THEORY OF COMPUTING, Volume 14 (8), 2018, pp. 1–35
www.theoryofcomputing.org

# The Complexity of Computing the Optimal Composition of Differential Privacy

Jack Murtagh* Salil Vadhan†

Received Month 1, 2012; Revised July 29, 2017; Published June 2, 2018

**Abstract:** In the study of differential privacy, composition theorems (starting with the original paper of Dwork, McSherry, Nissim, and Smith (TCC'06)) bound the degradation of privacy when composing several differentially private algorithms. Kairouz, Oh, and Viswanath (ICML'15) showed how to compute the optimal bound for composing $k$ arbitrary $(\varepsilon, \delta)$-differentially private algorithms. We characterize the optimal composition for the more general case of $k$ arbitrary $(\varepsilon_1, \delta_1), \ldots, (\varepsilon_k, \delta_k)$-differentially private algorithms where the privacy parameters may differ for each algorithm in the composition. We show that computing the optimal composition in general is $\#\mathsf{P}$-complete. Since computing optimal composition exactly is infeasible (unless $\mathsf{FP} = \#\mathsf{P}$), we give an approximation algorithm that computes the composition to arbitrary accuracy in polynomial time. The algorithm is a modification of Dyer's dynamic programming approach to approximately counting solutions to knapsack problems (STOC'03).

A conference version of this paper appeared in the Proceedings of the 13th IACR Theory of Cryptography Conference (TCC), 2016-A [14].

*Supported by NSF grant CNS-1237235 and a grant from the Sloan Foundation.
†Supported by NSF grant CNS-1237235, a grant from the Sloan Foundation, and a Simons Investigator Award.

**ACM Classification:** F.2

**AMS Classification:** 68Q17, 68W25, 68Q25

**Key words and phrases:** complexity theory, approximation algorithms, differential privacy, composition

 DOI: 10.4086/toc.2018.v014a008

# 1 Introduction

Differential privacy is a framework that allows statistical analysis of private databases while minimizing the risks to individuals in the databases. The idea is that an individual should be relatively unaffected whether he or she decides to join or opt out of a research dataset. More specifically, the probability distribution of outputs of a statistical analysis of a database should be nearly identical to the distribution of outputs on the same database with a single person's data removed. Here the probability space is over the coin flips of the randomized differentially private algorithm that handles the queries. To formalize this, we call two databases $D_0, D_1$ with $n$ rows each *neighboring* if they are identical on at least $n-1$ rows, and define differential privacy as follows.

**Definition 1.1** (Differential Privacy [5, 4])**.** A randomized algorithm $M$ is $(\varepsilon, \delta)$*-differentially private* for $\varepsilon, \delta \geq 0$ if for all pairs of neighboring databases $D_0$ and $D_1$ and all output sets $S \subseteq \mathrm{Range}(M)$

$$\Pr[M(D_0) \in S] \leq \mathrm{e}^{\varepsilon} \Pr[M(D_1) \in S] + \delta$$

where the probabilities are over the coin flips of the algorithm $M$.

In the practice of differential privacy, we generally think of $\varepsilon$ as a small, non-negligible, constant (e. g., $\varepsilon = .1$). We view $\delta$ as a "security parameter" that is cryptographically small (e. g., $\delta = 2^{-30}$). One of the important properties of differential privacy is that if we run multiple distinct differentially private algorithms on the same database, the resulting composed algorithm is also differentially private, albeit with some degradation in the privacy parameters $(\varepsilon, \delta)$. In this paper, we are interested in quantifying the degradation of privacy under composition. We will denote the composition of $k$ differentially private algorithms $M_1, M_2, \ldots, M_k$ as $(M_1, M_2, \ldots, M_k)$ where

$$(M_1, M_2, \ldots, M_k)(x) = (M_1(x), M_2(x), \ldots, M_k(x)).$$

A handful of composition theorems already exist in the literature. The first basic result is the following.

**Theorem 1.2** (Basic Composition [4])**.** *For every $\varepsilon \geq 0$, $\delta \in [0,1]$, and $(\varepsilon, \delta)$-differentially private algorithms $M_1, M_2, \ldots, M_k$, the composition $(M_1, M_2, \ldots, M_k)$ satisfies $(k\varepsilon, k\delta)$-differential privacy.*

This tells us that under composition, the privacy parameters of the individual algorithms "sum up," so to speak. We care about understanding composition because in practice we rarely want to release only a single statistic about a dataset. Releasing many statistics may require running multiple differentially private algorithms on the same database. Composition is also a very useful tool in algorithm design. Often, new differentially private algorithms are created by combining several simpler algorithms. Composition theorems help us analyze the privacy properties of algorithms designed in this way.

Theorem 1.2 shows a linear degradation in global privacy as the number of algorithms in the composition ($k$) grows and it is of interest to improve on this bound. If we can prove that privacy degrades more slowly under composition, we can get more utility out of our algorithms under the same global privacy guarantees. Dwork, Rothblum, and Vadhan gave the following improvement on the basic summing composition above [7].

**Theorem 1.3** (Advanced Composition [7])**.** *For every $\varepsilon > 0, \delta, \delta' > 0$, $k \in \mathbb{N}$, and $(\varepsilon, \delta)$-differentially private algorithms $M_1, M_2, \ldots, M_k$, the composition $(M_1, M_2, \ldots, M_k)$ satisfies $(\varepsilon_g, k\delta + \delta')$-differential privacy for*

$$\varepsilon_g = \sqrt{2k\ln(1/\delta')} \cdot \varepsilon + k \cdot \varepsilon \cdot (\mathrm{e}^{\varepsilon} - 1)\,.$$

Theorem 1.3 shows that privacy under composition degrades by a function of $O(\sqrt{k\ln(1/\delta')})$ which is an improvement if $\delta' = 2^{-O(k)}$. It can be shown that a degradation function of $\Omega(\sqrt{k\ln(1/\delta)})$ is necessary even for the simplest differentially private algorithms, such as randomized response [15].

Despite giving an asymptotically correct upper bound for the global privacy parameter, $\varepsilon_g$, Theorem 1.3 is not exact. We want an exact characterization because, beyond being theoretically interesting, constant factors in composition theorems can make a substantial difference in the practice of differential privacy. Furthermore, Theorem 1.3 only applies to "homogeneous" composition where each individual algorithm has the same pair of privacy parameters, $(\varepsilon, \delta)$. In practice we often want to analyze the more general case where some individual algorithms in the composition may offer more or less privacy than others. That is, given algorithms $M_1, M_2, \ldots, M_k$, we want to compute the best achievable privacy parameters for $(M_1, M_2, \ldots, M_k)$. Formally, we want to compute the following function.

$$\mathrm{OptComp}(M_1, M_2, \ldots, M_k, \delta_g) = \inf\{\varepsilon_g \geq 0 \colon (M_1, M_2, \ldots, M_k) \text{ is } (\varepsilon_g, \delta_g)\text{-DP}\}\,.$$

It is convenient for us to view $\delta_g$ as given and then compute the best $\varepsilon_g$, but the dual formulation, viewing $\varepsilon_g$ as given, is equivalent (by binary search). Actually, we want a function that depends only on the privacy parameters of the individual algorithms,

$$\begin{aligned}&\mathrm{OptComp}((\varepsilon_1, \delta_1), (\varepsilon_2, \delta_2), \ldots, (\varepsilon_k, \delta_k), \delta_g) \\ &\qquad = \sup\{\mathrm{OptComp}(M_1, M_2, \ldots, M_k, \delta_g) \colon M_i \text{ is } (\varepsilon_i, \delta_i)\text{-DP } \forall i \in [k]\}. \qquad (1.1)\end{aligned}$$

In other words we want OptComp to give us the minimum possible $\varepsilon_g$ that maintains privacy for every sequence of algorithms with the given privacy parameters $(\varepsilon_i, \delta_i)$. A result from Kairouz, Oh, and Viswanath [12] characterizes OptComp for the homogeneous case.

**Theorem 1.4** (Optimal Homogeneous Composition [12][1])**.** *For every $\varepsilon \geq 0$ and $\delta \in [0, 1)$,*

$$\mathrm{OptComp}(\underbrace{(\varepsilon, \delta), (\varepsilon, \delta), \ldots, (\varepsilon, \delta)}_{k}, \delta_g)$$

*equals the least value of $\varepsilon_g \geq 0$ such that*

$$\frac{1}{(1+\mathrm{e}^{\varepsilon})^k} \sum_{\ell = \left\lceil \frac{\varepsilon_g + k\varepsilon}{2\varepsilon} \right\rceil}^{k} \binom{k}{\ell} \left(\mathrm{e}^{\ell\varepsilon} - \mathrm{e}^{\varepsilon_g} \mathrm{e}^{(k-\ell)\varepsilon}\right) \leq 1 - \frac{1-\delta_g}{(1-\delta)^k}\,.$$

[1]The phrasing of Theorem 1.4 is not exactly how it is presented in [12] (which only refers to $\varepsilon_g$ of the form $(k-2i)\varepsilon$ for integer $i$), but this version can be deduced from the original.

Empirically (see Section 6), this optimal bound provides a 30-40% savings in $\varepsilon_g$ compared to Theorem 1.3 (and a 20% savings compared to an improved asymptotic bound from [12]). The problem remains to find the optimal composition behavior for the more general heterogeneous case. Kairouz, Oh, and Viswanath also provide an upper bound for heterogeneous composition that generalizes the $O(\sqrt{k\ln(1/\delta')})$ degradation found in Theorem 1.3 for homogeneous composition but do not comment on how close it is to optimal.

## 1.1 Our results

We begin by extending the results of Kairouz, Oh, and Viswanath [12] to the general heterogeneous case.

**Theorem 1.5** (Optimal Heterogeneous Composition)**.** *For all $\varepsilon_1,\ldots,\varepsilon_k \geq 0$ and $\delta_1,\ldots,\delta_k,\delta_g \in [0,1)$, $\mathrm{OptComp}((\varepsilon_1,\delta_1),(\varepsilon_2,\delta_2),\ldots,(\varepsilon_k,\delta_k),\delta_g)$ equals the least value of $\varepsilon_g \geq 0$ such that*

$$\frac{1}{\prod_{i=1}^{k}(1+\mathrm{e}^{\varepsilon_i})}\sum_{S\subseteq\{1,\ldots,k\}}\max\left\{\mathrm{e}^{\sum\limits_{i\in S}\varepsilon_i}-\mathrm{e}^{\varepsilon_g}\cdot\mathrm{e}^{\sum\limits_{i\notin S}\varepsilon_i},0\right\}\leq 1-\frac{1-\delta_g}{\prod_{i=1}^{k}(1-\delta_i)}\,. \tag{1.2}$$

Theorem 1.5 exactly characterizes the optimal composition behavior for any arbitrary set of differentially private algorithms. It also shows that optimal composition can be computed in time exponential in $k$ by computing the sum over $S\subseteq\{1,\ldots,k\}$ by brute force. Of course in practice an exponential-time algorithm is not satisfactory for large $k$. Our next result shows that this exponential complexity is necessary.

**Theorem 1.6.** *Computing* OptComp *is* #P*-complete, even on instances where $\delta_1=\delta_2=\cdots=\delta_k=0$ and $\sum_{i\in[k]}\varepsilon_i\leq\varepsilon$ for any desired constant $\varepsilon>0$.*

Recall that #P is the class of counting problems associated with decision problems in NP. So being #P-complete means that there is no polynomial-time algorithm for OptComp unless there is a polynomial-time algorithm for counting the number of satisfying assignments of Boolean formulas (or equivalently for counting the number of solutions of all NP problems). So there is almost certainly no efficient algorithm for OptComp and therefore no analytic solution. Despite the intractability of exact computation, we show that OptComp can be approximated efficiently.

**Theorem 1.7.** *There is a polynomial-time algorithm that given rational $\varepsilon_1,\ldots,\varepsilon_k\geq 0,\delta_1,\ldots\delta_k,\delta_g\in[0,1)$, and $\eta\in(0,1)$, outputs $\varepsilon^*$ satisfying*

$$\mathrm{OptComp}((\varepsilon_1,\delta_1),\ldots,(\varepsilon_k,\delta_k),\delta_g)\leq\varepsilon^*\leq\mathrm{OptComp}((\varepsilon_1,\delta_1),\ldots,(\varepsilon_k,\delta_k),\mathrm{e}^{-\eta/2}\cdot\delta_g)+\eta\,.$$

*The algorithm runs in time*

$$O\left(\frac{k^3\cdot\overline{\varepsilon}\cdot(1+\overline{\varepsilon})}{\eta}\cdot\log\left(\frac{k^2\cdot\overline{\varepsilon}\cdot(1+\overline{\varepsilon})}{\eta}\right)\right)$$

*where $\overline{\varepsilon}=\sum_{i\in[k]}\varepsilon_i/k$, assuming constant-time arithmetic operations.*

Note that we incur a relative error of $\eta$ in approximating $\delta_g$ *and* an additive error of $\eta$ in approximating $\varepsilon_g$. Since we always take $\varepsilon_g$ to be non-negligible or even constant, we get a very good approximation when $\eta$ is polynomially small or even a constant. Thus, it is acceptable that the running time is polynomial in $1/\eta$.

In addition to the results listed above, our proof of Theorem 1.5 also provides a somewhat simpler proof of the Kairouz-Oh-Viswanath homogeneous composition theorem (Theorem 1.4 [12]). The proof in [12] introduces a view of differential privacy through the lens of hypothesis testing and uses geometric arguments. Our proof relies only on elementary techniques commonly found in the differential privacy literature.

**Practical application.** The theoretical results presented here were motivated by our work on an applied project called "Privacy Tools for Sharing Research Data"[2] [10]. We are building a system that will allow researchers with sensitive datasets to make differentially private statistics about their data available through data repositories using the Dataverse[3] platform [3, 13]. Part of this system is a tool that helps both data depositors and data analysts distribute a global privacy budget across many statistics. Users select which statistics they would like to compute and are given estimates of how accurately each statistic can be computed. They can also redistribute their privacy budget according to which statistics they think are most valuable in their dataset. We implemented the approximation algorithm from Theorem 1.7 and integrated it with this tool to ensure that users get the most utility out of their privacy budget.

**Utility-theoretic interpretation.** As suggested to us by an anonymous referee, another natural perspective on composition is to maximize the "utility" $u(M_1, \ldots, M_k)$ provided by $k$ differentially private algorithms for a particular set of analysts subject to a global privacy constraint, $\mathrm{OptComp}(M_1, \ldots, M_k, \delta_g) \leq \varepsilon_g$. Our results can be interpreted as studying this problem in the special case where $u(M_1, \ldots, M_k) = 1$ if and only if for every $i \in [k]$, $M_i$ is a "randomized response" algorithm (see Definition 3.1) with privacy parameters *at least* $\varepsilon_i, \delta_i$ (recall that larger privacy parameters generally yield greater accuracy and hence greater utility). Following [12], we show in Lemma 3.2 that randomized response achieves the worst-case bound $\mathrm{OptComp}((\varepsilon_1, \delta_1), \ldots, (\varepsilon_k, \delta_k), \delta_g)$ over all sequences of algorithms $M_i$ that are $(\varepsilon_i, \delta_i)$-DP. Studying this utility maximization problem in greater generality is an interesting direction for future work. (See [1, 11] for a general study of utility maximization under one-shot differential privacy, without composition.)

## 2 Technical preliminaries

A useful notation for thinking about differential privacy is defined below.

**Definition 2.1.** For two discrete random variables $Y$ and $Z$ taking values in the same output space $S$, the $\delta$*-approximate max-divergence* of $Y$ and $Z$ is defined as follows.

$$D_\infty^\delta(Y \| Z) \equiv \max_S \left[ \ln \frac{\Pr[Y \in S] - \delta}{\Pr[Z \in S]} \right].$$

[2] https://privacytools.seas.harvard.edu/
[3] https://dataverse.org/

Notice that an algorithm $M$ is $(\varepsilon,\delta)$ differentially private if and only if for all pairs of neighboring databases, $D_0, D_1$, we have $D_\infty^\delta(M(D_0)\|M(D_1)) \le \varepsilon$. The standard fact that differential privacy is closed under "post processing" [5, 6] now can be formulated as follows.

