Faster Privacy Accounting via Evolving Discretization

Badih Ghazi [1] **Pritish Kamath** [1] **Ravi Kumar** [1] **Pasin Manurangsi** [1]

Abstract

We introduce a new algorithm for numerical composition of privacy random variables, useful for computing the accurate differential privacy parameters for composition of mechanisms. Our algorithm achieves a running time and memory usage of $\mathrm{polylog}(k)$ for the task of self-composing a mechanism, from a broad class of mechanisms, k times; this class, e.g., includes the sub-sampled Gaussian mechanism, that appears in the analysis of differentially private stochastic gradient descent. By comparison, recent work by Gopi et al. (2021) has obtained a running time of $\widetilde{O}(\sqrt{k})$ for the same task. Our approach extends to the case of composing k different mechanisms in the same class, improving upon their running time and memory usage from $\widetilde{O}(k^{1.5})$ to $\widetilde{O}(k)$.

1. Introduction

Differential privacy (DP) (Dwork et al., 2006b;a) has become the preferred notion of privacy in both academia and the industry. Fueled by the increased awareness and demand for privacy, several systems that use DP mechanisms to guard users' privacy have been deployed in the industry (Erlingsson et al., 2014; Shankland, 2014; Greenberg, 2016; Apple Differential Privacy Team, 2017; Ding et al., 2017; Kenthapadi & Tran, 2018; Rogers et al., 2021), and the US Census (Abowd, 2018). Besides the large volume of data, many of these systems offer real-time private data analytic and inference capabilities, which entail strict computational efficiency requirements on the underlying DP operations.

We recall the definition of DP (Dwork et al., 2006b;a):

Definition 1.1 (DP)**.** Let $\varepsilon > 0$ and $\delta \in [0, 1]$. A randomized algorithm $\mathcal{M} : \mathcal{X}^n \to \mathcal{Y}$ is (ε, δ)-*DP* if, for all $x, x' \in \mathcal{X}^n$ differing on a single index and all outputs $S \subseteq \mathcal{Y}$, we have $\Pr[\mathcal{M}(x) \in S] \leq e^{\varepsilon} \cdot \Pr[\mathcal{M}(x') \in S] + \delta$.

The DP guarantees of a mechanism are captured by the privacy *parameters* ε and δ; the smaller these parameters, the more private the mechanism. Often a mechanism is simultaneously DP for multiple privacy parameters; this is captured by studying the *privacy loss* of a mechanism $\mathcal{M}$—the curve $\delta_{\mathcal{M}}(\cdot)$ for which the mechanism is $(\varepsilon, \delta_{\mathcal{M}}(\varepsilon))$-DP.

A fundamental mathematical property satisfied by DP is *composition*, which prescribes the privacy guarantees of results from executing multiple DP mechanisms. In *basic* composition (Dwork et al., 2006a), a mechanism that returns the results of executing an $(\varepsilon_1, \delta_1)$-DP mechanism and an $(\varepsilon_2, \delta_2)$-DP mechanism is $(\varepsilon_1 + \varepsilon_2, \delta_1 + \delta_2)$-DP. Unfortunately, this bound becomes weak when composing a large number of mechanisms. The *advanced* composition (Dwork et al., 2010) offers stronger bounds: roughly speaking, composing k mechanisms that are each (ε, δ)-DP yields a mechanism whose privacy parameters are of the order of $\sqrt{k}\varepsilon$ and $k\delta$—clearly more desirable than the basic composition. In fact, obtaining tight composition bounds has been an active research topic. Kairouz et al. (2015) showed how to obtain the exact privacy parameters of self-composition (mechanism composed with itself), while Murtagh & Vadhan (2016) showed that the corresponding problem for the more general case is #P-complete.

Privacy Loss Distribution (PLD). Tighter bounds for privacy parameters that go beyond advanced composition are possible if the privacy loss of the mechanism is taken into account. Meiser & Mohammadi (2018); Sommer et al. (2019) initiated the study of numerical methods for accurately estimating the privacy parameters, using the *privacy loss distribution* (PLD) of a mechanism. The PLD is the probability mass function of the so-called *privacy loss random variable* (PRV) for discrete mechanisms, and its density function for continuous mechanisms (see Section 2 for formal definitions). PLDs have two nice properties: (i) tight privacy parameters can be computed from the PLD of a mechanism and (ii) the PLD of a composed mechanism is the convolution of the individual PLDs. Property (ii) makes PLD amenable to FFT-based methods.

[*]Equal contribution [1]Google Research, USA. Correspondence to: Pritish Kamath <pritish@alum.mit.edu>, Pasin Manurangsi <pasin@google.com>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

Koskela et al. (2020) exploited this property to speed up the computation of the PLD of the composition. An important step to retain efficiency using PLDs is approximating the distribution so that it has finite support; this is especially needed in the case when the PLD is continuous or has a large support. PLD implementations have been at the heart of many DP systems and open-source libraries including (Lukas Prediger, 2020; Google, 2020; Microsoft, 2021). To enable scale and support latency considerations, the PLD composition has to be as efficient as possible. This is the primary focus of our paper.

Our starting point is the work of Koskela et al. (2020; 2021); Koskela & Honkela (2021), who derived explicit error bounds for the approximation obtained by the FFT-based algorithm. The running time of this algorithm for k-fold self-composition of a mechanism $\mathcal{M}$ that is $(\varepsilon_0, 0)$-DP is[1] $\widetilde{O}\left(\frac{k^2\varepsilon_0}{\delta_{\text{err}}}\right)$. When $\varepsilon_0 = \frac{1}{\sqrt{k \cdot \log(1/\delta_{\text{err}})}}$, this running time is $\widetilde{O}\left(\frac{k^{1.5}}{\delta_{\text{err}}}\right)$, where δ_{err} is the additive error in δ. Gopi et al. (2021) improved this bound to obtain an algorithm with running time of

$$\widetilde{O}\left(\frac{k^{0.5}\sqrt{\log\frac{1}{\delta_{\text{err}}}}}{\varepsilon_{\text{err}}}\right),$$

where ε_{err} is the additive error in ε. When composing k different mechanisms, their running time is

$$\widetilde{O}\left(\frac{k^{1.5}\sqrt{\log\frac{1}{\delta_{\text{err}}}}}{\varepsilon_{\text{err}}}\right).$$

Our Contributions. We design and study new algorithms for k-fold numerical composition of PLDs. Specifically, for reasonable choice of mechanisms, we

- ▷ obtain (Section 3) a two-stage algorithm for self-composing PLDs, with running time

$$\widetilde{O}\left(\frac{k^{0.25}\sqrt{\log\frac{1}{\delta_{\text{err}}}}}{\varepsilon_{\text{err}}}\right).$$

- ▷ provide (Section 4) an experimental evaluation of the two-stage algorithm, comparing its runtime to that of the algorithm of Gopi et al. (2021). We find that the speedup obtained by our algorithm improves with k.
- ▷ extend (Section 5) the two-stage algorithm to a recursive multi-stage algorithm with a running time of

$$\widetilde{O}\left(\frac{\text{polylog}(k)\sqrt{\log\frac{1}{\delta_{\text{err}}}}}{\varepsilon_{\text{err}}}\right).$$

[1]For any positive T, we denote by $\widetilde{O}(T)$ any quantity of the form $O(T \cdot \text{polylog}(T))$.

