\right)}{\sum_{O' \in \mathbf{O}} \exp\left(\epsilon q(X, O')/2\Delta q\right)}\right) \\
&\leq \exp\left(\frac{\epsilon \Delta q}{2\Delta q}\right)\left(\frac{\sum_{O' \in \mathbf{O}} \exp\left(\epsilon q(X', O')/2\Delta q\right)}{\sum_{O' \in \mathbf{O}} \exp\left(\epsilon q(X, O')/2\Delta q\right)}\right) \\
&= \exp\left(\frac{\epsilon}{2}\right)\left(\frac{\sum_{O' \in \mathbf{O}} \exp\left(\epsilon q(X', O')/2\Delta q\right)}{\sum_{O' \in \mathbf{O}} \exp\left(\epsilon q(X, O')/2\Delta q\right)}\right) \\
&\leq \exp\left(\frac{\epsilon}{2}\right)\left(\frac{\sum_{O' \in \mathbf{O}} \exp\left(\epsilon(\Delta q + q(X, O'))/2\Delta q\right)}{\sum_{O' \in \mathbf{O}} \exp\left(\epsilon q(X, O')/2\Delta q\right)}\right) \\
&= \exp\left(\frac{\epsilon}{2}\right)\left(\frac{\exp(\epsilon/2)\sum_{O' \in \mathbf{O}} \exp\left(\epsilon q(X, O')/2\Delta q\right)}{\sum_{O' \in \mathbf{O}} \exp\left(\epsilon q(X, O')/2\Delta q\right)}\right) \\
&= \exp\left(\frac{\epsilon}{2}\right)\exp\left(\frac{\epsilon}{2}\right) \\
&= \exp(\epsilon)
\end{aligned}$$

Intuitively, a change of one row will change the numerator by at most a factor $\exp(\epsilon\Delta q)$, and the denominator might decrease by a factor $\exp(\epsilon\Delta q)$. Thus, if we did not have the factor $2\Delta q$ in the denominator of Equation 22.17, we would have gotten $\exp(2\epsilon\Delta q)$. Dividing by $2\Delta q$ gives us ϵ-differential privacy. ■

Example 22.9 *Consider the example from before, where our query attempts to find the most common class rank in 15316. Suppose that there are 0 Freshman, 4 Sophomores, 6 Juniors, and 6 Seniors in X. We can compute the exponential mechanism using our scoring function*

$$q(X, O) = |\{x_i \mid x_i = O\}| \tag{22.20}$$

as follows. First note that $\Delta q = 1$, because changing one student's class will change the number of students in any class by exactly 1. Suppose that $\epsilon = \ln(2)$, so that the total "probability mass" that we must draw values proportional to is,

$$\sum_{O \in \{\text{Fr,So,Ju,Se}\}} \exp\left(\frac{\epsilon q(X,O)}{2\Delta q}\right) = \sum_{O \in \{\text{Fr,So,Ju,Se}\}} \exp\left(\frac{\ln(2) q(X,O)}{2}\right) \tag{22.21}$$

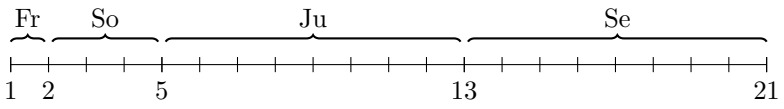
$$= \sum_{O \in \{\text{Fr,So,Ju,Se}\}} 2^{q(X,O)/2} \tag{22.22}$$

$$= 2^0 + 2^2 + 2^3 + 2^3 = 20 \tag{22.23}$$

Then we see that we have the following selection probabilities for each of the outputs:

$$\begin{aligned}
\Pr[\textit{output Fr}] &= 2^0/21 = 1/21 \\
\Pr[\textit{output So}] &= 2^2/21 = 4/21 \\
\Pr[\textit{output Ju}] &= 2^3/21 = 8/21 \\
\Pr[\textit{output Se}] &= 2^3/21 = 8/21
\end{aligned}$$

One simple way of sampling given such a table is to partition the integers in the range $[1, 21]$, an create a bijective mapping between partitions and outputs where partitions are sized according to the probability of their corresponding output. This is visualized in the following diagram.