**Fact 2.2.** *If $f\colon S \to R$ is any randomized function, then*

$$D_\infty^\delta(f(Y)\|f(Z)) \le D_\infty^\delta(Y\|Z)\,.$$

**Adaptive composition.** The composition results in our paper actually hold for a more general model of composition than the one described in the introduction. The model is called adaptive composition and was formalized in [7]. We generalize their formulation to the heterogeneous setting where privacy parameters may differ across different algorithms in the composition.

The idea is that instead of running $k$ differentially private algorithms chosen all at once on a single database, we can imagine an adversary adaptively engaging in a "composition game." The game takes as input a bit $b \in \{0,1\}$ and privacy parameters $(\varepsilon_1,\delta_1),\ldots,(\varepsilon_k,\delta_k)$. A randomized adversary $A$, tries to learn $b$ through $k$ rounds of interaction as follows. On the $i$-th round of the game, $A$ chooses an $(\varepsilon_i,\delta_i)$-differentially private algorithm $M_i$ and two neighboring databases $D_{(i,0)}, D_{(i,1)}$. $A$ then receives an output $y_i = M_i(D_{(i,b)})$ where the internal randomness of $M_i$ is independent of the internal randomness of $M_1,\ldots,M_{i-1}$. The choices of $M_i, D_{(i,0)}$, and $D_{(i,1)}$ may depend on $y_0,\ldots,y_{i-1}$ as well as the adversary's own randomness.

The outcome of this game is called the *view of the adversary*, $V^b$ which is defined to be $(y_1,\ldots,y_k)$ along with $A$'s coin tosses. The algorithms $M_i$ and databases $D_{(i,0)}, D_{(i,1)}$ from each round can be reconstructed from $V^b$. Now we can formally define privacy guarantees under adaptive composition.

**Definition 2.3.** We say that the sequences of privacy parameters $\varepsilon_1,\ldots,\varepsilon_k \ge 0$, $\delta_1,\ldots,\delta_k \in [0,1)$ satisfy $(\varepsilon_g,\delta_g)$*-differential privacy under adaptive composition* if for every adversary $A$ we have

$$D_\infty^{\delta_g}(V^0\|V^1) \le \varepsilon_g\,,$$

where $V^b$ represents the view of $A$ in composition game $b$ with privacy parameter inputs

$$(\varepsilon_1,\delta_1),\ldots,(\varepsilon_k,\delta_k)\,.$$

**Computing real-valued functions.** Many of the computations we discuss involve irrational numbers and we need to be explicit about how we model such computations on finite, discrete machines. Namely when we talk about computing a function $f:\{0,1\}^* \to \mathbb{R}$, what we really mean is computing $f$ to any desired number $q$ bits of precision. More precisely, given $x,q$, the task is to compute a number $y \in \mathbb{Q}$ such that $|f(x)-y| \le 1/2^q$. We measure the complexity of algorithms for this task as a function of $|x|+q$. In order to reason about the complexity of OptComp, we will also require that the inputs be rational. So when we talk about computing OptComp exactly, we actually mean given $\varepsilon_1,\ldots,\varepsilon_k \ge 0, \delta_1,\ldots,\delta_k,\delta_g \in [0,1)$ all rational and an integer $q$, compute $\varepsilon^*$ such that

$$|\varepsilon_g - \varepsilon^*| \le \frac{1}{2^q}$$

where $\varepsilon_g$ is the true optimal parameter with full precision.

# 3 Characterization of OptComp

Following [12], we show that to analyze the composition of arbitrary $(\varepsilon_i, \delta_i)$-DP algorithms, it suffices to analyze the composition of the following simple variant of randomized response [15].

**Definition 3.1** ([12]). Define a randomized algorithm $\tilde{M}_{(\varepsilon,\delta)} \colon \{0,1\} \to \{0,1,2,3\}$ as follows, setting $\alpha = 1-\delta$.

$$\begin{array}{ll}
\Pr[\tilde{M}_{(\varepsilon,\delta)}(0)=0] = \delta\,, & \Pr[\tilde{M}_{(\varepsilon,\delta)}(1)=0] = 0\,,\\
\Pr[\tilde{M}_{(\varepsilon,\delta)}(0)=1] = \alpha \cdot \frac{\mathrm{e}^{\varepsilon}}{1+\mathrm{e}^{\varepsilon}}\,, & \Pr[\tilde{M}_{(\varepsilon,\delta)}(1)=1] = \alpha \cdot \frac{1}{1+\mathrm{e}^{\varepsilon}}\,,\\
\Pr[\tilde{M}_{(\varepsilon,\delta)}(0)=2] = \alpha \cdot \frac{1}{1+\mathrm{e}^{\varepsilon}}\,, & \Pr[\tilde{M}_{(\varepsilon,\delta)}(1)=2] = \alpha \cdot \frac{\mathrm{e}^{\varepsilon}}{1+\mathrm{e}^{\varepsilon}}\,,\\
\Pr[\tilde{M}_{(\varepsilon,\delta)}(0)=3] = 0\,, & \Pr[\tilde{M}_{(\varepsilon,\delta)}(1)=3] = \delta\,.
\end{array}$$

Note that $\tilde{M}_{(\varepsilon,\delta)}$ is in fact $(\varepsilon,\delta)$-DP. Kairouz, Oh, and Viswanath showed that $\tilde{M}_{(\varepsilon,\delta)}$ can be used to simulate the output of every $(\varepsilon,\delta)$-DP algorithm on adjacent databases.

**Lemma 3.2** ([12]). *For every $(\varepsilon,\delta)$-DP algorithm $M$ and neighboring databases $D_0, D_1$, there exists a randomized algorithm $T$ such that $T(\tilde{M}_{(\varepsilon,\delta)}(b))$ is identically distributed to $M(D_b)$ for $b=0$ and $b=1$.*

For the sake of completeness, we show how this lemma can be deduced from a recent characterization, due to Bun and Steinke [2], of approximate max-divergence as equivalent to exact max-divergence conditioned on events of probability $1-\delta$.

**Lemma 3.3** ([2]). *Let $X$ and $Y$ be random variables. The following are equivalent.*

1. $D_\infty^\delta(X\|Y) \le \varepsilon$ *and* $D_\infty^\delta(Y\|X) \le \varepsilon$.
2. *There are probabilistic events $E = E(X)$ and $F = F(Y)$ such that*
$$\Pr[E] = \Pr[F] = 1-\delta \quad \textit{and} \quad D_\infty^0(X|_E \parallel Y|_F) \le \varepsilon \quad \textit{and} \quad D_\infty^0(Y|_F \parallel X|_E) \le \varepsilon\,.$$
3. *There are probabilistic events $E = E(X)$ and $F = F(Y)$ such that*
$$\Pr[E] = \Pr[F] \ge 1-\delta \quad \textit{and} \quad D_\infty^0(X|_E \parallel Y|_F) \le \varepsilon \quad \textit{and} \quad D_\infty^0(Y|_F \parallel X|_E) \le \varepsilon\,.$$

Bun and Steinke originally proved Lemma 3.3 using Lemma 3.2, but we give a direct proof here that is in a similar spirit to the proof of an alternate characterization of approximate max-divergence from [7] (which uses statistical distance rather than conditioning on high-probability events). This avoids the need for the hypothesis testing and geometric arguments used in [12] to establish their optimal composition theorem. In an earlier version of our paper, we included a direct proof of Lemma 3.2, without going through Lemma 3.3, but that proof was rather long and tedious, although using similar ideas to the proofs below.

*Proof of Lemma 3.3.* It is immediate that statement 2 implies statement 3. To see that statement 3 implies statement 1, note that assuming statement 3, for every set $S$ we have

$$\Pr[X \in S] \le \Pr[X \in S|E]\cdot\Pr[E] + \Pr[\neg E] \le (\mathrm{e}^{\varepsilon}\cdot\Pr[Y \in S|F])\cdot\Pr[F] + \Pr[\neg F] \le \mathrm{e}^{\varepsilon}\cdot\Pr[Y\in S] + \delta\,.$$

Thus we have $D^{\delta}_{\infty}(X\|Y) \le \varepsilon$ and by symmetry we also have $D^{\delta}_{\infty}(Y\|X) \le \varepsilon$.

It remains to show that statement 1 implies statement 2, so assume that statement 1 holds. Finding events $E$ and $F$ as in statement 2 is equivalent to finding functions $e$ and $f$ such that the following hold.

1. For all $x$, $0 \le e(x) \le \Pr[X = x]$ and $0 \le f(x) \le \Pr[Y = x]$.

2. For all $x$, $e(x) \le \mathrm{e}^{\varepsilon} \cdot f(x)$ and $f(x) \le \mathrm{e}^{\varepsilon} \cdot e(x)$.

3. $\sum_x e(x) = \sum_x f(x) = 1 - \delta$.

Indeed, the corresponding events $E$ and $F$ are defined by

$$\Pr[E \mid X = x] = e(x)/\Pr[X = x] \qquad \text{and} \qquad \Pr[F \mid Y = x] = f(x)/\Pr[Y = x]\,.$$

Then by Bayes' Rule, the conditions above imply that

$$\Pr[X = x|E] = \frac{\Pr[E|X = x] \cdot \Pr[X = x]}{\Pr[E]} = \frac{e(x)}{\sum_y e(y)} \le \frac{\mathrm{e}^{\varepsilon} \cdot f(x)}{\sum_y f(y)} = \mathrm{e}^{\varepsilon} \cdot \Pr[Y = x|F]\,.$$

Thus we have $D^{0}_{\infty}(X|_E \parallel Y|_F) \le \varepsilon$ and by symmetry we also have $D^{0}_{\infty}(Y|_F \parallel X|_E) \le \varepsilon$.

We now proceed to defining the functions $e$ and $f$. Define the following disjoint subsets of the domain.

$$\begin{aligned} S &= \{x : \Pr[X = x] > \mathrm{e}^{\varepsilon} \cdot \Pr[Y = x]\}\,, \\ T &= \{x : \Pr[Y = x] > \mathrm{e}^{\varepsilon} \cdot \Pr[X = x]\}\,. \end{aligned}$$

Let

$$\alpha = \Pr[X \in S] - \mathrm{e}^{\varepsilon} \cdot \Pr[Y \in S] \ge 0 \quad \text{and} \quad \beta = \Pr[Y \in T] - \mathrm{e}^{\varepsilon} \cdot \Pr[X \in T] \ge 0\,.$$

We know that $\alpha, \beta \le \delta$, because $D^{\delta}_{\infty}(X\|Y) \le \varepsilon$ and $D^{\delta}_{\infty}(Y\|X) \le \varepsilon$. We start by trying to define the functions $e$ and $f$ as follows.

1. For $x \in S$, $e(x) = \mathrm{e}^{\varepsilon} \cdot \Pr[Y = x]$ and for $x \notin S$, $e(x) = \Pr[X = x]$.

2. For $x \in T$, $f(x) = \mathrm{e}^{\varepsilon} \cdot \Pr[X = x]$ and for $x \notin T$, $f(x) = \Pr[Y = x]$.

Figure 1 shows the functions $e$ and $f$ as defined above as well as the sets $S$ and $T$ and regions whose areas are $\alpha$ and $\beta$. Notice that we have satisfied the conditions

$$\begin{aligned} 0 \le e(x) &\le \Pr[X = x]\,, \\ 0 \le f(x) &\le \Pr[Y = x]\,, \\ e(x) &\le \mathrm{e}^{\varepsilon} \cdot f(x)\,, \qquad \text{and} \\ f(x) &\le \mathrm{e}^{\varepsilon} \cdot e(x)\,. \end{aligned}$$

It is also clear from the figure that $\sum_x e(x) = 1 - \alpha$ and $\sum_x f(x) = 1 - \beta$. Indeed, when we sum $e(x)$ and

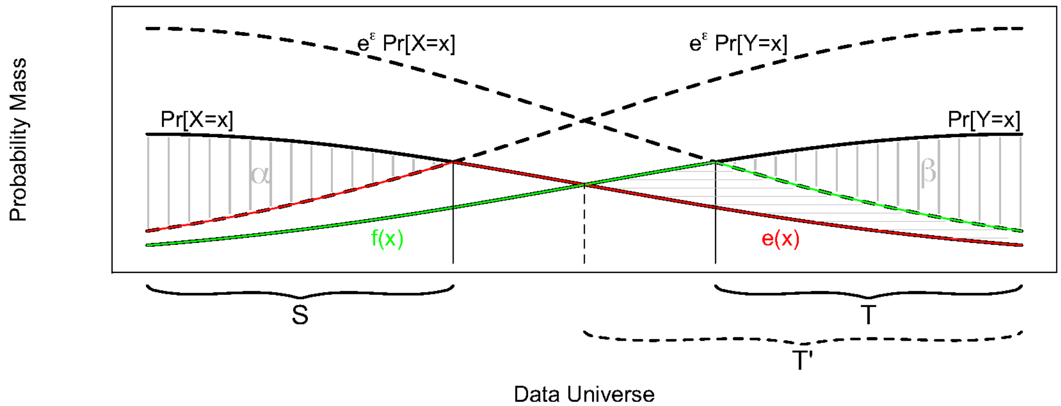


Figure 1: Depiction of the functions $e(x)$ (red) and $f(x)$ (green). The solid black curves are the probability mass functions of $X$ and $Y$ and the dashed curves are the solid curves scaled by a factor of $\mathrm{e}^{\varepsilon}$. The regions with areas $\alpha$ and $\beta$ are shaded with vertical lines and the region above $e(x)$ and below $f(x)$ is shaded with horizontal lines. The sets $S$, $T$, and $T'$ from the proof are also depicted.

$f(x)$ over all $x$, we obtain the following.

$$\begin{aligned}
\sum_x e(x) &= \sum_{x\in S} e(x) + \sum_{x\notin S} e(x) \\
&= \sum_{x\in S} \mathrm{e}^{\varepsilon}\cdot \Pr[Y=x] + \sum_{x\notin S} \Pr[X=x] \\
&= \mathrm{e}^{\varepsilon}\cdot \Pr[Y\in S] + \Pr[X\notin S] \\
&= \Pr[X\in S] - \alpha + \Pr[X\notin S] \\
&= 1-\alpha\,.
\end{aligned}$$

Similarly, we have

$$\sum_x f(x) = 1-\beta.$$

If $\alpha=\beta=\delta$, then we're done. Otherwise, we will modify $e$ and $f$ to ensure that $\alpha=\beta$ and then we will modify them again to achieve $\alpha=\beta=\delta$. Suppose without loss of generality that $\alpha>\beta$. Then we will reduce the function $f$ on the set $T'=\{x:\Pr[Y=x]>\Pr[X=x]\}\supseteq T$ (see Figure 1) to reduce the sum $\sum_x f(x)$ by $\alpha-\beta$ (while maintaining all the other conditions). The only condition that might be violated if we reduce the function $f$ is the one that $e(x)\le \mathrm{e}^{\varepsilon}\cdot f(x)$. So we just need to confirm that

$$\sum_{x\in T'} \left(f(x) - \mathrm{e}^{-\varepsilon}\cdot e(x)\right)$$

is at least $\alpha-\beta$ to be able to reduce $f$ as much as we need. We have

$$\sum_{x\in T'}(f(x)-\mathrm{e}^{-\varepsilon}\cdot e(x))\geq\sum_{x\in T'}(f(x)-e(x))\,,$$

so it suffices to show that the area of the region that is above $e(x)$ and below $f(x)$ is at least $\alpha-\beta$. To see this, we make the following observations.