Both the two-stage and multi-stage algorithms extend to the case of composing k different mechanisms. In each case, the running time increases by a multiplicative $O(k)$ factor. Note that this factor is inevitable since the input "size" is k—indeed, the algorithm needs to read the k input PLDs. We defer the details of this extension to Appendix B.

Algorithm Overview. The main technique underlying our algorithms is the evolution of the discretization and truncation intervals of the approximations of the PRV with the number of compositions. To describe our approach, we first present a high-level description of the algorithm of Gopi et al. (2021). Their algorithm discretizes an $O(1)$-length interval into bucket intervals each with mesh size $\approx \frac{1}{k^{0.5}}$, thus leading to a total of $O(k^{0.5})$ buckets and a running time of $\widetilde{O}(k^{0.5})$ for the FFT convolution algorithm. Both these aspects of their algorithm are in some sense necessary: a truncation interval of length $\ll O(1)$ would lose significant information about the k-fold composition PRV, whereas a discretization interval of length $\gg \frac{1}{k^{0.5}}$ would lose significant information about the original PRV; so relaxing either would lead to large approximation error.

The key insight in our work is that it is possible to avoid having both these aspects *simultaneously*. In particular, in our two-stage algorithm, the first stage performs a $k^{0.5}$-fold composition using an interval of length $\approx \frac{1}{k^{0.25}}$ discretized into bucket intervals with mesh size $\approx \frac{1}{k^{0.5}}$, followed by another $k^{0.5}$-fold composition using an interval of length $O(1)$ discretized into bucket intervals with mesh size $\approx \frac{1}{k^{0.25}}$. Thus, in each stage the number of discretization buckets is $\widetilde{O}(k^{0.25})$ leading to a final running time of $\widetilde{O}(k^{0.25})$ for performing two FFT convolutions.

The recursive multi-stage algorithm extends this idea to $O(\log k)$ stages, each performed with an increasing discretization interval and truncation interval, ensuring that the number of discretization buckets at each step is $O(\text{polylog}(k))$.

Experimental Evaluation. We implement our two-stage algorithm and compare it to baselines from the literature. We consider the sub-sampled Gaussian mechanism, which is a fundamental primitive in private machine learning and constitutes a core primitive of training algorithms that use differentially private stochastic gradient descent (DP-SGD) (see, e.g., (Abadi et al., 2016)). For 2^{16} compositions and a standard deviation of ≈ 226.86 and with subsampling probability of 0.2, we obtain a speedup of $2.66\times$ compared to the state-of-the-art. We also consider compositions of the widely-used Laplace mechanism. For 2^{16} compositions, and a scale parameter of ≈ 1133.84 for the Laplace distribution, we achieve a speedup of $2.3\times$. The parameters were chosen such that the composed mechanism satisfies $(\varepsilon = 1.0, \delta = 10^{-6})$-DP. See Section 4 for more details.

Related Work. In addition to Moments Accountant (Abadi et al., 2016) and Rényi DP (Mironov, 2017) (which were originally used to bound the privacy loss in DP deep learning), several other tools can also be used to upper-bound the privacy parameters of composed mechanisms, including concentrated DP (Dwork & Rothblum, 2016; Bun & Steinke, 2016), its truncated variant (Bun et al., 2018), and Gaussian DP (Dong et al., 2019; Bu et al., 2020). However, none of these methods is tight; furthermore, none of them allows a high-accuracy estimation of the privacy parameters. In fact, some of them require that the moments of the PRV are bounded; the PLD composition approach does not have such restrictions and hence is applicable to mechanisms such as DP-SGD-JL (Bu et al., 2021).

In a recent work, Doroshenko et al. (2022) proposed a different discretization procedure based on whether we want the discretized PLD to be "optimistic" or "pessimistic". They do not analyze the error bounds explicitly but it can be seen that their running time is $\tilde{O}(k)$, which is slower than both our algorithm and that of Gopi et al. (2021).

Another recent work (Zhu et al., 2022) developed a rigorous notion of "worst-case" PLD for mechanisms, under the name *dominating PLDs*. Our algorithms can be used for compositions of dominating PLDs; indeed, our experimental results for Laplace and (subsampled) Gaussian mechanisms are already doing this implicitly. Furthermore, Zhu et al. (2022) propose a different method for computing tight DP composition bounds. However, their algorithm requires an analytical expression for the characteristic function of the PLDs. This may not exist, e.g., we are unaware of such an analytical expression for subsampled Gaussian mechanisms.

2. Preliminaries

Let $\mathbb{Z}_{>0}$ denote the set of all positive integers, $\mathbb{R}_{\geq 0}$ the set of all non-negative real numbers, and let $\overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty, +\infty\}$. For any two random variables X and Y, we denote by $d_{\mathrm{TV}}(X, Y)$ their *total variation distance*.

2.1. Privacy Loss Random Variables

We will use the following privacy definitions and tools that appeared in previous works on PLDs (Sommer et al., 2019; Koskela et al., 2020; Gopi et al., 2021).

For any mechanism $\mathcal{M}$, and any $\varepsilon \in \mathbb{R}_{\geq 0}$, we denote by $\delta_{\mathcal{M}}(\varepsilon)$ the smallest value δ such that $\mathcal{M}$ is (ε, δ)-DP. The resulting function $\delta_{\mathcal{M}}(\cdot)$ is said to be the *privacy curve* of the mechanism $\mathcal{M}$. A closely related notion is the *privacy curve* $\delta(X||Y) : \mathbb{R}_{\geq 0} \to [0, 1]$ between two random variables X, Y, and which is defined, for any $\varepsilon \in \mathbb{R}_{\geq 0}$ as

$$\delta(X||Y)(\varepsilon) = \sup_{S \subset \Omega} \Pr[Y \in S] - e^{\varepsilon} \Pr[X \in S],$$

where Ω is the support of X and Y. The *privacy loss random variables* associated with a privacy curve $\delta_{\mathcal{M}}$ are random variables (X, Y) such that $\delta_{\mathcal{M}}$ is the same curve as $\delta(X||Y)$, and that satisfy certain additional properties (which make them unique). While PRVs have been defined earlier in Dwork & Rothblum (2016); Bun & Steinke (2016), we use the definition of Gopi et al. (2021):

Definition 2.1 (PRV)**.** Given a privacy curve $\delta_{\mathcal{M}} : \mathbb{R}_{\geq 0} \to [0, 1]$, we say that random variables (X, Y) are *privacy loss random variables (PRVs)* for $\delta_{\mathcal{M}}$, if (i) X and Y are supported on $\overline{\mathbb{R}}$, (ii) $\delta(X||Y) = \delta_{\mathcal{M}}$, (iii) $Y(t) = e^t X(t)$ for every $t \in \mathbb{R}$, and (iv) $Y(-\infty) = X(\infty) = 0$, where $X(t)$ and $Y(t)$ denote the PDFs of X and Y, respectively.