Could we have implemented this without the exponential mechanism? Looking at our scoring function, it does indeed resemble a count query. Couldn't we just perform four differentially-private counts:

$$\begin{aligned}
\widehat{N}_{\text{Fr}} &= \mathsf{count}(X, \text{Fr}) + \mathsf{Lap}(1/\epsilon) \\
\widehat{N}_{\text{So}} &= \mathsf{count}(X, \text{So}) + \mathsf{Lap}(1/\epsilon) \\
\widehat{N}_{\text{Ju}} &= \mathsf{count}(X, \text{Ju}) + \mathsf{Lap}(1/\epsilon) \\
\widehat{N}_{\text{Se}} &= \mathsf{count}(X, \text{Se}) + \mathsf{Lap}(1/\epsilon)
\end{aligned}$$

Once we have these counts, can we just take the arg max? It turns out that if we want ϵ-differential privacy, we cannot. To understand why, next we'll look at some compositional properties of differential privacy.

22.3 Composing Differentially-Private Computations

When writing programs that apply differential privacy, we'd generally like to confine our attention to a few (hopefully) small parts of the overall program. This simplifies the task of reasoning about and proving their correctness against Definition 22.1, and is consistent with our accepted wisdom of minimizing the trusted computing base. However, if we wish to conclude that the "rest" of the program doesn't violate differential privacy by handling the results of such components incorrectly, then we need properties that specify how the results of differentially-private computations compose with other computations.

Post-processing. The first important property we'll discuss covers post-processing, or computations that are performed that take the result of a differentially-private function as input. Differential privacy enjoys a post-processing guarantee when the post-processor is deterministic, as shown in Theorem 22.10.

Theorem 22.10 *Post-processing. Let* $\mathsf{San} : \mathbf{X} \mapsto \mathbf{O}$ *be a randomized* ϵ*-differentially private function, and* $f : \mathbf{O} \mapsto \mathbf{Y}$ *be any deterministic function. Then* $f \circ \mathsf{San}$ *is* ϵ*-differentially private.*

Proof: Let X_1, X_2 be neighboring databases, and $Y \in \mathbf{Y}$ be any output of f. Let $I \in \mathbf{O}$ be such that $f(I) = Y$. Then,

$$\begin{aligned}\Pr[f(\mathsf{San}(X_1)) = Y] &= \Pr[\mathsf{San}(X_1) = I] \\ &\leq e^{\epsilon}\Pr[\mathsf{San}(X_2) = I] \\ &= e^{\epsilon}\Pr[f(\mathsf{San}(X_2)) = Y]\end{aligned}$$

■

The usefulness of the post-processing theorem is apparent: we can always perform deterministic computations over data produced by ϵ-DP computations, and still arrive at ϵ-DP results. Intuitively, the information content of a signal cannot be increased by local deterministic processing. If the input to f contains no information about an individual, then f cannot add any.

Sequential composition. The next type of composition that we'll consider applies a sequence of functions San_i, each of which provide ϵ_i-differential privacy, and releases the union of their results. We can still obtain a privacy guarantee, but the budgets increase additively.

Theorem 22.11 *Sequential Composition (McSherry 2009). Let* $\mathsf{San}_i : \mathbf{X} \mapsto \mathbf{O}$*,* $1 \leq i \leq n$ *be a sequence of* n *randomized* ϵ_i*-differentially private functions, and let* $\mathsf{San}(X) = (\mathsf{San}_1(X), \ldots, \mathsf{San}_n(X))$*. Then* San *is* $(\sum_{1 \leq i \leq n} \epsilon_i)$*-differentially private.*