1. The area of the region that is above $\Pr[X=x]$ and below $\Pr[Y=x]$ equals the area of the region that is above $\Pr[Y=x]$ and below $\Pr[X=x]$.

$$\begin{aligned}\sum_{x\in T'}(\Pr[Y=x]-\Pr[X=x]) &= \Pr[Y\in T']-\Pr[X\in T'] \\ &= (1-\Pr[Y\notin T'])-(1-\Pr[X\notin T']) \\ &= \sum_{x\notin T'}(\Pr[X=x]-\Pr[Y=x])\,.\end{aligned}$$

(This quantity is simply the total variation distance between $X$ and $Y$.)

2. The area of the region that is above $\Pr[X=x]$ and below $\Pr[Y=x]$ equals $\beta$ plus the area of the region that is above $e(x)$ and below $f(x)$.

$$\begin{aligned}\sum_{x\in T'}(\Pr[Y=x]-\Pr[X=x]) &= \sum_{x\in T'}(\Pr[Y=x]-f(x))+\sum_{x\in T'}(f(x)-\Pr[X=x]) \\ &= \sum_{x\in T}(\Pr[Y=x]-f(x))+\sum_{x\in T'}(f(x)-e(x)) \\ &= \beta+\sum_{x\in T'}(f(x)-e(x))\,.\end{aligned}$$

3. The area of the region that is above $\Pr[Y=x]$ and below $\Pr[X=x]$ is at least $\alpha$.

$$\begin{aligned}\sum_{x\notin T'}(\Pr[X=x]-\Pr[Y=x]) &\geq \sum_{x\in S}(\Pr[X=x]-\mathrm{e}^{\varepsilon}\cdot\Pr[Y=x] \\ &= \alpha\,.\end{aligned}$$

Putting it all together, we have

$$\begin{aligned}\sum_{x\in T'}(f(x)-\mathrm{e}^{-\varepsilon}\cdot e(x)) &\geq \sum_{x\in T'}(f(x)-e(x)) \\ &= \sum_{x\in T'}(\Pr[Y=x]-\Pr[X=x])-\beta \\ &\geq \alpha-\beta\,.\end{aligned}$$

So we can afford to reduce the sum over all $x$ of $f(x)$ by $\alpha-\beta$ without violating the other conditions and thus have found two functions $e$ and $f$ such that

1. for all $x$, $0\leq e(x)\leq\Pr[X=x]$ and $0\leq f(x)\leq\Pr[Y=x]$;

2. for all $x$, $e(x) \leq \mathrm{e}^{\varepsilon} \cdot f(x)$ and $f(x) \leq \mathrm{e}^{\varepsilon} \cdot e(x)$;

3. $\sum_x e(x) = \sum_x f(x) \geq 1-\delta$.

Suppose $\sum_x e(x) = \sum_x f(x) = 1-\delta_0 > 1-\delta$. Then we can scale them both by a multiplicative factor of $(1-\delta_0)/(1-\delta)$ and achieve all the desired conditions. □

Now we can prove Lemma 3.2 using Lemma 3.3.

*Proof of Lemma 3.2.* Let $M$ be an $(\varepsilon,\delta)$-DP algorithm and let $D_0, D_1$ be neighboring databases. Let $X \sim M(D_0)$ and $Y \sim M(D_1)$ be random variables and note that

$$D_\infty^\delta(X\|Y) \leq \varepsilon \qquad \text{and} \qquad D_\infty^\delta(Y\|X) \leq \varepsilon$$

because $M$ is $(\varepsilon,\delta)$-DP. Lemma 3.3 says that there exist events $E,F$ such that

$$D_\infty^0(X|_E \parallel Y|_F) \leq \varepsilon\,, \qquad D_\infty^0(Y|_F \parallel X|_E) \leq \varepsilon\,, \qquad \text{and} \qquad \Pr[E] = \Pr[F] = 1-\delta\,.$$

Let $R$ be the output space of $M$ and fix $r \in R$. We define the simulating mechanism $T\colon \{0,1,2,3\} \to R$ as follows.

$$\begin{aligned}
\Pr[T(0)=r] &= \Pr[M(D_0)=r|\neg E]\,,\\
\Pr[T(1)=r] &= \frac{1}{\mathrm{e}^{\varepsilon}-1}\cdot\left(\mathrm{e}^{\varepsilon}\Pr[M(D_0)=r|E] - \Pr[M(D_1)=r|F]\right)\,,\\
\Pr[T(2)=r] &= \frac{1}{\mathrm{e}^{\varepsilon}-1}\cdot\left(\mathrm{e}^{\varepsilon}\Pr[M(D_1)=r|F] - \Pr[M(D_0)=r|E]\right)\,,\\
\Pr[T(3)=r] &= \Pr[M(D_1)=r|\neg F]\,.
\end{aligned}$$

Note that for each input to $T$, the probabilities of the outputs sum to 1 and are all non-negative because

$$D_\infty^0(X|_E \parallel Y|_F) \leq \varepsilon \qquad \text{and} \qquad D_\infty^0(Y|_F \parallel X|_E) \leq \varepsilon\,.$$

So the outputs of $T$ form a valid probability distribution.

We now show that for all $r \in R$,

$$\Pr[T(\tilde{M}_{(\varepsilon,\delta)}(0)) = r] = \Pr[M(D_0)=r]\,.$$

It follows that

$$T(\tilde{M}_{(\varepsilon,\delta)}(0)) \sim M(D_0)$$

and by symmetry that

$$T(\tilde{M}_{(\varepsilon,\delta)}(1)) \sim M(D_1)\,,$$

which will complete the proof. We use the shorthand $P_b(r|H) = \Pr[M(D_b) = r|H]$ for $b \in \{0,1\}$ and any event $H$. Fix $r \in R$.

$$\begin{aligned}
&\Pr[T(\tilde{M}_{(\varepsilon,\delta)}(0)) = r] \\
&\quad = \delta \cdot \Pr[T(0) = r] + \frac{(1-\delta)\mathrm{e}^{\varepsilon}}{1+\mathrm{e}^{\varepsilon}} \cdot \Pr[T(1) = r] + \frac{(1-\delta)}{1+\mathrm{e}^{\varepsilon}} \cdot \Pr[T(2) = r] + 0 \\
&\quad = \delta \cdot P_0(r|\neg E) + \frac{(1-\delta)\mathrm{e}^{\varepsilon}}{\mathrm{e}^{2\varepsilon}-1} \cdot (\mathrm{e}^{\varepsilon} P_0(r|E) - P_1(r|F)) + \frac{(1-\delta)}{\mathrm{e}^{2\varepsilon}-1} \cdot (\mathrm{e}^{\varepsilon} P_1(r|F) - P_0(r|E)) \\
&\quad = \delta \cdot P_0(r|\neg E) + \frac{(1-\delta)\mathrm{e}^{\varepsilon}}{\mathrm{e}^{2\varepsilon}-1} \cdot \mathrm{e}^{\varepsilon} \cdot P_0(r|E) - \frac{(1-\delta)}{\mathrm{e}^{2\varepsilon}-1} \cdot P_0(r|E) \\
&\quad = \delta \cdot P_0(r|\neg E) + (1-\delta) \cdot P_0(r|E) \\
&\quad = \Pr[\neg E] \cdot P_0(r|\neg E) + \Pr[E] \cdot P_0(r|E) \\
&\quad = \Pr[M(D_0) = r]\,. \qquad \square
\end{aligned}$$

So $\tilde{M}_{(\varepsilon,\delta)}$ can simulate any $(\varepsilon,\delta)$ differentially private algorithm. Since it is known that post-processing preserves differential privacy (Fact 2.2), it follows that to analyze the composition of arbitrary differentially private algorithms, it suffices to analyze the composition of algorithms $\tilde{M}_{(\varepsilon_i,\delta_i)}$.

**Lemma 3.4.** *For all* $\varepsilon_1,\ldots,\varepsilon_k \geq 0$*,* $\delta_1,\ldots,\delta_k$*,* $\delta_g \in [0,1)$*,*

$$\mathrm{OptComp}((\varepsilon_1,\delta_1),\ldots,(\varepsilon_k,\delta_k),\delta_g) = \mathrm{OptComp}(\tilde{M}_{(\varepsilon_1,\delta_1)},\ldots,\tilde{M}_{(\varepsilon_k,\delta_k)},\delta_g)\,.$$

*Proof.* Since $\tilde{M}_{(\varepsilon_1,\delta_1)},\ldots,\tilde{M}_{(\varepsilon_k,\delta_k)}$ are $(\varepsilon_1,\delta_1),\ldots,(\varepsilon_k,\delta_k)$-differentially private, we have

$$\begin{aligned}
\mathrm{OptComp}((\varepsilon_1,\delta_1),\ldots,(\varepsilon_k,\delta_k),\delta_g) &= \sup\{\mathrm{OptComp}(M_1,\ldots,M_k,\delta_g)\colon M_i \text{ is } (\varepsilon_i,\delta_i)\text{-DP } \forall i \in [k]\} \\
&\geq \mathrm{OptComp}(\tilde{M}_{(\varepsilon_1,\delta_1)},\ldots,\tilde{M}_{(\varepsilon_k,\delta_k)},\delta_g)\,.
\end{aligned}$$

For the other direction, it suffices to show that for every $M_1,\ldots,M_k$ that are $(\varepsilon_1,\delta_1),\ldots,(\varepsilon_k,\delta_k)$-differentially private, we have

$$\mathrm{OptComp}(M_1,\ldots,M_k,\delta_g) \leq \mathrm{OptComp}(\tilde{M}_{(\varepsilon_1,\delta_1)},\ldots,\tilde{M}_{(\varepsilon_k,\delta_k)},\delta_g)\,.$$

That is,

$$\inf\{\varepsilon_g \geq 0\colon (M_1,\ldots,M_k) \text{ is } (\varepsilon_g,\delta_g)\text{-DP}\} \leq \inf\{\varepsilon_g \geq 0\colon (\tilde{M}_{(\varepsilon_1,\delta_1)},\ldots,\tilde{M}_{(\varepsilon_k,\delta_k)}) \text{ is } (\varepsilon_g,\delta_g)\text{-DP}\}\,.$$

So suppose $(\tilde{M}_{(\varepsilon_1,\delta_1)},\ldots,\tilde{M}_{(\varepsilon_k,\delta_k)})$ is $(\varepsilon_g,\delta_g)$-DP. We will show that $(M_1,\ldots,M_k)$ is also $(\varepsilon_g,\delta_g)$-DP. Taking the infimum over $\varepsilon_g$ then completes the proof.

We know from Lemma 3.2 that for every pair of neighboring databases $D_0, D_1$, there must exist randomized algorithms $T_1,\ldots,T_k$ such that $T_i(\tilde{M}_{(\varepsilon_i,\delta_i)}(b))$ is identically distributed to $M_i(D_b)$ for all $i \in \{1,\ldots,k\}$. By hypothesis we have

$$D_{\infty}^{\delta_g}\left((\tilde{M}_{(\varepsilon_1,\delta_1)}(0),\ldots,\tilde{M}_{(\varepsilon_k,\delta_k)}(0)) \| (\tilde{M}_{(\varepsilon_1,\delta_1)}(1),\ldots,\tilde{M}_{(\varepsilon_k,\delta_k)}(1))\right) \leq \varepsilon_g\,.$$

Thus by Fact 2.2 we have

$$\begin{aligned}
&D_\infty^{\delta_g}\big((M_1(D_0),\dots,M_k(D_0))\|(M_1(D_1),\dots,M_k(D_1))\big)\\
&\quad = D_\infty^{\delta_g}\big((T_1(\tilde{M}_{(\varepsilon_1,\delta_1)}(0)),\dots,T_k(\tilde{M}_{(\varepsilon_k,\delta_k)}(0)))\|(T_1(\tilde{M}_{(\varepsilon_1,\delta_1)}(1)),\dots,T_k(\tilde{M}_{(\varepsilon_k,\delta_k)}(1)))\big) \le \varepsilon_g\,. \quad \square
\end{aligned}$$

Now we are ready to characterize OptComp for an arbitrary set of differentially private algorithms.

**Theorem 1.5 (restated).** *For all $\varepsilon_1,\dots,\varepsilon_k \ge 0$ and $\delta_1,\dots,\delta_k,\delta_g \in [0,1)$,*

$$\mathrm{OptComp}((\varepsilon_1,\delta_1),(\varepsilon_2,\delta_2),\dots,(\varepsilon_k,\delta_k),\delta_g)$$

*equals the least value of $\varepsilon_g \ge 0$ such that*

$$\frac{1}{\prod_{i=1}^k(1+\mathrm{e}^{\varepsilon_i})}\sum_{S\subseteq\{1,\dots,k\}}\max\left\{\mathrm{e}^{\sum\limits_{i\in S}\varepsilon_i}-\mathrm{e}^{\varepsilon_g}\cdot\mathrm{e}^{\sum\limits_{i\notin S}\varepsilon_i},0\right\} \le 1-\frac{1-\delta_g}{\prod_{i=1}^k(1-\delta_i)}\,.$$

*Proof of Theorem 1.5.* Given $(\varepsilon_1,\delta_1),\dots,(\varepsilon_k,\delta_k)$ and $\delta_g$, let $\tilde{M}^k(b)$ denote the composition

$$(\tilde{M}_{(\varepsilon_1,\delta_1)}(b),\dots,\tilde{M}_{(\varepsilon_k,\delta_k)}(b))$$

and let $\tilde{P}_b^k(x)$ be the probability mass function of $\tilde{M}^k(b)$, for $b=0$ and $b=1$. By Lemma 3.4,

$$\mathrm{OptComp}((\varepsilon_1,\delta_1),\dots,(\varepsilon_\mathrm{k},\delta_\mathrm{k}),\delta_\mathrm{g})$$

is the smallest value of $\varepsilon_g$ such that

$$\delta_g \ge \max_{Q\subseteq\{0,1,2,3\}^k}\left\{\tilde{P}_0^k(Q)-\mathrm{e}^{\varepsilon_g}\cdot\tilde{P}_1^k(Q),\tilde{P}_1^k(Q)-\mathrm{e}^{\varepsilon_g}\cdot\tilde{P}_0^k(Q)\right\}.$$

Since $\tilde{M}$ is symmetric, we can instead consider the smallest value of $\varepsilon_g$ such that

$$\delta_g \ge \max_{Q\subseteq\{0,1,2,3\}^k}\left\{\tilde{P}_0^k(Q)-\mathrm{e}^{\varepsilon_g}\cdot\tilde{P}_1^k(Q)\right\}, \tag{3.1}$$

without loss of generality. Given $\varepsilon_g$, the set $S\subseteq\{0,1,2,3\}^k$ that maximizes the right-hand side is

$$S=S(\varepsilon_g)=\left\{x\in\{0,1,2,3\}^k \;\middle|\; \tilde{P}_0^k(x)\ge \mathrm{e}^{\varepsilon_g}\cdot\tilde{P}_1^k(x)\right\}.$$

We can further split $S(\varepsilon_g)$ into $S(\varepsilon_g)=S_0(\varepsilon_g)\cup S_1(\varepsilon_g)$ with

$$\begin{aligned}
S_0(\varepsilon_g)&=\left\{x\in\{0,1,2,3\}^k \;\middle|\; \tilde{P}_1^k(x)=0\right\},\\
S_1(\varepsilon_g)&=\left\{x\in\{0,1,2,3\}^k \;\middle|\; \tilde{P}_0^k(x)\ge \mathrm{e}^{\varepsilon_g}\cdot\tilde{P}_1^k(x),\text{ and }\tilde{P}_1^k(x)>0\right\}.
\end{aligned}$$