Theorem 2.2 (Gopi et al. (2021))**.** *Let δ be a privacy curve that is identical to $\delta(P||Q)$ for some random variables P and Q. Then, the PRVs (X, Y) for δ are given by*

$$X = \log\left(\frac{Q(w)}{P(w)}\right) \text{ where } \omega \sim P,$$

and

$$Y = \log\left(\frac{Q(w)}{P(w)}\right) \text{ where } \omega \sim Q.$$

Moreover, we define $\delta_Y(\varepsilon) := \mathbb{E}_Y[(1 - e^{\varepsilon - Y})_+]$ for every $\varepsilon \in \mathbb{R}$ and define $\varepsilon_Y(\delta)$ as the smallest ε such that $\delta_Y(\varepsilon) \leq \delta$.

Note that $\delta_Y(\varepsilon)$ is well defined even for Y that is *not* a privacy loss random variable.

If δ_1 is a privacy curve identical to $\delta(X_1||Y_1)$ and δ_2 is a privacy curve identical to $\delta(X_2||Y_2)$, then the composition $\delta_1 \otimes \delta_2$ is defined as the privacy curve $\delta((X_1, X_2)||(Y_1, Y_2))$, where X_1 is independent of X_2, and Y_1 is independent of Y_2. A crucial property is that composition of privacy curves corresponds to addition of PRVs:

Theorem 2.3 (Dwork & Rothblum (2016))**.** *Let δ_1 and δ_2 be privacy curves with PRVs (X_1, Y_1) and (X_2, Y_2) respectively. Then, the PRVs for the privacy curve $\delta_1 \otimes \delta_2$ are given by $(X_1 + X_2, Y_1 + Y_2)$.*

We interchangeably use the same letter to denote both a random variable and its corresponding probability distribution. For any two distributions Y_1, Y_2, we use $Y_1 \oplus Y_2$ to denote its *convolution* (same as the random variable $Y_1 + Y_2$). We use $Y^{\oplus k}$ to denote the k-fold convolution of Y with itself.

Finally, we use the following tail bounds for PRVs.

Lemma 2.4 (Gopi et al. (2021))**.** *For all PRV Y, $\varepsilon \geq 0$ and $\alpha > 0$, it holds that*

$$\begin{aligned} \Pr[|Y| \geq \varepsilon + \alpha] &\leq \delta_Y(\varepsilon) \cdot \tfrac{(1 + e^{-\varepsilon - \alpha})}{1 - e^{-\alpha}} \\ &\leq \tfrac{4}{\alpha} \delta_Y(\varepsilon) \quad \text{if } \alpha < 1. \end{aligned}$$

2.2. Coupling Approximation

To describe and analyze our algorithm, we use the coupling approximation tool used by Gopi et al. (2021). They showed

that, in order to provide an approximation guarantee on the privacy loss curve, it suffices to approximate a PRV according to the following coupling notion of approximation:

Definition 2.5 (Coupling Approximation)**.** For two random variables Y_1, Y_2, we write $|Y_1 - Y_2| \leq_\eta h$ if there exists a *coupling* between them such that $\Pr[|Y_1 - Y_2| > h] \leq \eta$.

We use the following properties of coupling approximation shown by Gopi et al. (2021). We provide the proofs in Appendix A for completeness.

Lemma 2.6 (Properties of Coupling Approximation)**.** *Let X, Y, Z be random variables.*

1. *If $|X - Y| \leq_{\delta_{\mathrm{err}}} \varepsilon_{\mathrm{err}}$, then for all $\varepsilon \in \mathbb{R}_{\geq 0}$,*
$$\delta_Y(\varepsilon + \varepsilon_{\mathrm{err}}) - \delta_{\mathrm{err}} \;\leq\; \delta_X(\varepsilon) \;\leq\; \delta_Y(\varepsilon - \varepsilon_{\mathrm{err}}) + \delta_{\mathrm{err}}.$$
2. *If $|X - Y| \leq_{\eta_1} h_1$ and $|Y - Z| \leq_{\eta_2} h_2$, then $|X - Z| \leq_{\eta_1+\eta_2} h_1 + h_2$ ("triangle inequality").*
3. *If $d_{\mathrm{TV}}(X, Y) \leq \eta$, then $|X - Y| \leq_\eta 0$; furthermore, for all $k \in \mathbb{Z}_{>0}$, it holds that $|X^{\oplus k} - Y^{\oplus k}| \leq_{k\eta} 0$.*
4. *If $|X - Y - \mu| \leq_0 h$ (for any μ) and $\mathbb{E}[X] = \mathbb{E}[Y]$, then for all $\eta > 0$ and $k \in \mathbb{Z}_{>0}$,*
$$\left|X^{\oplus k} - Y^{\oplus k}\right| \;\leq_\eta\; h\sqrt{2k \log \tfrac{2}{\eta}}.$$

2.3. Discretization Procedure

We adapt the discretization procedure from (Gopi et al., 2021). The only difference is that we assume the support of the input distribution is already in the specified range as opposed to being truncated as part of the algorithm. A complete description of the procedure is given in Algorithm 1.

Algorithm 1 DiscretizeRV (adapted from Gopi et al., 2021)

Input: $\mathsf{CDF}_Y(\cdot)$ of a RV Y, mesh size h, truncation parameter $L \in h \cdot (\frac{1}{2} + \mathbb{Z}_{>0})$.
Constraint: Support of Y is contained in $(-L, L]$.
Output: PDF of an approximation $\widetilde{Y}$ supported on $\mu + (h\mathbb{Z} \cap [-L, L])$ for some $\mu \in [-\frac{h}{2}, \frac{h}{2}]$.

$n \leftarrow \frac{L - \frac{h}{2}}{h}$
for $i = -n$ **to** n **do**
  $q_i \leftarrow \mathsf{CDF}_Y(ih + h/2) - \mathsf{CDF}_Y(ih - h/2)$
end for
$q \leftarrow q/(\sum_{i=-n}^{n} q_i)$ ▷ Normalize q
$\mu \leftarrow \mathbb{E}[Y] - \sum_{i=-n}^{n} ih \cdot q_i$
$\widetilde{Y} \leftarrow \{ih + \mu \quad \text{w.p. } q_i \quad \text{for } -n \leq i \leq n$
return $\widetilde{Y}$

The procedure can be shown to satisfy the following key property.

Proposition 2.7. *For any random variable Y supported in $(-L, L]$, the output $\widetilde{Y}$ of* DiscretizeRV *with mesh size h and truncation parameter L satisfies $\mathbb{E}[Y] = \mathbb{E}[\widetilde{Y}]$ and $|Y - \widetilde{Y} - \mu| \leq_0 \frac{h}{2}$, for some μ with $|\mu| \leq \frac{h}{2}$.*

Algorithm 2 TwoStageSelfComposePRV

Input: PRV Y, number of compositions $k = k_1 \cdot k_2 + r$ (for $r < k_1$), mesh sizes $h_1 \leq h_2$, truncation parameters $L_1 \leq L_2$, where each $L_i \in h_i \cdot (\frac{1}{2} + \mathbb{Z}_{>0})$.
Output: PDF of an approximation Y_2 for $Y^{\oplus k}$. Y_2 will be supported on $\mu + (h_2\mathbb{Z} \cap [-L_2, L_2])$ for some $\mu \in [0, \frac{h_2}{2}]$.