Proof: Let $O \in \mathbf{O}^n$ be some value in the range of San, and X_1, X_2 be neighboring databases. Then we can simply calculate:

$$\begin{aligned}\frac{\Pr[\mathsf{San}(X_1) = O]}{\Pr[\mathsf{San}(X_2) = O]} &= \frac{\prod_{1 \leq i \leq n} \Pr[\mathsf{San}_i(X_1) = O]}{\prod_{1 \leq i \leq n} \Pr[\mathsf{San}_i(X_2) = O]} \\ &= \left(\frac{\Pr[\mathsf{San}_1(X_1) = O]}{\Pr[\mathsf{San}_1(X_2) = O]}\right) \cdots \left(\frac{\Pr[\mathsf{San}_n(X_1) = O]}{\Pr[\mathsf{San}_n(X_2) = O]}\right) \\ &\leq e^{\epsilon_1} \cdots e^{\epsilon_n} \\ &= e^{\epsilon_1 + \cdots + \epsilon_n}\end{aligned}$$

■

The sequential composition theorem is crucial for any practical system that hopes to achieve differential privacy. Notably, it implies that for a fixed privacy budget ϵ, it isn't safe to apply a differentially-private computation an arbitrary number of times to the same database X. If the total sum of the computations' budgets exceeds ϵ, then the composed computation is no longer ϵ-differentially private. If we want to ensure a certain level of privacy in a computation composed of multiple queries, then we need to carefully account for the amount of privacy budget that is "consumed" by each query. If the amount ever exceeds our budget, then we can never answer another query from that database.

Parallel composition. The last form of composition that we'll look at is targeted towards the use of multiple differentially-private queries over **disjoint** partitions of X.

Theorem 22.12 *Parallel Composition (McSherry 2009). Let* $\mathsf{San}_i : \mathbf{X} \mapsto \mathbf{O}$, $1 \leq i \leq n$ *be a sequence of* n *randomized* ϵ_i*-differentially private functions,* $P_1, \ldots, P_n$ *be disjoint subsets of* X*, and let* $\mathsf{San}(X) = (\mathsf{San}_1(P_1), \ldots, \mathsf{San}_n(P_n))$*. Then* San *is* $(\max_{1 \leq i \leq n} \epsilon_i)$*-differentially private.*

Proof: Let $O \in \mathbf{O}^n$ be some value in the range of San, and X_1, X_2 be neighboring databases. Let $P_{1,1}, \ldots, P_{1,n}$ be the disjoint subsets of X_1, and $P_{2,1}, \ldots, P_{2,n}$ similarly for X_2. Observe that there is only one partition index, i, at which $P_{1,i}$ and $P_{2,i}$ differ. Assume without loss of generality that the index is 1. Then:

$$\begin{aligned}
\frac{\Pr[\mathsf{San}(X_1) = O]}{\Pr[\mathsf{San}(X_2) = O]} &= \frac{\prod_{1 \leq i \leq n} \Pr[\mathsf{San}_i(P_{1,i}) = O]}{\prod_{1 \leq i \leq n} \Pr[\mathsf{San}_i(P_{2,i}) = O]} \\
&= \frac{\Pr[\mathsf{San}_1(X_1) = O]}{\Pr[\mathsf{San}_1(X_2) = O]} \\
&\leq \max_{1 \leq i \leq n} \epsilon_i
\end{aligned}$$

The second line follows because we assumed that the differing row occured in the first partition, so:

$$\prod_{2 \leq i \leq n} \Pr[\mathsf{San}_i(P_{1,i}) = O] = \prod_{2 \leq i \leq n} \Pr[\mathsf{San}_i(P_{2,i}) = O]$$

The last line follows under the pessimistic assumption that the largest ϵ_i applies to the first partition. ■

Parallel composition is probably not as widely applicable as the sequential form, but can be very useful in certain cases because it does not expend privacy budget additively. Whenever computations can be broken into smaller sub-computations over disjoint data, applying the parallel composition theorem followed by post-processing can lead to strong utility for a fixed privacy budget.