Note that $S_0(\varepsilon_g)\cap S_1(\varepsilon_g)=\emptyset$. We have

$$\tilde{P}_1^k(S_0(\varepsilon_g))=0 \qquad\text{and}\qquad \tilde{P}_0^k(S_0(\varepsilon_g))=1-\Pr[\tilde{M}^k(0)\in\{1,2,3\}^k]=1-\prod_{i=1}^k(1-\delta_i)\,.$$

So

$$\begin{aligned}\tilde{P}_0^k(S(\varepsilon_g)) - \mathrm{e}^{\varepsilon_g}\tilde{P}_1^k(S(\varepsilon_g)) &= \tilde{P}_0^k(S_0(\varepsilon_g)) - \mathrm{e}^{\varepsilon_g}\tilde{P}_1^k(S_0(\varepsilon_g)) + \tilde{P}_0^k(S_1(\varepsilon_g)) - \mathrm{e}^{\varepsilon_g}\tilde{P}_1^k(S_1(\varepsilon_g)) \\ &= 1 - \prod_{i=1}^{k}(1-\delta_i) + \tilde{P}_0^k(S_1(\varepsilon_g)) - \mathrm{e}^{\varepsilon_g}\tilde{P}_1^k(S_1(\varepsilon_g))\,. \qquad (3.2)\end{aligned}$$

Now we just need to analyze

$$\tilde{P}_0^k(S_1(\varepsilon_g)) - \mathrm{e}^{\varepsilon_g}\tilde{P}_1^k(S_1(\varepsilon_g))\,.$$

Notice that $S_1(\varepsilon_g) \subseteq \{1,2\}^k$ because for all $x \in S_1(\varepsilon_g)$, we have $\tilde{P}_0(x) > \tilde{P}_1(x) > 0$. So we can write

$$\begin{aligned}&\tilde{P}_0^k(S_1(\varepsilon_g)) - \mathrm{e}^{\varepsilon_g}\cdot\tilde{P}_1^k(S_1(\varepsilon_g)) \\ &= \sum_{y\in\{1,2\}^k} \max\left\{\prod_{i:\,y_i=1}\frac{(1-\delta_i)\mathrm{e}^{\varepsilon_i}}{1+\mathrm{e}^{\varepsilon_i}}\cdot\prod_{i:\,y_i=2}\frac{(1-\delta_i)}{1+\mathrm{e}^{\varepsilon_i}} - \mathrm{e}^{\varepsilon_g}\prod_{i:\,y_i=1}\frac{(1-\delta_i)}{1+\mathrm{e}^{\varepsilon_i}}\cdot\prod_{i:\,y_i=2}\frac{(1-\delta_i)\mathrm{e}^{\varepsilon_i}}{1+\mathrm{e}^{\varepsilon_i}},0\right\} \\ &= \prod_{i=1}^{k}\frac{1-\delta_i}{1+\mathrm{e}^{\varepsilon_i}}\sum_{y\in\{0,1\}^k}\max\left\{\frac{\mathrm{e}^{\sum_{i=1}^k \varepsilon_i}}{\mathrm{e}^{\sum_{i=1}^k y_i\varepsilon_i}} - \mathrm{e}^{\varepsilon_g}\cdot\mathrm{e}^{\sum_{i=1}^k y_i\varepsilon_i},0\right\}. \qquad (3.3)\end{aligned}$$

Combining Equations (3.1), (3.2), and (3.3) together yields

$$\begin{aligned}\delta_g &\geq \tilde{P}_0^k(S_0(\varepsilon_g)) - \mathrm{e}^{\varepsilon_g}\tilde{P}_1^k(S_0(\varepsilon_g)) + \tilde{P}_0^k(S_1(\varepsilon_g)) - \mathrm{e}^{\varepsilon_g}\tilde{P}_1^k(S_1(\varepsilon_g)) \\ &= 1 - \prod_{i=1}^{k}(1-\delta_i) + \frac{\prod_{i=1}^k(1-\delta_i)}{\prod_{i=1}^k(1+\mathrm{e}^{\varepsilon_i})}\sum_{S\subseteq\{1,\ldots,k\}}\max\left\{\mathrm{e}^{\sum\limits_{i\in S}\varepsilon_i} - \mathrm{e}^{\varepsilon_g}\cdot\mathrm{e}^{\sum\limits_{i\notin S}\varepsilon_i},0\right\}. \qquad \square\end{aligned}$$

We have characterized the optimal composition for an arbitrary set of differentially private algorithms $(M_1,\ldots,M_k)$ under the assumption that the algorithms are chosen in advance and all run on the same database. Next we show that OptComp under this restrictive model of composition is actually equivalent under the more general adaptive composition discussed in Section 2.

**Theorem 3.5.** *The privacy parameters* $\varepsilon_1,\ldots,\varepsilon_k \geq 0$, $\delta_1,\ldots,\delta_k \in [0,1)$, *satisfy* $(\varepsilon_g,\delta_g)$*-differential privacy under adaptive composition for* $\varepsilon_g,\delta_g \geq 0$ *if and only if*

$$\mathrm{OptComp}((\varepsilon_1,\delta_1),\ldots,(\varepsilon_k,\delta_k),\delta_g) \leq \varepsilon_g.$$

*Proof.* First suppose the privacy parameters $\varepsilon_1,\ldots,\varepsilon_k,\delta_1,\ldots,\delta_k$ satisfy $(\varepsilon_g,\delta_g)$-differential privacy under adaptive composition. Then $\mathrm{OptComp}((\varepsilon_1,\delta_1),\ldots,(\varepsilon_k,\delta_k),\delta_g) \leq \varepsilon_g$ because adaptive composition is more general than the composition defining OptComp.

Conversely, suppose $\mathrm{OptComp}((\varepsilon_1,\delta_1),\ldots,(\varepsilon_k,\delta_k),\delta_g) \leq \varepsilon_g$. In particular, this means

$$\mathrm{OptComp}(\tilde{M}_{(\varepsilon_1,\delta_1)},\ldots,\tilde{M}_{(\varepsilon_k,\delta_k)},\delta_g) \leq \varepsilon_g\,.$$

To complete the proof, we must show that the privacy parameters $\varepsilon_1,\ldots,\varepsilon_k,\delta_1,\ldots,\delta_k$ satisfy $(\varepsilon_g,\delta_g)$-differential privacy under adaptive composition.

Fix an adversary $A$. On each round $i$, $A$ uses its coin tosses $r$ and the previous outputs $y_1,\dots,y_{i-1}$ to select an $(\varepsilon_i,\delta_i)$-differentially private algorithm $M_i = M_i^{r,y_1,\dots,y_{i-1}}$ and neighboring databases

$$D_0 = D_0^{r,y_1,\dots,y_{i-1}}, \qquad D_1 = D_1^{r,y_1,\dots,y_{i-1}}.$$

Let $V^b$ be the view of $A$ with the given privacy parameters under composition game $b$ for $b=0$ and $b=1$.

Lemma 3.2 tells us that there exists an algorithm $T_i = T_i^{r,y_1,\dots,y_{i-1}}$ such that

$$T_i(\tilde{M}_{(\varepsilon_i,\delta_i)}(b))$$

is identically distributed to $M_i(D_b)$ for both $b=0,1$ for all $i\in[k]$. Define $\hat{T}(z_1,\dots,z_k)$ for $z_1,\dots,z_k\in\{0,1,2,3\}$ as follows.

1. Randomly choose coins $r$ for $A$.
2. For $i=1,\dots,k$, let $y_i \leftarrow T_i^{r,y_1,\dots,y_{i-1}}(z_i)$.
3. Output $(r,y_1,\dots,y_k)$.

Notice that

$$\hat{T}(\tilde{M}_{(\varepsilon_1,\delta_1)}(b),\dots,\tilde{M}_{(\varepsilon_k,\delta_k)}(b))$$

is identically distributed to $V^b$ for both $b=0,1$. By hypothesis we have

$$D_\infty^{\delta_g}\left((\tilde{M}_{(\varepsilon_1,\delta_1)}(0),\dots,\tilde{M}_{(\varepsilon_k,\delta_k)}(0))\|(\tilde{M}_{(\varepsilon_1,\delta_1)}(1),\dots,\tilde{M}_{(\varepsilon_k,\delta_k)}(1))\right)\le\varepsilon_g\,.$$

Thus by Fact 2.2 we have

$$D_\infty^{\delta_g}\left(V^0\|V^1\right) = D_\infty^{\delta_g}\left(\hat{T}(\tilde{M}_{(\varepsilon_1,\delta_1)}(0),\dots,\tilde{M}_{(\varepsilon_k,\delta_k)}(0))\|\hat{T}(\tilde{M}_{(\varepsilon_1,\delta_1)}(1),\dots,\tilde{M}_{(\varepsilon_k,\delta_k)}(1))\right)\le\varepsilon_g\,. \qquad \square$$

# 4 Hardness of OptComp

$\#\mathsf{P}$ is the class of all counting problems associated with decision problems in $\mathsf{NP}$. It is a set of functions that count the solutions to some $\mathsf{NP}$ problem. More formally,

**Definition 4.1.** A function $f\colon\{0,1\}^*\to\mathbb{N}$ is in the class $\#\mathsf{P}$ if there exists a polynomial $p\colon\mathbb{N}\to\mathbb{N}$ and a polynomial time algorithm $M$ such that for every $x\in\{0,1\}^*$,

$$f(x) = \left|\left\{y\in\{0,1\}^{p(|x|)}\colon M(x,y)=1\right\}\right|.$$

**Definition 4.2.** For functions $f,g\in\#\mathsf{P}$, we say that $f$ *reduces to* $g$ (written $f\le g$) if there exists a polynomial time algorithm $M$ such that for all $x\in\{0,1\}^*$, $M(x)=f(x)$ when $M$ is given oracle access to $g$. That is, evaluations of $g$ can be done in one time step.

**Definition 4.3.** A function $g$ is called $\#\mathsf{P}$-*hard* if every function $f\in\#\mathsf{P}$ can be computed in polynomial time given oracle access to $g$. That is, evaluations of $g$ can be done in one time step.

If a function is #P-hard, then there is no polynomial-time algorithm for computing it unless there is a polynomial-time algorithm for counting the solutions of all NP problems.

**Definition 4.4.** A function $f$ is called #P-*easy* if there is some function $g \in \#\mathsf{P}$ such that $f$ can be computed in polynomial time given oracle access to $g$.

If a function is both #P-hard and #P-easy, we say it is #P-complete. In this section we prove Theorem 1.6.

**Theorem 1.6 (restated).** *Computing* OptComp *is* #P-*complete, even on instances where* $\delta_1 = \delta_2 = \ldots = \delta_k = 0$ *and* $\sum_{i\in[k]} \varepsilon_i \le \varepsilon$ *for any desired constant* $\varepsilon > 0$.

Proving that computing OptComp is #P-complete can be broken into two steps: showing that it is #P-easy and showing that it is #P-hard.

**Lemma 4.5.** *Computing OptComp is* #P-*easy.*

*Proof.* For convenience we will view rational $(\varepsilon_1,\delta_1),\ldots,(\varepsilon_k,\delta_k)$ and $\varepsilon_g$ as given arguments to OptComp and compute $\delta_g$. Recall that the two versions of OptComp, viewing $\varepsilon_g$ as given and computing $\delta_g$ and vice versa, are equivalent up to a polynomial factor (just run binary search over values of $\delta_g$ computing polynomially many bits of precision). So the formulation we choose for the proof will not affect whether OptComp is in #P or not. Recall that in our model of computing real valued functions, we will take another input $q$ and we will output an approximation of $\delta_g$ to $q$ bits of precision in polynomial time using a #P oracle where $\delta_g$ satisfies the following.

$$\frac{1}{\prod_{i=1}^{k}(1+\mathrm{e}^{\varepsilon_i})}\sum_{S\subseteq\{1,\ldots,k\}}\max\left\{\mathrm{e}^{\sum\limits_{i\in S}\varepsilon_i}-\mathrm{e}^{\varepsilon_g}\cdot\mathrm{e}^{\sum\limits_{i\notin S}\varepsilon_i},0\right\}=1-\frac{1-\delta_g}{\prod_{i=1}^{k}(1-\delta_i)}.$$

Notice that the only part of the expression above that cannot be computed in polynomial time is the summation over subsets of $\{1,\ldots,k\}$. If we knew the sum, computing $\delta_g$ would be easy given our inputs. We show how to compute the sum in polynomial time using a #P oracle and it follows that computing $\delta_g$ is #P-easy.

Define $f\colon 2^{[k]} \to \mathbb{R}$ as

$$f(S)=\max\left\{\mathrm{e}^{\sum\limits_{i\in S}\varepsilon_i}-\mathrm{e}^{\varepsilon_g}\cdot\mathrm{e}^{\sum\limits_{i\notin S}\varepsilon_i},0\right\}.$$

$f$ is computable in polynomial time (to any desired precision). Let $\hat{f}$ be a function computable in polynomial time where $\left|\hat{f}(S)-f(S)\right| < 1/2^{q+k}$ for all $S$. Set $m = 10^q$. Now define the function $g\colon 2^{[k]} \times \mathbb{N} \to \{0,1\}$ as follows.

$$g(S,n)=\begin{cases}1 & \text{if } m\cdot\hat{f}(S)\ge n,\\ 0 & \text{otherwise.}\end{cases}$$

We can now phrase a decision problem in NP: Does there exist a pair $(S,n)$ such that $g(S,n) = 1$? This is in NP because given a witness $(S,n)$, we can compute $m\cdot\hat{f}(S)$ and compare the output to $n$, thereby verifying the solution, in polynomial time. Since this is an NP problem, a #P oracle can count the solutions to it in one time step. Notice that for every set $S$, the number of solutions (pairs of the form $(S,n)$

satisfying $g(S,n)=1$) is exactly $m\cdot\hat{f}(S)$ because $g$ will output 1 for $g(S,1),g(S,2),\ldots,g(S,m\cdot\hat{f}(S))$. So over all possible sets $S$, the number of solutions as counted by the #P oracle equals $m\cdot\sum_{S\subseteq[k]}\hat{f}(S)$. Dividing this by $m$ gives us the sum up to an additive error of $2^k/2^{q+k}=1/2^q$, which can be used to compute $\delta_g$ to $q$ bits of precision in polynomial time. This only required one call to a #P oracle. So computing OptComp is #P-easy. □

Next we show that computing OptComp is also #P-hard through a series of reductions. We start with a multiplicative version of the partition problem that is known to be #P-complete by Ehrgott [9]. The problems in the chain of reductions are defined below.

**Definition 4.6.** #INT-PARTITION is the following problem. Given a set $Z=\{z_1,z_2,\ldots,z_k\}$ of positive integers, count the partitions $P\subseteq[k]$ such that

$$\prod_{i\in P} z_i-\prod_{i\notin P} z_i=0\,.$$

All of the remaining problems in our chain of reductions take inputs $\{w_1,\ldots,w_k\}$ where $1\le w_i\le e$ is the $D$-th root of a positive integer $z_i$ for all $i\in[k]$ and some positive integer $D$. All of the reductions we present actually hold for every positive integer $D$, including $D=1$ (in which case the inputs are integers). However, we will constrain $D$ to be large enough so that our inputs are in the range $[1,e]$. This is because in the final reduction to OptComp, $\varepsilon_i$ values in the proof are set to $\ln(w_i)$. We want to show that our reductions hold for reasonable values of the $\varepsilon_i$ in a differential privacy setting so throughout the proofs we use the $w_i\in[1,e]$ to correspond to the $\varepsilon_i\in[0,1]$ in the final reduction. In fact, we will later state our reductions as applying to instances where $\prod_i w_i\le \mathrm{e}^{\varepsilon}$ (and hence $\sum_i\varepsilon_i\le\varepsilon$) for any desired $\varepsilon>0$.