$Y_0 \leftarrow Y|_{|Y| \leq L_1}$ ▷ Y conditioned on $|Y| \leq L_1$
$\widetilde{Y}_0 \leftarrow \mathsf{DiscretizeRV}(Y_0, h_1, L_1)$
$Y_1 \leftarrow \widetilde{Y}_0^{\oplus_{L_1} k_1}$ ▷ k_1-fold FFT convolution
$\widetilde{Y}_1 \leftarrow \mathsf{DiscretizeRV}(Y_1, h_2, L_2)$
$Y_2 \leftarrow \widetilde{Y}_1^{\oplus_{L_2} k_2}$ ▷ k_2-fold FFT convolution
$Y_r \leftarrow \widetilde{Y}_0^{\oplus_{L_1} r}$ ▷ r-fold FFT convolution
$\widetilde{Y}_r \leftarrow \mathsf{DiscretizeRV}(Y_r, h_2, L_2)$
return $Y_2 \oplus_{L_2} \widetilde{Y}_r$ ▷ FFT convolution

3. Two-Stage Composition Algorithm

Our two-stage algorithm for the case of k-fold self-composition is given in Algorithm 2. We assume $k = k_1 \cdot k_2 + r$ where $k_1, k_2 \in \mathbb{Z}_{>0}$, $r < k_1$, and $k_1 \approx k_2 \approx \sqrt{k}$, which for instance can be achieved by taking $k_1 = \lfloor \sqrt{k} \rfloor$, $k_2 = \lfloor k/k_1 \rfloor$, and $r = k - k_1 \cdot k_2$.

The algorithm implements the *circular convolution* $\oplus_L$ using Fast Fourier Transform (FFT). For any L and $x \in \mathbb{R}$, we define $x \pmod{2L} = x - 2Ln$ where $n \in \mathbb{Z}$ such that $x - 2Ln \in (-L, L]$. Given $x, y \in \mathbb{R}$ the *circular addition* is defined as

$$x \oplus_L y \;:=\; x + y \pmod{2L}.$$

Similarly, for random variables X, Y, we define $X \oplus_L Y$ to be their convolution modulo $2L$ and $Y^{\oplus_L k}$ to be the k-fold convolution of Y modulo $2L$.

Observe that DiscretizeRV with mesh size h and truncation parameter L runs in time $O(L/h)$, assuming an $O(1)$-time access to $\mathsf{CDF}_Y(\cdot)$. The FFT convolution step runs in time $O\left(\frac{L_i}{h_i} \log \frac{L_i}{h_i}\right)$; thus the overall running time of TwoStageSelfComposePRV is

$$O\left(\tfrac{L_1}{h_1} \log\left(\tfrac{L_1}{h_1}\right) + \tfrac{L_2}{h_2} \log\left(\tfrac{L_2}{h_2}\right)\right).$$

The approximation guarantees provided by our two-stage algorithm are captured in the following theorem. For convenience, we assume that k is a perfect square (we set $k_1 = k_2 = \sqrt{k}$). The complete proof is in Section 3.1.

Theorem 3.1. *For any PRV Y, the approximation Y_2 returned by* TwoStageSelfComposePRV *satisfies*

$$|Y^{\oplus k} - Y_2| \leq_{\delta_{\text{err}}} \varepsilon_{\text{err}},$$

when invoked with $k_1 = k_2 = k^{0.5}$ (assumed to be an integer) and parameters given below (using $\eta := \frac{\delta_{\text{err}}}{(8\sqrt{k}+16)}$)

$$h_1 := \frac{\varepsilon_{\text{err}}}{k^{0.5}\sqrt{2\log\frac{1}{\eta}}}, \quad h_2 := \frac{\varepsilon_{\text{err}}}{k^{0.25}\sqrt{2\log\frac{1}{\eta}}}$$

$$\begin{aligned} L_1 &\geq \max\left\{ \begin{matrix} \varepsilon_Y\left(\frac{\varepsilon_{\text{err}}\delta_{\text{err}}}{16k^{1.25}}\right), \\ \varepsilon_{Y^{\oplus\sqrt{k}}}\left(\frac{\varepsilon_{\text{err}}\delta_{\text{err}}}{64k^{0.75}}\right) \end{matrix} \right\} + \frac{\varepsilon_{\text{err}}}{k^{0.25}} \\ L_2 &\geq \max\left\{\varepsilon_{Y^{\oplus k}}\left(\frac{\varepsilon_{\text{err}}\delta_{\text{err}}}{16}\right) + 2\varepsilon_{\text{err}}, L_1\right\}. \end{aligned}$$

In terms of a concrete running time bound, Theorem 3.1 implies:

Corollary 3.2. *For any DP algorithm $\mathcal{M}$, the privacy curve $\delta_{\mathcal{M}^{\circ k}}(\varepsilon)$ of the k-fold (adaptive) composition of $\mathcal{M}$ can be approximated in time $\widetilde{O}\left(\frac{\varepsilon_{\text{up}}\cdot k^{0.25}\cdot\sqrt{\log(k/\delta_{\text{err}})}}{\varepsilon_{\text{err}}}\right)$, where ε_{err} is the additive error in ε, δ_{err} is the additive error in δ, and ε_{up} is an upper bound on*

$$\max\left\{ \begin{matrix} \varepsilon_{Y^{\oplus k}}\left(\frac{\varepsilon_{\text{err}}\delta_{\text{err}}}{16}\right), \\ \sqrt[4]{k}\cdot\varepsilon_{Y^{\oplus\sqrt{k}}}\left(\frac{\varepsilon_{\text{err}}\delta_{\text{err}}}{64k^{0.75}}\right), \\ \sqrt[4]{k}\cdot\varepsilon_{Y}\left(\frac{\varepsilon_{\text{err}}\delta_{\text{err}}}{16k^{1.25}}\right) \end{matrix} \right\} + \varepsilon_{\text{err}}.$$

In many practical regimes of interest, $\varepsilon_{\text{up}}/\varepsilon_{\text{err}}$ is a constant. For ease of exposition in the following, we assume that ε_{err} is a small constant, e.g. 0.1 and suppress the dependence on ε_{err}. Suppose the original mechanism $\mathcal{M}$ underlying Y satisfies $(\varepsilon = \frac{1}{\sqrt{k\cdot\log(1/\delta_{\text{err}})}}, \delta = o_k\left(\frac{1}{k^{1.25}}\right))$-DP. Then by advanced composition (Dwork et al., 2010), we have that $\mathcal{M}^{\circ t}$ satisfies $(\varepsilon\sqrt{2t\log\frac{1}{\delta'}} + 2t\varepsilon(e^\varepsilon - 1), t\delta + \delta')$-DP. If $t\delta + \delta' \lesssim \frac{t\delta_{\text{err}}}{k^{1.25}}$, then we have that $\varepsilon_{Y^{\oplus t}}\left(\frac{t\delta_{\text{err}}}{k^{1.25}}\right) \lesssim \sqrt{\frac{t}{k}\ln\frac{k}{t\delta_{\text{err}}}}$. Instantiating this with $t = 1, \sqrt{k}$, and k gives us that ε_{up} is at most a constant.