**Definition 4.7.** #PARTITION is the following problem. Given a positive integer $D\in\mathbb{N}$ and a set $W=\{w_1,w_2,\ldots,w_k\}$ of real numbers where $1\le w_1,\ldots,w_k\le e$ are $D$-th roots of positive integers $z_1,\ldots z_k$, count the partitions $P\subseteq[k]$ such that

$$\prod_{i\in P} w_i-\prod_{i\notin P} w_i=0\,.$$

(The real numbers $w_1,\ldots,w_k$ are specified in the input by $z_1,\ldots,z_k$ and $D$ with the input size being the combined bit-length of these integers in binary.)

**Definition 4.8.** #T-PARTITION is the following problem. Given a positive integer $D\in\mathbb{N}$, a set $W=\{w_1,w_2\ldots,w_k\}$ of real numbers and a *positive* real number $T$, where $1\le w_1,\ldots,w_k\le e$ are $D$-th roots of positive integers $z_1,\ldots z_k$, and $T=\sqrt[2D]{t}-\sqrt[2D]{t'}$ for two integers $t,t'$, count the partitions $P\subseteq[k]$ such that

$$\prod_{i\in P} w_i-\prod_{i\notin P} w_i=T.$$

(The real numbers $w_1,\ldots,w_k$ and $T$ are specified in the input by $z_1,\ldots,z_k,t,t'$ and $D$ with the input size being the combined bit-length of these integers in binary.)

**Definition 4.9.** SUM-PARTITION is the following problem. Given a positive integer $D \in \mathbb{N}$ and a set $W = \{w_1, w_2, \dots, w_k\}$ of real numbers where $1 \le w_1, \dots, w_k \le e$ are $D$-th roots of positive integers $z_1, \dots z_k$, and a rational number $r > 1$, find

$$\sum_{P \subseteq [k]} \max\left\{\prod_{i \in P} w_i - r \cdot \prod_{i \notin P} w_i, 0\right\}.$$

(The real numbers $w_1, \dots, w_k$ are specified in the input by $z_1, \dots, z_k$ and $D$ with the input size being the combined bit-length of these integers and the numerator and denominator of $r$ in binary.)

Since the output of SUM-PARTITION is irrational, the actual computational problem is defined according to our convention in Section 2 for computing real-valued functions. That is, given an additional input $q$, compute a number $y$ such that

$$\left| y - \sum_{P \subseteq [k]} \max\left\{\prod_{i \in P} w_i - r \cdot \prod_{i \notin P} w_i, 0\right\}\right| < \frac{1}{2^q}.$$

We prove that computing OptComp is $\#\mathsf{P}$-hard by the following series of reductions.

$$\text{\#INT-PARTITION} \le \text{\#PARTITION} \le \text{\#T-PARTITION} \le \text{SUM-PARTITION} \le \text{OptComp}\,.$$

Since #INT-PARTITION is known to be $\#\mathsf{P}$-complete [9], the chain of reductions will prove that OptComp is $\#\mathsf{P}$-hard.

**Lemma 4.10.** *For every constant $c > 1$, #PARTITION is $\#\mathsf{P}$-hard, even on instances where $\prod_i w_i \le c$.*

*Proof.* Given an instance of #INT-PARTITION, $\{z_1, \dots, z_k\}$, we show how to find the solution in polynomial time using a #PARTITION oracle. Set $D = \lceil \log_c(\prod_i z_i) \rceil$ and $w_i = \sqrt[D]{z_i}\ \forall i \in [k]$. Note that $\prod_i w_i = (\prod_i z_i)^{1/D} \le c$. Let $P \subseteq [k]$.

$$\begin{aligned}\prod_{i \in P} w_i = \prod_{i \notin P} w_i &\iff \left(\prod_{i \in P} w_i\right)^D = \left(\prod_{i \notin P} w_i\right)^D \\ &\iff \prod_{i \in P} z_i = \prod_{i \notin P} z_i\,.\end{aligned}$$

There is a one-to-one correspondence between solutions to the #PARTITION problem and solutions to the given #INT-PARTITION instance. We can solve #INT-PARTITION in polynomial time with a #PARTITION oracle. Therefore #PARTITION is $\#\mathsf{P}$-hard. □

**Lemma 4.11.** *For every constant $c > 1$, #T-PARTITION is $\#\mathsf{P}$-hard, even on instances where $\prod_i w_i \le c$.*

*Proof.* Let $c > 1$ be a constant. We will reduce from #PARTITION, so consider an instance of the #PARTITION problem, $W = \{w_1, w_2, \dots, w_k\}$ of $D$-th roots of integers $z_1, \dots, z_k$. We may assume $\prod_i w_i \le \sqrt{c}$ since $\sqrt{c}$ is also a constant greater than 1.

Set $W' = W \cup \{w_{k+1}\}$, where $w_{k+1} = \prod_{i=1}^{k} w_i$. Notice that

$$\prod_{i=1}^{k+1} w_i \leq (\sqrt{c})^2 = c\,.$$

Set $T = \sqrt{w_{k+1}}\,(w_{k+1}-1)$. Notice that $w_{k+1} = \left(\prod_{i=1}^{k} z_i\right)^{\frac{1}{D}}$ so by setting integers

$$t = \left(\prod_{i=1}^{k} z_i\right)^3 \qquad \text{and} \qquad t' = \prod_{i=1}^{k} z_i$$

we get that

$$T = \sqrt[2D]{t} - \sqrt[2D]{t'}$$

which meets the input requirement for #T-PARTITION. So we can use a #T-PARTITION oracle to count partitions $Q \subseteq \{1,\ldots,k+1\}$ such that

$$\prod_{i\in Q} w_i - \prod_{i\notin Q} w_i = T.$$

Let $P = Q \cap \{1,\ldots,k\}$. We will argue that $\prod_{i\in Q} w_i - \prod_{i\notin Q} w_i = T$ if and only if $\prod_{i\in P} w_i = \prod_{i\notin P} w_i$, which completes the proof. There are two cases to consider: $w_{k+1} \in Q$ and $w_{k+1} \notin Q$.

**Case 1:** $w_{k+1} \in Q$. In this case, we have

$$\begin{aligned}
& w_{k+1}\cdot\left(\prod_{i\in P} w_i\right) - \prod_{i\notin P} w_i = \prod_{i\in Q} w_i - \prod_{i\notin Q} w_i = T = \sqrt{w_{k+1}}\,(w_{k+1}-1) \\
\iff\ & \left(\prod_{i\in[k]} w_i\right)\left(\prod_{i\in P} w_i\right)^2 - \prod_{i\in[k]} w_i = \sqrt{\prod_{i\in[k]} w_i}\left(\prod_{i\in[k]} w_i - 1\right)\left(\prod_{i\in P} w_i\right) && \text{multiplied by} \prod_{i\in P} w_i \\
\iff\ & \left(\prod_{i\in P} w_i - \sqrt{\prod_{i\in[k]} w_i}\right)\left(\prod_{i\in[k]} w_i \prod_{i\in P} w_i + \sqrt{\prod_{i\in[k]} w_i}\right) = 0 && \text{factored quadratic in} \prod_{i\in P} w_i \\
\iff\ & \prod_{i\in P} w_i = \sqrt{\prod_{i\in[k]} w_i} \\
\iff\ & \prod_{i\notin P} w_i = \prod_{i\in P} w_i.
\end{aligned}$$

So there is a one-to-one correspondence between solutions to the #T-PARTITION instance $W'$ where $w_{k+1} \in Q$ and solutions to the original #PARTITION instance $W$.

**Case 2:** $w_{k+1} \notin Q$. Solutions now look like

$$\prod_{i\in P} w_i - \prod_{i\in[k]} w_i \prod_{i\notin P} w_i = \sqrt{\prod_{i\in[k]} w_i}\left(\prod_{i\in[k]} w_i - 1\right).$$

One way this can be true is if $w_i = 1$ for all $i \in [k]$. We can check ahead of time if our input set $W$ contains all ones. If it does, then there are $2^k - 2$ partitions that yield equal products (all except $P = [k]$ and $P = \emptyset$) so we can just output $2^k - 2$ as the solution and not even use our oracle. The only other way to satisfy the above expression is for

$$\prod_{i \in P} w_i > \prod_{i \in [k]} w_i$$

which cannot happen because $P \subseteq [k]$. So there are no solutions in the case that $w_{k+1} \notin Q$.

Therefore the output of the #T-PARTITION oracle on $W'$ is the solution to the #PARTITION problem. So #T-PARTITION is $\#\mathsf{P}$-hard. □

For the next two proofs we will make use of the following fact to bound the amount of precision needed when approximating irrational numbers by rational ones in our reductions.

**Fact 4.12.** *For all real numbers $y > x$ and functions $f$ that are differentiable on the interval $[x, y]$,*

$$f(y) - f(x) \geq (y - x) \cdot \min_{z \in (x,y)} f'(z)\,.$$

**Lemma 4.13.** *For every constant $c > 1$,* SUM-PARTITION *is $\#\mathsf{P}$-hard even on instances where $\prod_i w_i \leq c$ and where there are no partitions $S$ such that*

$$\prod_{i \in S} w_i = r \cdot \prod_{i \notin S} w_i\,.$$

*Proof.* We will use a SUM-PARTITION oracle to solve #T-PARTITION given a set $W = \{w_1, \ldots, w_k\}$ of $D$-th roots of positive integers $z_1, \ldots, z_k$, and a positive real number $T = \sqrt[2D]{t} - \sqrt[2D]{t'}$ for integers $t, t'$ given in the input. Notice that for every $x > 0$,

$$\begin{aligned} \prod_{i \in P} w_i - \prod_{i \notin P} w_i = x &\implies \prod_{i \in P} w_i - \frac{\prod_{i \in [k]} w_i}{\prod_{i \in P} w_i} = x \\ &\implies \exists\, j \in \mathbb{Z}^+ \text{such that } \sqrt[D]{j} - \frac{\prod_{i \in [k]} w_i}{\sqrt[D]{j}} = x\,. \end{aligned}$$

Above, $j$ must be a positive integer greater than $\left(\prod_{i=1}^k z_i\right)^{1/2}$, which tells us that the gap in products from every partition must take a particular form. This means that for a given $D$ and $W$, #x-PARTITION can only be non-zero on a discrete set of possible values of $x$. So given our #T-PARTITION instance we can find a $T' > T$ such that the above has no solutions for $x$ in the interval $(T, T')$. Specifically, solve the above quadratic for $\sqrt[D]{j}$. If $j$ is not an integer, then we know the answer to the #T-PARTITION instance is 0, so assume $j$ is an integer and set $T' = \sqrt[D]{j+1} - \prod_i w_i / \sqrt[D]{j+1}$. We can also find an interval $(T'', T)$ just below $T$ where no value of $x$ in the interval can yield a solution above by setting $T'' = \sqrt[D]{j-1} - \prod_i w_i / \sqrt[D]{j-1}$. We use these discreteness properties twice in the proof. Also notice that these intervals are not too small.

**Claim 4.14.** *$T' - T \geq 2^{-poly(n)}$ and $T - T'' \geq 2^{-poly(n)}$ where $n$ is the input length (i. e., the bit-lengths of the integers $z_1, \ldots, z_k, t, t'$).*

*Proof of Claim.*

$$T' - T = \sqrt[D]{j+1} - \frac{\prod_{i\in[k]} w_i}{\sqrt[D]{j+1}} - \sqrt[D]{j} + \frac{\prod_{i\in[k]} w_i}{\sqrt[D]{j}} \geq \sqrt[D]{j+1} - \sqrt[D]{j} \geq \frac{1}{D(j+1)},$$

where the last inequality follows from Fact 4.12. This final value is only exponentially small because $j$ is upper bounded by $\prod_{i=1}^{k} z_i$, which is at most exponentially large in the bit-length of the $z_i$. A very similar proof shows that $(T'', T)$ is only exponentially small. □

This means that we can always find $\hat{T} \in (T, T')$ such that $\hat{T}$ is rational and can be fully specified with a bit-length that is polynomial in the input length. Fix such a quantity $\hat{T}$. For all $y > 0$, define

$$P^y \equiv \left\{ P \subseteq [k] \,\middle|\, \prod_{i\in P} w_i - \prod_{i\notin P} w_i \geq y \right\}.$$

Then, since $x$-PARTITION has no solutions for $x \in (T, T')$,

$$\begin{aligned}
\left| \left\{ P \subseteq [k] \,\middle|\, \prod_{i\in P} w_i - \prod_{i\notin P} w_i = T \right\} \right| &= \left| P^T \setminus P^{\hat{T}} \right| \\
&= \frac{1}{T} \left( \sum_{P\in P^T \setminus P^{\hat{T}}} \left( \prod_{i\in P} w_i - \prod_{i\notin P} w_i \right) \right) \\
&= \frac{1}{T} \left( \sum_{P\in P^T} \left( \prod_{i\in P} w_i - \prod_{i\notin P} w_i \right) - \sum_{P\in P^{\hat{T}}} \left( \prod_{i\in P} w_i - \prod_{i\notin P} w_i \right) \right).
\end{aligned}$$

We now show how to compute the two sums in the final term using the SUM-PARTITION oracle. We will give the procedure for computing

$$\sum_{P\in P^T} \left( \prod_{i\in P} w_i - \prod_{i\notin P} w_i \right)$$

and the case with $\hat{T}$ will follow by symmetry. The oracle returns a real number, so by our model of computing real valued functions, we will also give the oracle an additional input that specifies the number of bits of precision in its output. Ultimately we only need to approximate each sum to within $\pm T/4$. This will give an approximation to the #T-PARTITION problem to within $\pm 1/2$, thereby solving it by rounding the approximation because the solution will be an integer. We want to set the input $r$ to the SUM-PARTITION oracle to be $r = r_T$ such that for all $P \subseteq [k]$, we have

$$\prod_{i\in P} w_i - r_T \cdot \prod_{i\notin P} w_i \geq 0 \iff \prod_{i\in P} w_i - \prod_{i\notin P} w_i \geq T. \tag{4.1}$$

Taking $w = \prod_{i\in[k]} w_i$ and thinking of $v = \prod_{i\in P} w_i$, it suffices that all positive solutions to each of the following two inequalities are the same.

$$v - r_T \frac{w}{v} \geq 0 \qquad \text{and} \qquad v - \frac{w}{v} \geq T.$$

The positive solutions to the left one are $v \geq \sqrt{r_T w}$, and to the right one are $v \geq (T + \sqrt{T^2 + 4w})/2$. Setting the right-hand sides equal gives

$$r_T = \frac{\left(T + \sqrt{T^2 + 4w}\right)^2}{4w}. \tag{4.2}$$

Since $r_T$ might be irrational and SUM-PARTITION takes as input rational values of $r$, we need to find a rational $r$ that approximates $r_T$ and preserves the set of solutions $P^T$. Recall from Claim 4.14 that there is an (only) exponentially small interval $(T'', T)$ below $T$ such that for all $\bar{T} \in (T'', T)$, $P^T = P^{\bar{T}}$. This translates to a corresponding interval $(r_{T''}, r_T)$ such that for all $r \in (r_{T''}, r_T)$, Equivalence (4.1) holds. Furthermore, this interval is also only exponentially small.