Note that, to set the value of L_1 and L_2, we do not need the exact value of $\varepsilon_{Y^{\oplus k}}$ (or $\varepsilon_{Y^{\oplus\sqrt{k}}}$ or ε_Y). We only need an upper bound on $\varepsilon_{Y^{\oplus k}}$, which can often be obtained by using the RDP accountant or some other method.

For the case when k is not a perfect square, using a similar analysis, it is easy to see that the approximation error would be no worse than the error in $k_2(k_1+1)$ self-compositions. The running time increases by a constant because of the additional step of r-fold convolution to get Y_r and the final convolution step to get $Y_2 \oplus_{L_2} \widetilde{Y}_r$; however this does not affect the asymptotic time complexity of the algorithm. Moreover, as seen in Section 4, even with this additional cost, TwoStageSelfComposePRV is still faster than the algorithm in Gopi et al. (2021).

3.1. Analysis

In this section we establish Theorem 3.1. The proof proceeds are follows. We establish coupling approximations between consecutive random variables in the following sequence:

$$Y^{\oplus k_1k_2},\ Y_0^{\oplus k_1k_2},\ \widetilde{Y}_0^{\oplus k_1k_2},\ Y_1^{\oplus k_2},\ \widetilde{Y}_1^{\oplus k_2}, Y_2,$$

and then apply Lemma 2.6(2).

To establish each coupling approximation, we use Lemmas 2.6(3) and 2.6(4).

Coupling $\left(\boldsymbol{Y^{\oplus k_1k_2}}, \boldsymbol{Y_0^{\oplus k_1k_2}}\right)$. Since $d_{\text{TV}}(Y, Y_0) = \Pr[|Y| > L_1] =: \delta_0$, we have from Lemma 2.6(3) that

$$|Y^{\oplus k_1k_2} - Y_0^{\oplus k_1k_2}| \leq_{k_1k_2\delta_0} 0. \tag{1}$$

Coupling $\left(\boldsymbol{Y_0^{\oplus k_1k_2}}, \boldsymbol{\widetilde{Y}_0^{\oplus k_1k_2}}\right)$ and $\left(\boldsymbol{Y_1^{\oplus k_2}}, \boldsymbol{\widetilde{Y}_1^{\oplus k_2}}\right)$. We have from Proposition 2.7 that $\mathbb{E}[Y_0] = \mathbb{E}[\widetilde{Y}_0]$ and that $|Y_0 - \widetilde{Y}_0 - \mu| \leq_0 \frac{h_1}{2}$. Thus, by applying Lemma 2.6(4), we have that (for any η; to be chosen later)

$$\left|Y_0^{\oplus k_1} - \widetilde{Y}_0^{\oplus k_1}\right| \leq_\eta h_1\sqrt{\tfrac{k_1}{2}\log\tfrac{2}{\eta}}, \tag{2}$$

$$\left|Y_0^{\oplus k_1k_2} - \widetilde{Y}_0^{\oplus k_1k_2}\right| \leq_\eta h_1\sqrt{\tfrac{k_1k_2}{2}\log\tfrac{2}{\eta}}. \tag{3}$$

Similarly, we have that

$$\left|Y_1^{\oplus k_2} - \widetilde{Y}_1^{\oplus k_2}\right| \leq_\eta h_2\sqrt{\tfrac{k_2}{2}\log\tfrac{2}{\eta}}. \tag{4}$$

Coupling $\left(\boldsymbol{\widetilde{Y}_0^{\oplus k_1k_2}}, \boldsymbol{Y_1^{\oplus k_2}}\right)$ and $\left(\boldsymbol{\widetilde{Y}_1^{\oplus k_2}}, \boldsymbol{Y_2}\right)$. Since $d_{\text{TV}}(\widetilde{Y}_0^{\oplus k_1}, \widetilde{Y}_0^{\oplus_{L_1}k_1}) \leq \Pr\left[\left|\widetilde{Y}_0^{\oplus k_1}\right| > L_1\right] =: \delta_1$, and $Y_1 = Y_0^{\oplus_{L_1}k_1}$, it holds via Lemma 2.6(3) that

$$\left|\widetilde{Y}_0^{\oplus k_1k_2} - Y_1^{\oplus k_2}\right| \leq_{k_2\delta_1} 0. \tag{5}$$

Similarly, for $\delta_2 := \Pr\left[\left|\widetilde{Y}_1^{\oplus k_2}\right| > L_2\right]$, we have that

$$\left|\widetilde{Y}_1^{\oplus k_2} - Y_2\right| \leq_{\delta_2} 0. \tag{6}$$

Towards combining the bounds. Combining Equations (1) and (3) to (6) using Lemma 2.6(2), we have that

$$\begin{aligned} \left|Y^{\oplus k_1k_2} - Y_2\right| &\leq_{\delta_{\text{err}}} \varepsilon_{\text{err}}, \\ \text{where}\quad \delta_{\text{err}} &= 2\eta + k_1k_2\delta_0 + k_2\delta_1 + \delta_2, \qquad (7) \\ \text{and}\quad \varepsilon_{\text{err}} &= \left(h_1\sqrt{k_1k_2} + h_2\sqrt{k_2}\right)\sqrt{\tfrac{1}{2}\log\tfrac{2}{\eta}}. \qquad (8) \end{aligned}$$

We set $h_1 := \frac{\varepsilon_{\text{err}}}{\sqrt{2k_1k_2\log\frac{2}{\eta}}}$ and $h_2 := \frac{\varepsilon_{\text{err}}}{\sqrt{2k_2\log\frac{2}{\eta}}}$. The key step remaining is to bound δ_0, δ_1, and δ_2 in terms of h_i's, L_i's, and η_i's.

To do so, we use Lemma 2.4 for α_i's to be chosen later.

Bounding δ_0. We have $\delta_0 := \Pr[|Y| > L_1]$ and hence

$$\delta_0 \;\le\; \tfrac{4}{\alpha_0}\delta_Y(L_1 - \alpha_0).$$

Bounding δ_1. For $\widetilde{h}_1 := h_1\sqrt{\frac{k_1}{2}\log\frac{2}{\eta}} = \frac{\varepsilon_{\text{err}}}{2\sqrt{k_2}}$, we have

$$\begin{aligned}
\delta_1 \;&:=\; \Pr\left[\left|\widetilde{Y}_0^{\oplus k_1}\right| > L_1\right] \\
&\le\; \Pr\left[\left|\widetilde{Y}_0^{\oplus k_1} - Y_0^{\oplus k_1}\right| > \widetilde{h}_1\right] \\
&\quad+\; \Pr\left[\left|Y_0^{\oplus k_1}\right| > L_1 - \widetilde{h}_1\right] \\
&\le\; \eta + \Pr\left[\left|Y^{\oplus k_1}\right| > L_1 - \widetilde{h}_1\right] \\
\implies\quad \delta_1 \;&\le\; \eta + \tfrac{4}{\alpha_1}\delta_{Y^{\oplus k_1}}\left(L_1 - \tfrac{\varepsilon_{\text{err}}}{2\sqrt{k_2}} - \alpha_1\right),
\end{aligned}$$

where the second inequality uses Equation (2) and the third inequality uses the fact that the tails of $Y^{\oplus k_1}$ are only heavier than the tails of $Y_0^{\oplus k_1}$ since Y_0 is a truncation of Y.

Bounding δ_2. First, we combine Equations (3) to (5) to get (recall ε_{err} in Equation (8))

$$\left|Y_0^{\oplus k_1 k_2} - \widetilde{Y}_1^{\oplus k_2}\right| \;\le_{2\eta + k_2\delta_1,\;\varepsilon_{\text{err}}}.$$

Using this, we get

$$\begin{aligned}
\delta_2 \;&:=\; \Pr\left[\left|\widetilde{Y}_1^{\oplus k_2}\right| > L_2\right] \\
&\le\; \Pr\left[\left|\widetilde{Y}_1^{\oplus k_2} - Y_0^{\oplus k_1 k_2}\right| > \varepsilon_{\text{err}}\right] \\
&\quad+\; \Pr\left[\left|Y_0^{\oplus k_1 k_2}\right| > L_2 - \varepsilon_{\text{err}}\right] \\
&\le\; 2\eta + k_2\delta_1 + \Pr\left[\left|Y^{\oplus k_1 k_2}\right| > L_2 - \varepsilon_{\text{err}}\right] \\
&\le\; 2\eta + k_2\delta_1 + \tfrac{4}{\alpha_2}\delta_{Y^{\oplus k_1 k_2}}\left(L_2 - \varepsilon_{\text{err}} - \alpha_2\right) \\
\implies\quad \delta_2 \;&\le\; (k_2+2)\eta + k_2\cdot\tfrac{4}{\alpha_1}\delta_{Y^{\oplus k_1}}(L_1 - \tfrac{\varepsilon_{\text{err}}}{2\sqrt{k_2}} - \alpha_1) \\
&\quad+\; \tfrac{4}{\alpha_2}\delta_{Y^{\oplus k_1 k_2}}\left(L_2 - \varepsilon_{\text{err}} - \alpha_2\right),
\end{aligned}$$

where we use in the third step that tails of $Y^{\oplus k_1 k_2}$ are only heavier than tails of $Y_0^{\oplus k_1 k_2}$ since Y_0 is a truncation of Y.

Putting it all together. Plugging in these bounds for δ_0, δ_1, and δ_2 in Equation (7), we get that

$$\begin{aligned}
&\left|Y^{\oplus k_1 k_2} - Y_2\right| \;\le_{\delta_{\text{err}},\;\varepsilon_{\text{err}}}, \\
\text{where}\quad \delta_{\text{err}} \;&\le\; (2k_2+4)\eta + k_1 k_2\cdot\tfrac{4}{\alpha_0}\delta_Y(L_1 - \alpha_0) \\
&\quad+\; 2k_2\cdot\tfrac{4}{\alpha_1}\delta_{Y^{\oplus k_1}}(L_1 - \tfrac{\varepsilon_{\text{err}}}{2\sqrt{k_2}} - \alpha_1) \\
&\quad+\; \tfrac{4}{\alpha_2}\delta_{Y^{\oplus k_1 k_2}}\left(L_2 - \varepsilon_{\text{err}} - \alpha_2\right), \qquad (9) \\
\text{and}\quad \varepsilon_{\text{err}} \;&=\; \left(h_1\sqrt{k_1 k_2} + h_2\sqrt{k_2}\right)\sqrt{2\log\tfrac{2}{\eta}}.
\end{aligned}$$

Thus, we can set parameters L_1, L_2, and η as follows such that each of the four terms in Equation (9) is at most $\delta_{\text{err}}/4$ to satisfy the above:

$$\begin{aligned}
\eta \;&=\; \tfrac{\delta_{\text{err}}}{8k_2+16}, \\
L_1 \;&\ge\; \max\left\{\begin{matrix} \varepsilon_Y\left(\frac{\alpha_0\delta_{\text{err}}}{16k_1k_2}\right) + \alpha_0, \\ \varepsilon_{Y^{\oplus k_1}}\left(\frac{\alpha_1\delta_{\text{err}}}{32k_2}\right) + \frac{\varepsilon_{\text{err}}}{2\sqrt{k_2}} + \alpha_1 \end{matrix}\right\}, \\
L_2 \;&\ge\; \max\left\{\varepsilon_{Y^{\oplus k_1 k_2}}\left(\tfrac{\alpha_2\delta_{\text{err}}}{16}\right) + \varepsilon_{\text{err}} + \alpha_2, L_1\right\}.
\end{aligned}$$

Setting $k_1 = k_2 = \sqrt{k}$ (assumed to be integers) and $\alpha_0 = \frac{\varepsilon_{\text{err}}}{\sqrt{k_2}}$, $\alpha_1 = \frac{\varepsilon_{\text{err}}}{2\sqrt{k_2}}$ and $\alpha_2 = \varepsilon_{\text{err}}$, completes the proof of Theorem 3.1.

Runtime analysis. As argued earlier, the total running time of TwoStageSelfComposePRV is given as $O\left(\frac{L_1}{h_1}\log\frac{L_1}{h_1} + \frac{L_2}{h_2}\log\frac{L_2}{h_2}\right)$. Substituting in the bounds for L_1, L_2, h_1, and h_2, we get a final running time of

$$\widetilde{O}\left(\frac{\varepsilon_{\text{up}}\cdot k^{0.25}\cdot\sqrt{\log(k/\delta_{\text{err}})}}{\varepsilon_{\text{err}}}\right).$$

where ε_{up} is as in Corollary 3.2.

3.2. Heterogeneous Compositions

Algorithm 2 can be easily generalized to handle *heterogeneous composition* of k different mechanisms, with a running time blow up of k over the homogeneous case. We defer the details to Appendix B.