**Claim 4.15.** *$r_T - r_{T''} \geq 2^{-poly(n)}$ where $n$ is the input length (i. e., the bit-lengths of the integers $z_1, \ldots, z_k, t, t'$).*

*Proof of Claim.* To see this, view $r_T$ from Equation (4.2) as a function $r(T)$ of $T$, and calculate the derivative.

$$r'(T) = \frac{\left(T + \sqrt{T^2 + 4w}\right)^2}{2w \cdot \sqrt{T^2 + 4w}}.$$

Fact 4.12 says that

$$r_T - r_{T''} = r(T) - r(T'') \geq \left(\min_{z \in (T'', T)} r'(z)\right) \cdot (T - T'') \geq (T - T'') \cdot \mathrm{poly}(T).$$

(Recall that $1 \leq w = \prod_i w_i \leq c$). This is only exponentially small in the input length by Claim 4.14. □

So we can choose a rational $r \in (r_{T''}, r_T)$ that can be specified with a number of bits that is polynomial in the input length and preserves

$$P^T = \left\{ P \subseteq [k] \;\middle|\; \prod_{i \in P} w_i - r \cdot \prod_{i \notin P} w_i \geq 0 \right\}.$$

However the SUM-PARTITION oracle gives us

$$\sum_{P \subseteq [k]} \max \left\{ \prod_{i \in P} w_i - r \cdot \prod_{i \notin P} w_i, 0 \right\} = \sum_{P \in P^T} \left( \prod_{i \in P} w_i - r \cdot \prod_{i \notin P} w_i \right),$$

whereas we want to compute the right-hand side without the $r$ coefficient. To get this we just pick another rational $r' \in (r_{T''}, r_T)$ such that $r' - r \geq 2^{-\mathrm{poly}(n)}$. If precision were not an issue, we could run our

SUM-PARTITION oracle for $r$ and $r'$ and receive the output

$$S_1 = \sum_{P \in P^T} \left( \prod_{i \in P} w_i - r \cdot \prod_{i \notin P} w_i \right),$$

$$S_2 = \sum_{P \in P^T} \left( \prod_{i \in P} w_i - r' \cdot \prod_{i \notin P} w_i \right).$$

Then the following linear combination of $S_1$ and $S_2$ gives us what we want.

$$\frac{r'-1}{r'-r} \cdot S_1 - \frac{r-1}{r'-r} \cdot S_2 = \sum_{P \in P^T} \left( \prod_{i \in P} w_i - \prod_{i \notin P} w_i \right).$$

**Claim 4.16.** *Computing $S_1$ and $S_2$ to within $\pm 2^{-poly(n)}$ yields an approximation of*

$$\sum_{P \in P^T} \left( \prod_{i \in P} w_i - \prod_{i \notin P} w_i \right)$$

*to within $\pm T/4$.*

*Proof of Claim.* We just need to approximate $S_1$ and $S_2$ to within

$$\pm \frac{T}{8} \cdot \frac{r'-r}{r'-1}$$

to get the desired precision. This additive error is only exponentially small by Claim 4.15. □

Running this whole procedure again for $\hat{T} \in (T, T')$, which we fixed above gives us all the information we need to count the solutions to the #T-PARTITION instance we were given. We can solve #T-PARTITION in polynomial time with four calls to a SUM-PARTITION oracle. Therefore SUM-PARTITION is #P-hard. □

Now we prove that computing OptComp is #P-complete.

*Proof of Theorem 1.6.* We have already shown that computing OptComp is #P-easy. Here we prove that it is also #P-hard, thereby proving #P-completeness.

We are given an instance $D$, $W = \{w_1, \ldots, w_k\}, r \in \mathbb{Q}$, and $q$ of SUM-PARTITION, where $\forall i \in [k]$, $w_i$ is the $D$-th root of a corresponding integer $z_i$, $\prod_i w_i \leq c$, and $q$ specifies the desired number of bits of precision in the output. If we disregard precision, we would like to set $\varepsilon_i = \ln(w_i)\ \forall i \in [k]$, $\delta_1 = \delta_2 = \ldots \delta_k = 0$ and $\varepsilon_g = \ln(r)$. Note that $\sum_i \varepsilon_i = \ln\left(\prod_i w_i\right) \leq \ln(c)$. Since we can take $c$ to be an arbitrary constant greater than 1, we can ensure that $\sum_i \varepsilon_i \leq \varepsilon$ for an arbitrary $\varepsilon > 0$.

Again we will use the version of OptComp that takes $\varepsilon_g$ as input and outputs $\delta_g$. After using an OptComp oracle to find $\delta_g$ we know the optimal composition Equation (1.2) from Theorem 1.5 is satisfied.

$$\frac{1}{\prod_{i=1}^{k}(1+\mathrm{e}^{\varepsilon_i})} \sum_{S \subseteq \{1,\ldots,k\}} \max\left\{ \mathrm{e}^{\sum\limits_{i \in S} \varepsilon_i} - \mathrm{e}^{\varepsilon_g} \cdot \mathrm{e}^{\sum\limits_{i \notin S} \varepsilon_i}, 0 \right\} = 1 - \frac{1-\delta_g}{\prod_{i=1}^{k}(1-\delta_i)} = \delta_g.$$

Thus we can compute

$$\delta_g \cdot \prod_{i=1}^{k}(1+\mathrm{e}^{\varepsilon_i}) = \sum_{S\subseteq\{1,\dots,k\}} \max\left\{\mathrm{e}^{\sum\limits_{i\in S}\varepsilon_i} - \mathrm{e}^{\varepsilon_g}\cdot \mathrm{e}^{\sum\limits_{i\notin S}\varepsilon_i}, 0\right\}$$
$$= \sum_{S\subseteq\{1,\dots,k\}} \max\left\{\prod_{i\in S} w_i - r\cdot\prod_{i\notin S} w_i, 0\right\}.$$

This last expression is exactly the solution to the instance of SUM-PARTITION we were given. Taking precision into account, the input SUM-PARTITION instance has an additional input $q$ that specifies the desired number of bits of precision in the output and we can only pass OptComp rational values so we will have to approximate $\varepsilon_i = \ln(w_i)$ for all $i$ and $\varepsilon_g = \ln(r)$. Again there is a worry that when we approximate these values the set of partitions $S$ that make

$$\prod_{i\in S} w_i - r\cdot\prod_{i\notin S} w_i > 0$$

might change. We want to get enough precision in our inputs so that the set of partitions over which we sum does not change and enough precision so that the output is accurate to $q$ bits. We will calculate the approximations required for each of these two goals separately and the final precision that we use will just be the maximum of the two. We prove that we can achieve both of these goals with the next two claims.

**Claim 4.17.** *There exists a polynomial $p(n)$ in the length $n$ of the input (the bit-lengths of $z_1,\dots,z_k,q$, and the numerator and denominator of $r$) such that if $|w_i - w_i'| \le 2^{-p(n)}$ for each $i$, then the set of partitions $S$ satisfying*

$$\prod_{i\in S} w_i - r\cdot\prod_{i\notin S} w_i > 0$$

*is the same as the set of partitions satisfying*

$$\prod_{i\in S} w_i' - r\cdot\prod_{i\notin S} w_i' > 0\,.$$

*Proof of Claim.* Recall that SUM-PARTITION is #P-hard even on instances where there are no partitions $S$ such that

$$\prod_{i\in S} w_i = r\cdot\prod_{i\notin S} w_i$$

so we may assume our input instance of SUM-PARTITION has no such partitions and still prove the hardness of OptComp. So to ensure that we have enough precision such that the set over which we sum does not change, we must make the error smaller than the minimum possible (in absolute value) nonzero outcome of

$$\prod_{i\in S} w_i - r\cdot\prod_{i\notin S} w_i\,.$$

We now bound this quantity. Let

$$\mathcal{S} = \left\{ S\subseteq[k] \;\middle|\; \prod_{i\in S} w_i \neq \prod_{i\notin S} w_i \right\}.$$

Since $r$ is rational, $r = a/b$ for two integers $a$ and $b$. Let $a' = a^D$ and $b' = b^D$.

$$\begin{aligned}\min_{S\in\mathcal{S}}\left\{\left|\prod_{i\in S} w_i - r\cdot\prod_{i\notin S} w_i\right|\right\} &= \min_{S\in\mathcal{S}}\left\{\left|\left(\prod_{i\in S} z_i\right)^{\frac{1}{D}} - \left(\frac{a'}{b'}\prod_{i\notin S} z_i\right)^{\frac{1}{D}}\right|\right\}\\ &\geq \min_{S\in\mathcal{S}}\left\{\left|\prod_{i\in S} z_i - \frac{a'}{b'}\prod_{i\notin S} z_i\right| \cdot \frac{1}{D\left(\prod_{i\in[k]} z_i\right)^{(D-1)/D}}\right\}.\end{aligned}$$

Where the last line follows from Fact 4.12 applied to the function $f(x) = x^{1/D}$.

$$1/\left(\prod_{i\in[k]} z_i\right)^{(D-1)/D}$$

is only exponentially small because $\prod_{i\in[k]} z_i$ is at most exponentially large in the bit-length of the integers $z_1,\ldots,z_k$. We claim that

$$\left|\prod_{i\in S} z_i - \frac{a'}{b'}\prod_{i\notin S} z_i\right|$$

is at least $1/b'$ for all $S\in\mathcal{S}$. Fix $S\in\mathcal{S}$.

$$\begin{aligned}\left|\prod_{i\in S} z_i - \frac{a'}{b'}\prod_{i\notin S} z_i\right| = h \implies \left|b'\cdot\prod_{i\in S} z_i - a'\cdot\prod_{i\notin S} z_i\right| = h\cdot b'\\ \implies h \geq 1/b'.\end{aligned}$$

Where the last implication follows because

$$b'\cdot\prod_{i\in S} z_i - a'\cdot\prod_{i\notin S} z_i$$

is just a difference of integers so the closest nonzero value it can take on is $\pm 1$. □

**Claim 4.18.** *There exists a polynomial $p(n)$ in the length $n$ of the input (the bit-lengths of $z_1,\ldots,z_k,q$, and the numerator and denominator of $r$) such that if $|w_i - w_i'| \leq 2^{-p(n)}$ for each $i$, then*

$$\left|\sum_{S\subseteq\{1,\ldots,k\}} \max\left\{\prod_{i\in S} w_i' - r\cdot\prod_{i\notin S} w_i', 0\right\} - \sum_{S\subseteq\{1,\ldots,k\}} \max\left\{\prod_{i\in S} w_i - r\cdot\prod_{i\notin S} w_i, 0\right\}\right| \leq 2^{-q}.$$

*Proof of Claim.* We will choose $p(n) = p_1(n) + p_2(n)$ where $p_1(n)$ is the polynomial that exists from Claim 4.17 and $p_2(n)$ will be determined later. Define

$$S^+ = \left\{S \subseteq [k] \mid \prod_{i\in S} w_i - r\cdot\prod_{i\notin S} w_i > 0\right\}.$$

Claim 4.17 says that

$$S^+ = \left\{ S \subseteq [k] \mid \prod_{i \in S} w_i' - r \cdot \prod_{i \notin S} w_i' > 0 \right\}.$$

Now we can write

$$\left| \sum_{S \subseteq \{1,\ldots,k\}} \max\left\{ \prod_{i \in S} w_i' - r \cdot \prod_{i \notin S} w_i', 0 \right\} - \sum_{S \subseteq \{1,\ldots,k\}} \max\left\{ \prod_{i \in S} w_i - r \cdot \prod_{i \notin S} w_i, 0 \right\} \right| =$$
$$\left| \sum_{S \in S^+} \left( \prod_{i \in S} w_i' - r \cdot \prod_{i \notin S} w_i' \right) - \sum_{S \in S^+} \left( \prod_{i \in S} w_i - r \cdot \prod_{i \notin S} w_i \right) \right| =$$
$$\left| \sum_{S \in S^+} \left( \prod_{i \in S} w_i' - \prod_{i \in S} w_i \right) - \sum_{S \in S^+} r \cdot \left( \prod_{i \notin S} w_i' - \prod_{i \notin S} w_i \right) \right| \leq$$
$$\left| \sum_{S \in S^+} \left( \prod_{i \in S} w_i' - \prod_{i \in S} w_i \right) \right| + \left| \sum_{S \in S^+} r \cdot \left( \prod_{i \notin S} w_i' - \prod_{i \notin S} w_i \right) \right|.$$

Bounding each term in the final expression above by $2^{-(q+1)}$ then gives us the accuracy we want. We will show directly how to bound the second term and the argument for the first term follows symmetrically. By hypothesis we have that for all $S \subseteq [k]$,

$$\prod_{i \notin S} w_i' \leq \prod_{i \notin S} \left( w_i + 2^{-p(n)} \right) \leq \prod_{i \notin S} \left( 1 + 2^{-p(n)} \right) w_i \leq \left( 1 + 2^{-p(n)} \right)^k \cdot \prod_{i \notin S} w_i$$

and similarly

$$\prod_{i \notin S} w_i' \geq \left( 1 - 2^{-p(n)} \right)^k \cdot \prod_{i \notin S} w_i .$$

It follows that for all $S \subseteq [k]$,

$$\left( \left( 1 - 2^{-p(n)} \right)^k - 1 \right) \cdot \prod_{i \notin S} w_i \leq \left( \prod_{i \notin S} w_i' - \prod_{i \notin S} w_i \right) \leq \left( \left( 1 + 2^{-p(n)} \right)^k - 1 \right) \cdot \prod_{i \notin S} w_i .$$

Since $|S^+| \leq 2^k$ and $1 \leq \prod_{i \notin S} w_i \leq c$ for all $S$ we get

$$2^k \cdot r \cdot \left( \left( 1 - 2^{-p(n)} \right)^k - 1 \right) \cdot \leq \sum_{S \in S^+} r \cdot \left( \prod_{i \notin S} w_i' - \prod_{i \notin S} w_i \right) \leq 2^k \cdot r \cdot \left( \left( 1 + 2^{-p(n)} \right)^k - 1 \right) \cdot c .$$

Picking $p_2(n)$ such that $p(n) = p_1(n) + p_2(n) > 2k + \log(rc) + q + 1$ then suffices to bound the absolute value of the sum by $2^{-(q+1)}$. Repeating the same calculation for

$$\sum_{S \in S^+} \left( \prod_{i \in S} w_i' - \prod_{i \in S} w_i \right)$$

will yield the same approximation except without the factor of $r$. So we can bound both terms by $2^{-(q+1)}$ (and therefore their sum by $2^{-q}$) by approximating each $w_i$ to a precision that is polynomial in $n$, which proves the claim. $\square$

So by the two claims above we can get an approximation of the SUM-PARTITION instance to $q$ bits of precision in polynomial time with access to an OptComp oracle. Therefore computing OptComp is #P-hard. $\square$

# 5 Approximation of OptComp

Although we cannot hope to efficiently compute the optimal composition for a general set of differentially private algorithms (assuming $\mathsf{P} \neq \mathsf{NP}$ or even $\mathsf{FP} \neq \#\mathsf{P}$), we show in this section that we can approximate OptComp to arbitrary precision in polynomial time.

**Theorem 1.7 (restated).** *There is a polynomial-time algorithm that given rational $\varepsilon_1, \ldots, \varepsilon_k \geq 0$, $\delta_1, \ldots, \delta_k, \delta_g \in [0,1)$, and $\eta \in (0,1)$, outputs $\varepsilon^*$ satisfying*

$$\mathrm{OptComp}((\varepsilon_1,\delta_1),\ldots,(\varepsilon_k,\delta_k),\delta_g) \leq \varepsilon^* \leq \mathrm{OptComp}((\varepsilon_1,\delta_1),\ldots,(\varepsilon_k,\delta_k),\mathrm{e}^{-\eta/2}\cdot\delta_g)+\eta.$$

*The algorithm runs in time*

$$O\left(\frac{k^3\cdot\overline{\varepsilon}\cdot(1+\overline{\varepsilon})}{\eta}\cdot\log\left(\frac{k^2\cdot\overline{\varepsilon}\cdot(1+\overline{\varepsilon})}{\eta}\right)\right)$$

*where $\overline{\varepsilon} = \sum_{i\in[k]}\varepsilon_i/k$, assuming constant-time arithmetic operations.*

We prove Theorem 1.7 using the following three lemmas:

**Lemma 5.1.** *Given non-negative integers $a_1, \ldots, a_k$, $B$ and weights $w_1, \ldots, w_k \in \mathbb{Q}$, one can compute*

$$\sum_{\substack{S\subseteq[k] \text{ s.t.}\\ \sum_{i\in S} a_i \leq B}} \prod_{i\in S} w_i$$

*in time $O(Bk)$.*

Notice that the constraint in Lemma 5.1 is the same one that characterizes knapsack problems. Indeed, the algorithm we give for computing $\sum_{S\subseteq[k]}\prod_{i\in S} w_i$ is a slight modification of the known pseudo-polynomial time algorithm for counting knapsack solutions, which uses dynamic programming. Next we show that we can use this algorithm to approximate OptComp.

**Lemma 5.2.** *Given a rational $\mathrm{e}^{\varepsilon_0}$ with $\varepsilon_0 \geq 0$ and $\varepsilon_1 = a_1\cdot\varepsilon_0, \ldots, \varepsilon_k = a_k\cdot\varepsilon_0, \varepsilon^* = a^*\cdot\varepsilon_0$ for positive integers $a_1, \ldots, a_k, a^*$ (given as input), and rational $\delta_1, \ldots \delta_k, \delta_g \in [0,1)$, there is an algorithm that determines whether or not* $\mathrm{OptComp}((\varepsilon_1,\delta_1),\ldots,(\varepsilon_k,\delta_k),\delta_g) \leq \varepsilon^*$ *and runs in time*

$$O\left(k\cdot\sum_{i=1}^{k} a_i\right)$$

*assuming constant-time arithmetic operations.*

In other words, if the $\varepsilon$ values we are given are all integer multiples of some $\varepsilon_0$ where $e^{\varepsilon_0}$ is rational, we can determine whether or not the composition of those privacy parameters is $(a^* \cdot \varepsilon_0, \delta_g)$-DP in pseudo-polynomial time, for every positive integer $a^*$. Running binary search over integers $a^*$, we can find the minimum such integer. When $\varepsilon_0$ is small, this gives us a good overestimate of the optimal composition of the discrete input privacy parameters. This means that given any inputs $(\varepsilon_1, \delta_1), \ldots, (\varepsilon_k, \delta_k), \delta_g$ to OptComp, we can discretize and polynomially bound the $\varepsilon_i$ values to new values $\varepsilon_i'$ for all $i \in [k]$ and use Lemma 5.2 to approximate $\mathrm{OptComp}((\varepsilon_1', \delta_1), \ldots, (\varepsilon_k', \delta_k), \delta_g)$. The next lemma tells us that this is also a good approximation of $\mathrm{OptComp}((\varepsilon_1, \delta_1), \ldots, (\varepsilon_k, \delta_k), \delta_g)$.

**Lemma 5.3.** *For all $\varepsilon_1, \ldots, \varepsilon_k, c_1, \ldots, c_k \geq 0$ and $\delta_1, \ldots, \delta_k, \delta_g \in [0,1)$,*

$$\mathrm{OptComp}((\varepsilon_1 + c_1, \delta_1), \ldots, (\varepsilon_k + c_k, \delta_k), \delta_g) \leq \mathrm{OptComp}((\varepsilon_1, \delta_1), \ldots, (\varepsilon_k, \delta_k), e^{-c/2} \cdot \delta_g) + c$$

*where $c = \sum_{i=1}^k c_i$.*

Next we prove the three lemmas and then show that Theorem 1.7 follows.

*Proof of Lemma 5.1.* We modify Dyer's algorithm for approximately counting solutions to knapsack problems [8]. The algorithm uses dynamic programming. Given non-negative integers $a_1, \ldots, a_k$, $B$, and weights $w_1, \ldots, w_k \in \mathbb{Q}$, define

$$F(r,s) = \sum_{\substack{S \subseteq [r] \text{ s.t.} \\ \sum_{i \in S} a_i \leq s}} \prod_{i \in S} w_i \,.$$

We want to compute $F(k,B)$, which we can do by tabulating $F(r,s)$ for $(0 \leq r \leq k,\ 0 \leq s \leq B)$ using the following recursion.

$$F(r,s) = \begin{cases} 1 & \text{if } r = 0, \\ F(r-1,s) + w_r F(r-1, s-a_r) & \text{if } r > 0 \text{ and } a_r \leq s, \\ F(r-1,s) & \text{if } r > 0 \text{ and } a_r > s. \end{cases}$$

Each cell $F(r,s)$ in the table can be computed in constant time given earlier cells $F(r',s')$ where $r' < r$. Thus filling the entire table takes time $O(Bk)$. □

*Proof of Lemma 5.2.* Given a rational $e^{\varepsilon_0} \geq 0$ and $\varepsilon_1 = a_1 \cdot \varepsilon_0, \ldots, \varepsilon_k = a_k \cdot \varepsilon_0, \varepsilon^* = a^* \cdot \varepsilon_0$ for positive integers $a_1, \ldots, a_k, a^*$ and rational $\delta_1, \ldots \delta_k, \delta_g \in [0,1)$ Theorem 1.5 tells us that answering whether or not

$$\mathrm{OptComp}((\varepsilon_1, \delta_1), \ldots, (\varepsilon_k, \delta_k), \delta_g) \leq \varepsilon^*$$

is equivalent to answering whether or not the following inequality holds.

$$\frac{1}{\prod_{i=1}^k (1 + e^{\varepsilon_i})} \sum_{S \subseteq \{1,\ldots,k\}} \max\left\{ e^{\sum_{i \in S} \varepsilon_i} - e^{\varepsilon^*} \cdot e^{\sum_{i \notin S} \varepsilon_i}, 0 \right\} \leq 1 - \frac{1 - \delta_g}{\prod_{i=1}^k (1 - \delta_i)} \,. \tag{5.1}$$

The right-hand side and $\prod_{i=1}^{k}(1+\mathrm{e}^{\varepsilon_i})$ are easy to compute given the inputs (note that $\mathrm{e}^{\varepsilon_i}$ is rational for all $i\in[k]$ because each is an integer power of $\mathrm{e}^{\varepsilon_0}$). So in order to check the inequality, we will show how to compute the sum. Define

$$\begin{aligned} K &= \left\{ T\subseteq[k] \mid \sum_{i\notin T}\varepsilon_i \geq \varepsilon^* + \sum_{i\in T}\varepsilon_i \right\} \\ &= \left\{ T\subseteq[k] \mid \sum_{i\in T}\varepsilon_i \leq \left(\sum_{i=1}^{k}\varepsilon_i - \varepsilon^*\right)/2 \right\} \\ &= \left\{ T\subseteq[k] \mid \sum_{i\in T} a_i \leq B \right\} \text{ for } B = \left\lfloor \left(\sum_{i=1}^{k} a_i - a^*\right)/2 \right\rfloor \end{aligned}$$

and observe that by setting $T=S^{\mathrm{c}}$, we have

$$\sum_{S\subseteq\{1,\dots,k\}} \max\left\{ \mathrm{e}^{\sum\limits_{i\in S}\varepsilon_i} - \mathrm{e}^{\varepsilon^*}\cdot \mathrm{e}^{\sum\limits_{i\notin S}\varepsilon_i}, 0 \right\} = \sum_{T\in K}\left( \left( \prod_{i=1}^{k}\mathrm{e}^{\varepsilon_i}\cdot\prod_{i\in T}\mathrm{e}^{-\varepsilon_i}\right) - \left(\mathrm{e}^{\varepsilon^*}\cdot\prod_{i\in T}\mathrm{e}^{\varepsilon_i}\right)\right).$$

We can now use Lemma 5.1 to compute each term separately since $K$ is a set of knapsack solutions. Specifically, setting $w_i=\mathrm{e}^{-\varepsilon_i}\ \forall i\in[k]$, Lemma 5.1 tells us that we can compute $\sum_{T\subseteq[k]}\prod_{i\in T}w_i$ subject to $\sum_{i\in T}a_i\leq B$, which is equivalent to

$$\sum_{T\in K}\prod_{i\in T}\mathrm{e}^{-\varepsilon_i}.$$

To compute $\sum_{T\in K}\prod_{i\in T}\mathrm{e}^{\varepsilon_i}$, we instead set $w_i=\mathrm{e}^{\varepsilon_i}$ and run the same procedure. (Note that $\mathrm{e}^{\varepsilon^*}=(\mathrm{e}^{\varepsilon_0})^{a^*}$, which is rational.) So we can determine whether or not Inequality (5.1) holds. We used the algorithm from Lemma 5.1 so the running time is

$$O(Bk)=O\left(k\cdot\sum_{i=1}^{k}a_i\right). \qquad \square$$

*Proof of Lemma 5.3.* Fix $\varepsilon_1,\dots,\varepsilon_k,c_1,\dots,c_k\geq 0$ and $\delta_1,\dots,\delta_k,\delta_g\in[0,1)$ and let $c=\sum_{i\in[k]}c_i$. Let $\mathrm{OptComp}((\varepsilon_1,\delta_1),\dots,(\varepsilon_k,\delta_k),\mathrm{e}^{-c/2}\cdot\delta_g)=\varepsilon_g$. From Equation (1.2) in Theorem 1.5 we know

$$\frac{1}{\prod_{i=1}^{k}(1+\mathrm{e}^{\varepsilon_i})}\sum_{S\subseteq\{1,\dots,k\}}\max\left\{\mathrm{e}^{\sum\limits_{i\in S}\varepsilon_i}-\mathrm{e}^{\varepsilon_g}\cdot\mathrm{e}^{\sum\limits_{i\notin S}\varepsilon_i},0\right\}\leq 1-\frac{1-\mathrm{e}^{-c/2}\cdot\delta_g}{\prod_{i=1}^{k}(1-\delta_i)}.$$

Multiplying both sides by $\mathrm{e}^{c/2}$ gives

$$\begin{aligned}\frac{\mathrm{e}^{c/2}}{\prod_{i=1}^{k}(1+\mathrm{e}^{\varepsilon_i})}\sum_{S\subseteq\{1,\dots,k\}}\max\left\{\mathrm{e}^{\sum\limits_{i\in S}\varepsilon_i}-\mathrm{e}^{\varepsilon_g}\cdot\mathrm{e}^{\sum\limits_{i\notin S}\varepsilon_i},0\right\} &\leq \mathrm{e}^{c/2}\cdot\left(1-\frac{1-\mathrm{e}^{-c/2}\cdot\delta_g}{\prod_{i=1}^{k}(1-\delta_i)}\right)\\ &\leq 1-\frac{1-\delta_g}{\prod_{i=1}^{k}(1-\delta_i)}.\end{aligned}$$

The above inequality together with Theorem 1.5 means that showing the following will complete the proof.

$$\sum_{S\subseteq\{1,\ldots,k\}} \max\left\{ e^{\sum_{i\in S}(\varepsilon_i+c_i)} - e^{\varepsilon_g+c}\cdot e^{\sum_{i\notin S}(\varepsilon_i+c_i)}, 0\right\} \le \frac{e^{c/2}\cdot\prod_{i=1}^{k}\left(1+e^{\varepsilon_i+c_i}\right)}{\prod_{i=1}^{k}\left(1+e^{\varepsilon_i}\right)} \sum_{S\subseteq\{1,\ldots,k\}} \max\left\{ e^{\sum_{i\in S}\varepsilon_i} - e^{\varepsilon_g}\cdot e^{\sum_{i\notin S}\varepsilon_i}, 0\right\}.$$

Since $(1+e^{\varepsilon_i+c_i})/(1+e^{\varepsilon_i}) \ge e^{c_i/2}$ for every $\varepsilon_i, c_i > 0$, it suffices to show

$$\sum_{S\subseteq\{1,\ldots,k\}} \max\left\{ e^{\sum_{i\in S}(\varepsilon_i+c_i)} - e^{\varepsilon_g+c}\cdot e^{\sum_{i\notin S}(\varepsilon_i+c_i)}, 0\right\} \le \sum_{S\subseteq\{1,\ldots,k\}} e^{c}\cdot \max\left\{ e^{\sum_{i\in S}\varepsilon_i} - e^{\varepsilon_g}\cdot e^{\sum_{i\notin S}\varepsilon_i}, 0\right\}.$$

This inequality holds term by term. If a right-hand term is zero

$$\left(\sum_{i\in S}\varepsilon_i \le \varepsilon_g + \sum_{i\notin S}\varepsilon_i\right),$$

then so is the corresponding left-hand term

$$\left(\sum_{i\in S}(\varepsilon_i+c_i) \le \varepsilon_g + c + \sum_{i\notin S}(\varepsilon_i+c_i)\right).$$

For the nonzero terms, the factor of $e^c$ ensures that the right-hand terms are larger than the left-hand terms. □

*Proof of Theorem 1.7.* Lemma 5.2 tells us that we can determine whether a set of privacy parameters satisfies some $(\varepsilon_g, \delta_g)$ differential privacy guarantee if the $\varepsilon_i$ values and $\varepsilon_g$ are all positive integer multiples of some $\varepsilon_0$ where $e^{\varepsilon_0}$ is rational. We are given rational $\varepsilon_1, \ldots, \varepsilon_k \ge 0, \delta_1, \ldots \delta_k, \delta_g \in [0,1)$, and $\eta \in (0,1)$. Let $\overline{\varepsilon} = \sum_{i\in[k]}\varepsilon_i/k$ be the arithmetic mean of the $\varepsilon_i$ values. Let $\beta = \eta/(k\cdot(1+\overline{\varepsilon})+1)$, set $\varepsilon_0 = \ln(1+\beta)$, and for all $i \in [k]$ set $a_i = \lceil \varepsilon_i\cdot(1/\beta+1)\rceil$ and $\varepsilon_i' = \varepsilon_0\cdot a_i$. We will use the following bounds on $\varepsilon_0$ in the proof.

$$\frac{\beta}{2} \le \frac{\beta}{1+\beta} \le \varepsilon_0 \le \beta.$$

With these settings, the $a_i$ are non-negative integers, the $\varepsilon_i'$ values are all integer multiples of $\varepsilon_0$ and $e^{\varepsilon_0}$ is rational. So for every positive integer $a$ we can apply Lemma 5.2 to determine whether or not $\mathrm{OptComp}((\varepsilon_1', \delta_1), \ldots, (\varepsilon_k', \delta_k), \delta_g) \le a\cdot\varepsilon_0$ in time $O\left(k\cdot\sum_{i\in[k]}a_i\right)$. Running binary search over integers $a$, we can find the minimum such integer, which we will call $a^*$. The algorithm's estimate of $\mathrm{OptComp}((\varepsilon_1, \delta_1), \ldots, (\varepsilon_k, \delta_k), \delta_g)$ will be $a^*\cdot\varepsilon_0$. However since this number is irrational, we will use the Taylor approximation of the natural logarithm to output $\varepsilon^*$ satisfying $a^*\cdot\varepsilon_0 \le \varepsilon^* \le a^*\cdot\varepsilon_0 + \beta - \varepsilon_0$. Since we only need to calculate a few terms of the Taylor expansion of $\ln(1+\beta)$ to achieve this approximation, this step will not affect our running time. The pseudocode for this procedure is below.