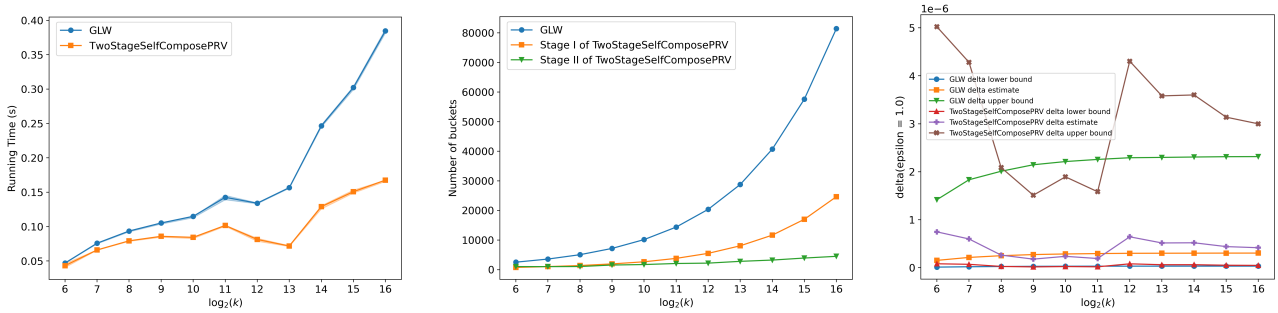
4. Experimental Evaluation of TwoStageSelfComposePRV

We empirically evaluate TwoStageSelfComposePRV on the tasks of self-composing two kinds of mechanism acting on dataset of real values $x_1, \ldots, x_n$ as

- ▷ Laplace Mechanism : returns $\sum_i x_i$ plus a noise drawn from $L(0, b)$ given by the PDF $P(x) = e^{-|x|/b}/2b$.
- ▷ Poisson Subsampled Gaussian Mechanism with probability γ: Subsamples a random subset S of indices by including each index independently with probability γ. Returns $\sum_{i\in S} x_i$ plus a noise drawn from the Gaussian distribution $\mathcal{N}(0, \sigma^2)$.

Both these mechanisms are highly used in practice. For each mechanism, we compare against the implementation by Gopi et al. (2021)[2] (referred to as GLW) on three fronts: (i) the running time of the algorithm, (ii) the number of discretized buckets used, and (iii) the final estimates on $\delta_{Y^{\oplus k}}(\varepsilon)$ which includes comparing lower bound $\delta_{Y_2}(\varepsilon + \varepsilon_{\text{err}}) - \delta_{\text{err}}$, estimates $\delta_{Y_2}(\varepsilon)$ and upper bounds $\delta_{Y_2}(\varepsilon -$

[2] github.com/microsoft/prv_accountant

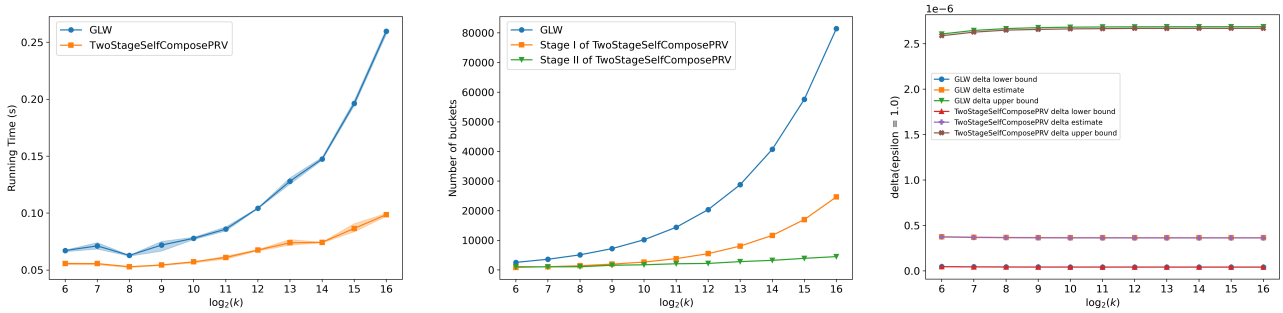


(a) Running times (average over 20 runs); shaded region indicates 20th–80th percentiles

(b) Number of discretization buckets

(c) Delta estimates

Figure 1. Compositions of the Laplace mechanism.



(a) Running times (average over 20 runs); shaded region indicates 20th–80th percentiles

(b) Number of discretization buckets

(c) Delta estimates

Figure 2. Compositions of Poisson subsampled (with probability $\gamma = 0.2$) Gaussian mechanism.

$\varepsilon_{\text{err}}) + \delta_{\text{err}}$. We use $\varepsilon_{\text{err}} = 0.1$ and $\delta_{\text{err}} = 10^{-10}$ in all the experiments.

We run each algorithm for a varying number k of self-compositions of a basic mechanism. The noise parameter of basic mechanism is chosen such that the final $\delta(\varepsilon)$ value after k-fold composition is equal to 10^{-6} for each value of k.[3] The exact choice of noise parameters used are shown in Figure 3.

The comparison for the Laplace mechanism is shown in Figure 1 and for the subsampled Gaussian mechanism is shown in Figure 2. In terms of accuracy we find that for the same choice of ε_{err} and δ_{err}, the estimates returned by TwoStageSelfComposePRV are nearly identical to the estimates returned by GLW for the subsampled Gaussian mechanism. On the other hand, the estimates for Laplace mechanism returned by both algorithms are similar and consistent with each other, but strictly speaking, incomparable with each other.

[3] these values were computed using the Google DP accountant `github.com/google/differential-privacy/tree/main/python`.

5. Multi-Stage Recursive Composition

We extend the approach in TwoStageSelfComposePRV to give a multi-stage algorithm (Algorithm 3), presented only when k is a power of 2 for ease of notation. Similar to the running time analysis of TwoStageSelfComposePRV, the running time of RecursiveSelfComposePRV is given as

$$O\left(\sum_{i=1}^{t} \frac{L_i}{h_i} \log\left(\frac{L_i}{h_i}\right)\right),$$

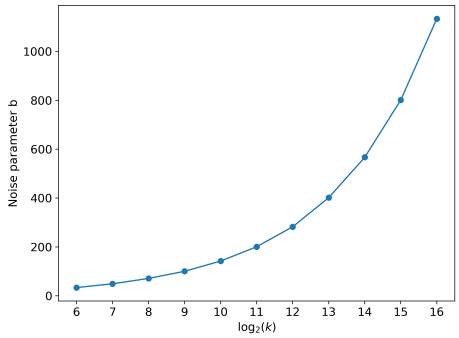
assuming an $O(1)$-time access to $\mathsf{CDF}_Y(\cdot)$.

Theorem 5.1. *For all PRV Y and $k = 2^t$, the approximation Y_t returned by* RecursiveSelfComposePRV *satisfies*

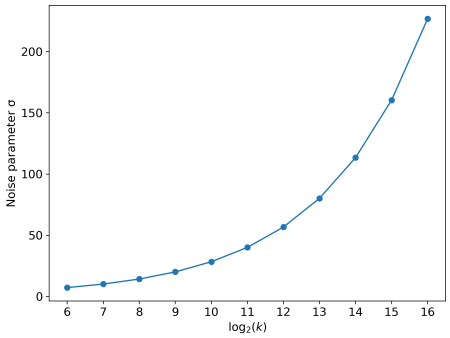
$$|Y^{\oplus k} - Y_t| \leq_{\delta_{\text{err}}} \varepsilon_{\text{err}},$$

using a choice of parameters satisfying for all $i \leq t$ that

$$\begin{aligned} h_i &= \Omega\left(\frac{\varepsilon_{\text{err}}}{t^{1.5}\sqrt{2^{t-i}\log\frac{1}{\delta_{\text{err}}}}}\right), \\ L_i &\geq \varepsilon_{Y^{\oplus 2^i}}\left(\frac{\varepsilon_{\text{err}}\delta_{\text{err}}}{2^{O(t)}}\right) + h_i \cdot \left(3 + 2i\sqrt{\tfrac{1}{2}\log\tfrac{2}{\eta}}\right), \\ L_t &\geq \cdots \geq L_1. \end{aligned}$$



(a) For Laplace mechanism (Figure 1).