**Input:** $\varepsilon_1,\ldots,\varepsilon_k \ge 0$, $\delta_1,\ldots,\delta_k,\delta_g \in [0,1)$, $\eta \in (0,1)$
**Output:** Approximation of $\mathrm{OptComp}((\varepsilon_1,\delta_1),\ldots,(\varepsilon_k,\delta_k),\delta_g)$
$\overline{\varepsilon} \leftarrow \sum_{i\in[k]} \varepsilon_i/k$
$\beta \leftarrow \eta/(k\cdot(1+\overline{\varepsilon})+1)$
$\varepsilon_0 \leftarrow \ln(1+\beta)$
**for** $i \in [k]$ **do**
  $a_i \leftarrow \lceil \varepsilon_i \cdot (1/\beta+1)\rceil$
  $\varepsilon_i' \leftarrow \varepsilon_0 \cdot a_i$
**end for**
Use binary search and Lemma 5.2 to find minimum integer $a^*$ such that

$$\mathrm{OptComp}((\varepsilon_1',\delta_1),\ldots,(\varepsilon_k',\delta_k),\delta_g) \le a^*\cdot\varepsilon_0$$

**return** Taylor approximation to $a^*\cdot\varepsilon_0$

Since we choose $a^*$ to be the minimum integer satisfying composition we have

$$\varepsilon^* - \beta \le (a^*-1)\cdot\varepsilon_0 \le \mathrm{OptComp}((\varepsilon_1',\delta_1),\ldots,(\varepsilon_k',\delta_k),\delta_g) \le a^*\cdot\varepsilon_0 \le \varepsilon^*\,.$$

$a^*$ can range from 0 to $\sum_{i\in[k]} a_i$ so the binary search can be done in

$$\log\left(\sum_{i\in[k]} a_i\right) = \log O\left(k^2\cdot\overline{\varepsilon}\cdot(1+\overline{\varepsilon})/\eta\right)$$

iterations. This gives us a total running time of

$$O\left(\frac{k^3\cdot\overline{\varepsilon}\cdot(1+\overline{\varepsilon})}{\eta}\cdot\log\left(\frac{k^2\cdot\overline{\varepsilon}\cdot(1+\overline{\varepsilon})}{\eta}\right)\right).$$

Now we argue that $\varepsilon^*$ is a good approximation of $\mathrm{OptComp}((\varepsilon_1,\delta_1),\ldots,(\varepsilon_k,\delta_k),\delta_g)$. For all $i\in[k]$ we have

$$\varepsilon_i' = \varepsilon_0\cdot a_i \ge \frac{\beta}{1+\beta}\cdot\left\lceil \varepsilon_i\cdot\left(\frac{1}{\beta}+1\right)\right\rceil \ge \varepsilon_i\,.$$

So all of the $\varepsilon_i'$ values are overestimates of their corresponding $\varepsilon_i$ values and therefore

$$\mathrm{OptComp}((\varepsilon_1,\delta_1),\ldots,(\varepsilon_k,\delta_k),\delta_g) \le \mathrm{OptComp}((\varepsilon_1',\delta_1),\ldots,(\varepsilon_k',\delta_k),\delta_g) \le \varepsilon^*$$

satisfying one of the inequalities in the theorem. We also have for all $i\in[k]$

$$\varepsilon_i' = \varepsilon_0\cdot\left\lceil \varepsilon_i\cdot\left(\frac{1}{\beta}+1\right)\right\rceil \le \beta\cdot\left(\varepsilon_i\cdot\left(\frac{1}{\beta}+1\right)+1\right) = \varepsilon_i+\beta\cdot(\varepsilon_i+1)\,.$$

Let $c_i = \beta \cdot (\varepsilon_i + 1)$ for all $i \in [k]$ and let $c = \sum_{i \in [k]} c_i = \beta \cdot k \cdot (1 + \overline{\varepsilon})$. Now we get

$$\begin{aligned} \varepsilon^* - \beta &\leq \mathrm{OptComp}((\varepsilon_1', \delta_1), \ldots, (\varepsilon_k', \delta_k), \delta_g) \\ &\leq \mathrm{OptComp}((\varepsilon_1 + c_1, \delta_1), \ldots, (\varepsilon_k + c_k, \delta_k), \delta_g) \\ &\leq \mathrm{OptComp}((\varepsilon_1, \delta_1), \ldots, (\varepsilon_k, \delta_k), \mathrm{e}^{-\beta \cdot k \cdot (1+\overline{\varepsilon})/2} \cdot \delta_g) + \beta \cdot k \cdot (1 + \overline{\varepsilon}) \end{aligned}$$

by Lemma 5.3. Noting that $\beta \cdot k \cdot (1 + \overline{\varepsilon})$ and $\beta \cdot k \cdot (1 + \overline{\varepsilon}) + \beta$ are both at most $\eta$ completes the proof. □

## 6 Comparison of composition theorems

The plots in Figure 2 below compare the performances of four homogeneous composition theorems. In all figures, "Summing" refers to basic composition—Theorem 1.2 [4], "DRV" refers to advanced composition—Theorem 1.3 [7], "KOV Bound" refers to the bound below [12] that is a closed form approximation of the optimal composition theorem, and "Optimal" refers to the optimal composition theorem—Theorem 1.4 [12].

**Theorem 6.1** (KOV Bound [12])**.** *For every $\varepsilon \geq 0$, $\delta, \tilde{\delta} \in [0,1]$, and $(\varepsilon, \delta)$-differentially private algorithms $M_1, M_2, \ldots, M_k$, the composition $(M_1, M_2, \ldots, M_k)$ satisfies $(\varepsilon_g, 1 - (1-\delta)^k (1 - \tilde{\delta}))$-DP for*

$$\varepsilon_g = \min\left\{ k\varepsilon, \frac{(\mathrm{e}^{\varepsilon} - 1)\varepsilon k}{\mathrm{e}^{\varepsilon} + 1} + \varepsilon \sqrt{2k \log\left(e + \frac{\sqrt{k\varepsilon^2}}{\tilde{\delta}}\right)}, \frac{(\mathrm{e}^{\varepsilon} - 1)\varepsilon k}{\mathrm{e}^{\varepsilon} + 1} + \varepsilon \sqrt{2k \log\left(\frac{1}{\tilde{\delta}}\right)} \right\}.$$

Here we are composing $k$ mechanisms that are $(\varepsilon, \delta)$ differentially private to obtain an $(\varepsilon_g, \delta_g)$ differentially private mechanism as guaranteed by one of the composition theorems. Note that DRV as typically formulated is the only one of the bounds that does not have simple summing as a special case. For that reason the DRV bound actually performs worse than summing for small values of $k$ before the asymptotic improvement kicks in.

## Acknowledgements

We thank Mark Bun, Cynthia Dwork, and Thomas Steinke for helpful comments, particularly regarding our proof of Lemma 3.2.

## References

[1] HAI BRENNER AND KOBBI NISSIM: Impossibility of differentially private universally optimal mechanisms. *SIAM J. Comput.*, 43(5):1513–1540, 2014. Preliminary version in FOCS'10. [doi:10.1137/110846671, arXiv:1008.0256] 5

[2] MARK BUN AND THOMAS STEINKE: Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Proc. 14th Theory of Cryptography Conf. (TCC'16)*, pp. 635–658. Springer, 2016. [doi:10.1007/978-3-662-53641-4_24, arXiv:1605.02065] 7

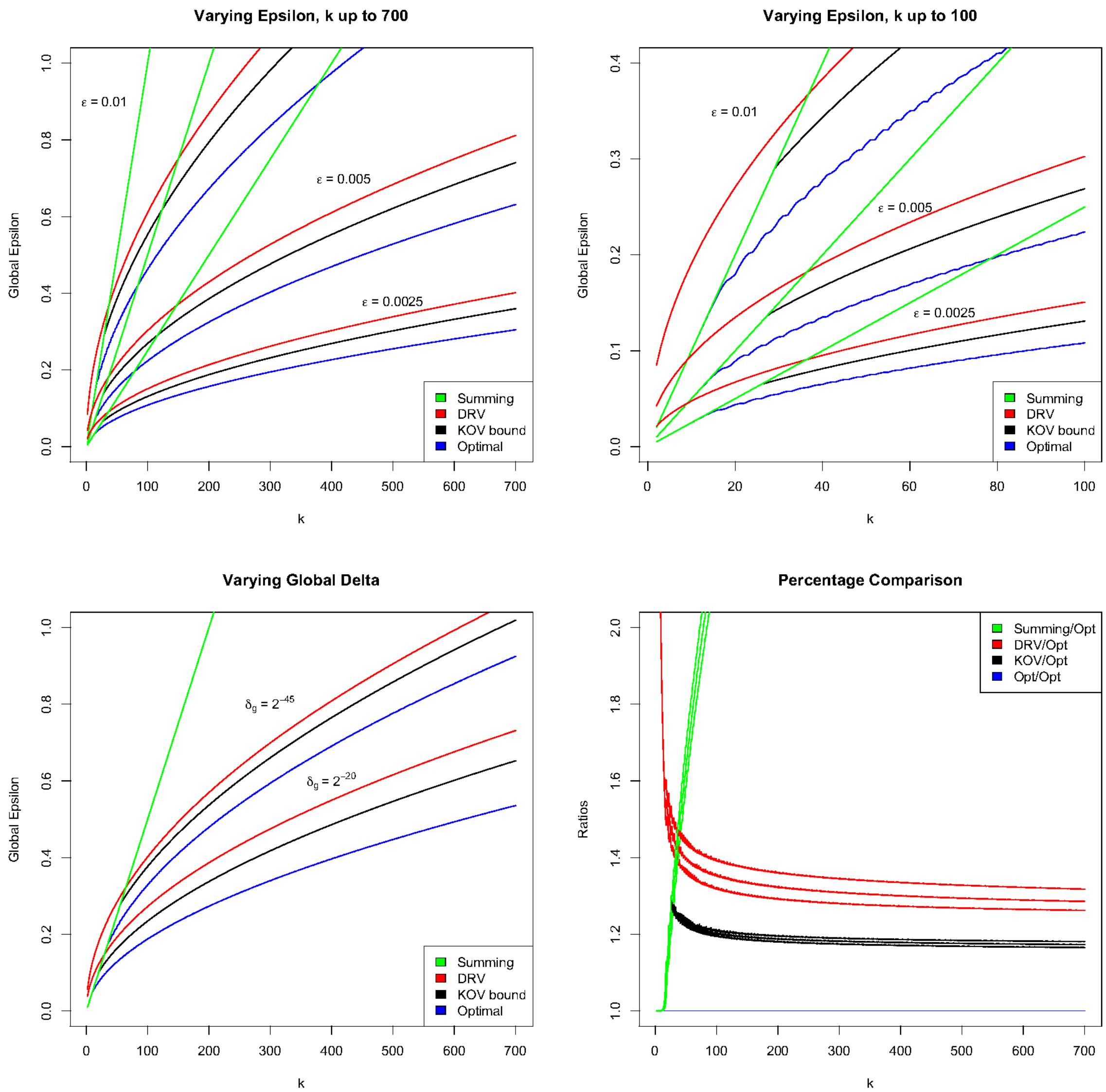


Figure 2: (Top Left) $\varepsilon_g$ given by four composition theorems for varying values of $\varepsilon$ as $k$ grows. Parameters $\delta = 0$ and $\delta_g = 2^{-25}$. (Top Right) Same as Top Left zoomed in on the $k < 100$ regime. We see that optimal composition gives substantial savings in $\varepsilon_g$, even for moderate values of $k$. (Bottom Left) $\varepsilon_g$ given by four composition theorems for varying values of $\delta_g$ as $k$ grows, with parameters $\delta = 0$ and $\varepsilon = .005$ for the individual mechanisms. $\delta_g$ does not affect $\varepsilon_g$ in basic composition. (Bottom Right) Performance of composition theorems measured relative to optimal composition. Depicts every curve in Figure 1 divided by the optimal composition curve. We see that relative performances of the KOV bound and DRV seem to converge to a constant. The $\varepsilon_g$ values given by the KOV bound are about 20% larger than optimal and the values given by advanced composition are about 30-40% larger than optimal.

[3] MERCÈ CROSAS: The dataverse network®: An open-source application for sharing, discovering and preserving data. *D-Lib Magazine*, 17(1/2), 2011. [doi:10.1045/january2011-crosas] 5

[4] CYNTHIA DWORK, KRISHNARAM KENTHAPADI, FRANK MCSHERRY, ILYA MIRONOV, AND MONI NAOR: Our data, ourselves: Privacy via distributed noise generation. In *Proc. 25th Int. Conf. on the Theory and Application of Cryptographic Techniques (EUROCRYPT'06)*, pp. 486–503. Springer, 2006. [doi:10.1007/11761679_29] 2, 32

[5] CYNTHIA DWORK, FRANK MCSHERRY, KOBBI NISSIM, AND ADAM SMITH: Calibrating noise to sensitivity in private data analysis. *J. Privacy and Confidentiality*, 7(3):17–51, 2016. Preliminary version in TCC'06. Available at journal's webpage. 2, 6

[6] CYNTHIA DWORK AND AARON ROTH: The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3-4):211–407, 2014. [doi:10.1561/0400000042] 6

[7] CYNTHIA DWORK, GUY N. ROTHBLUM, AND SALIL P. VADHAN: Boosting and differential privacy. In *Proc. 51st FOCS*, pp. 51–60. IEEE Comp. Soc. Press, 2010. [doi:10.1109/FOCS.2010.12] 2, 3, 6, 7, 32

[8] MARTIN E. DYER: Approximate counting by dynamic programming. In *Proc. 35th STOC*, pp. 693–699. ACM Press, 2003. [doi:10.1145/780542.780643] 28

[9] MATTHIAS EHRGOTT: Approximation algorithms for combinatorial multicriteria optimization problems. *Internat. Trans. Operational Res.*, 7(1):5–31, 2000. [doi:10.1111/j.1475-3995.2000.tb00182.x] 17, 18

[10] MARCO GABOARDI, JAMES HONAKER, GARY KING, JACK MURTAGH, KOBBI NISSIM, JONATHAN ULLMAN, AND SALIL P. VADHAN: PSI (Ψ): A private data sharing interface. *Harvard University Privacy Tools Project*, 2016. [arXiv:1609.04340] 5

[11] ARPITA GHOSH, TIM ROUGHGARDEN, AND MUKUND SUNDARARAJAN: Universally utility-maximizing privacy mechanisms. *SIAM J. Comput.*, 41(6):1673–1693, 2012. Preliminary version in STOC'09. [doi:10.1137/09076828X, arXiv:0811.2841] 5

[12] PETER KAIROUZ, SEWOONG OH, AND PRAMOD VISWANATH: The composition theorem for differential privacy. *IEEE Trans. Inform. Theory*, 63(6):4037–4049, 2017. Preliminary version in ICML'15. [doi:10.1109/TIT.2017.2685505, arXiv:1311.0776] 3, 4, 5, 7, 32

[13] GARY KING: An introduction to the dataverse network as an infrastructure for data sharing. *Sociological Methods & Research*, 36(2):173–199, 2007. [doi:10.1177/0049124107306660] 5

[14] JACK MURTAGH AND SALIL P. VADHAN: The complexity of computing the optimal composition of differential privacy. In *Proc. 14th Theory of Cryptography Conf. (TCC'16)*, pp. 157–175. Springer, 2016. [doi:10.1007/978-3-662-49096-9_7, arXiv:1507.03113v2] 1

[15] STANLEY L. WARNER: Randomized response: A survey technique for eliminating evasive answer bias. *J. Amer. Stat. Assoc.*, 60(309):63–69, 1965. [doi:10.1080/01621459.1965.10480775] 3, 7

## AUTHORS


Jack Murtagh
Graduate student
Harvard University, Cambridge, MA
jmurtagh@g.harvard.edu
https://scholar.harvard.edu/jmurtagh

Salil Vadhan
Vicky Joseph Professor of
Computer Science and Applied Mathematics
Harvard University, Cambridge, MA
salil_vadhan@harvard.edu
http://people.seas.harvard.edu/~salil/


## ABOUT THE AUTHORS

JACK MURTAGH is a graduate student at Harvard University where he is advised by Salil Vadhan. As an undergraduate, Jack studied mathematics at Tufts University. Jack is broadly interested in complexity theory and currently works on derandomization and data privacy.

SALIL VADHAN is the Vicky Joseph Professor of Computer Science and Applied Mathematics at the Harvard John A. Paulson School of Engineering & Applied Sciences. He received his Ph. D. under the supervision of Shafi Goldwasser at MIT in 1999; the title of his dissertation was "A Study of Statistical Zero-Knowledge Proofs." Other research interests include the theory of pseudorandomness the theory and practice of data privacy. He enjoys spending leisure time with his wife and two daughters, as well as learning to surf in the cold waters of New England.