(b) For Poisson subsampled Gaussian mechanism (Figure 2).

Figure 3. Noise parameters used for experiments.

Algorithm 3 RecursiveSelfComposePRV

Input: PRV Y, number of compositions $k = 2^t$, mesh sizes $h_1 \le \cdots \le h_t$, truncation parameters $L_1 \le \cdots \le L_t$, where each $L_i \in h_i \cdot (\frac{1}{2} + \mathbb{Z}_{>0})$ for all i.
Output: PDF of an approximation Y_t for $Y^{\oplus k}$. Y_t will be supported on $\mu + (h_t\mathbb{Z} \cap [-L_t, L_t])$ for some $\mu \in \left[0, \frac{h_t}{2}\right]$.

$Y_0 \leftarrow Y|_{|Y| \le L_1}$ ▷ Y conditioned on $|Y| \le L_1$
for $i = 0$ **to** $t - 1$ **do**
  $\widetilde{Y}_i \leftarrow \mathsf{DiscretizeRV}(Y_i, h_{i+1}, L_{i+1})$
  $Y_{i+1} \leftarrow \widetilde{Y}_i \oplus_{L_{i+1}} \widetilde{Y}_i$ ▷ FFT convolution
end for
return Y_t

Proof Outline. We establish coupling approximations between consecutive random variables in the sequence:

$$Y^{\oplus 2^t},\ Y_0^{\oplus 2^t},\ \widetilde{Y}_0^{\oplus 2^t},\ Y_1^{\oplus 2^{t-1}},\ \ldots,\ Y_{t-1}^{\oplus 2},\ \widetilde{Y}_{t-1}^{\oplus 2},\ Y_t\,,$$

using a similar approach as in the proof of Theorem 3.1.

Running time analysis. The overall running time is at most $O\left(\sum_i \frac{L_i}{h_i} \log \frac{L_i}{h_i}\right)$, which can be upper bounded by

$$\widetilde{O}\left(\frac{\varepsilon_{\text{up}} \cdot t^{2.5}\sqrt{\log \frac{1}{\delta_{\text{err}}}}}{\varepsilon_{\text{err}}}\right),$$

where $\varepsilon_{\text{up}} := \max_i \left(\sqrt{2^{t-i}} \cdot \varepsilon_{Y^{\oplus 2^i}}\left(\frac{\varepsilon_{\text{err}}\delta_{\text{err}}}{2^{O(t)}}\right)\right)$.

In many practical regimes of interest, $\varepsilon_{\text{up}}/\varepsilon_{\text{err}}$ is at most $\text{polylog}(t) = \text{polyloglog}(k)$. For ease of exposition in the following, we assume that ε_{err} is a small constant, e.g. 0.1 and suppress the dependence on ε_{err}. Suppose the original mechanism $\mathcal{M}$ underlying Y satisfies $(\varepsilon = \frac{1}{\sqrt{k \cdot \log(1/\delta_{\text{err}})}}, \delta = \frac{o_k(1)}{k^{O(1)}})$-DP. Then by advanced composition (Dwork et al., 2010), we have that $\mathcal{M}^{\circ 2^i}$ satisfies $(\varepsilon\sqrt{2^{i+1}\log\frac{1}{\delta'}} + 2^{i+1}\varepsilon(e^\varepsilon - 1), 2^i\delta + \delta')$-DP. If $2^i\delta + \delta' \lesssim \frac{\delta_{\text{err}}}{2^{O(t)}}$, then we have that $\varepsilon_{Y^{\oplus 2^i}}\left(\frac{\delta_{\text{err}}}{2^{O(t)}}\right) \lesssim \sqrt{\frac{1}{2^{t-i}}\ln\frac{2^{O(t)}}{\delta_{\text{err}}}}$. Instantiating this with $i = 1, \ldots, t$ gives us that ε_{up} is at most $\text{polylog}(k)$.

6. Conclusions and Discussion

In this work, we presented an algorithm with a running time and memory usage of $\text{polylog}(k)$ for the task of self-composing a broad class of DP mechanisms k times. We also extended our algorithm to the case of composing k different mechanisms in the same class, resulting in a running time and memory usage $\widetilde{O}(k)$; both of these improve the state-of-the-art by roughly a factor of $\sqrt{k}$. We also demonstrated the practical benefits of our framework compared to the state-of-the-art by evaluating on the sub-sampled Gaussian mechanism and the Laplace mechanism, both of which are widely used in the literature and in practice.

For future work, it would be interesting to tighten the log factors in our bounds. A related future direction is to make the RecursiveSelfComposePRV algorithm more practical, since the current recursive analysis is quite loose. Note that RecursiveSelfComposePRV could also be performed with an arity larger than 2; e.g., with an arity of 100, one would perform 100^3 compositions as a three-stage composition. For any constant arity, our algorithm gives an asymptotic runtime of $O(\text{polylog}k)$ as $k \to \infty$, however, for practical considerations, one may also consider adapting the arity with k to tighten the log factors. We avoided doing so for simplicity, since our focus in obtaining an $O(\text{polylog}(k))$ running time was primarily theoretical.

Acknowledgments

We would like to thank the anonymous reviewers for their thoughtful comments that have improved the quality of the paper.

References

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. Deep learning with differential privacy. In *CCS*, pp. 308–318, 2016.

Abowd, J. M. The US Census Bureau adopts differential privacy. In *KDD*, pp. 2867–2867, 2018.

Apple Differential Privacy Team. Learning with privacy at scale. *Apple Machine Learning Journal*, 2017.

Bu, Z., Dong, J., Long, Q., and Su, W. J. Deep learning with Gaussian differential privacy. *Harvard Data Science Review*, 2020(23), 2020.

Bu, Z., Gopi, S., Kulkarni, J., Lee, Y. T., Shen, J. H., and Tantipongpipat, U. Fast and memory efficient differentially private-SGD via JL projections. In *NeurIPS*, 2021.

Bun, M. and Steinke, T. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *TCC*, pp. 635–658, 2016.

Bun, M., Dwork, C., Rothblum, G. N., and Steinke, T. Composable and versatile privacy via truncated CDP. In *STOC*, pp. 74–86, 2018.

Ding, B., Kulkarni, J., and Yekhanin, S. Collecting telemetry data privately. In *NeurIPS*, pp. 3571–3580, 2017.

Dong, J., Roth, A., and Su, W. J. Gaussian differential privacy. *arXiv:1905.02383*, 2019.

Doroshenko, V., Ghazi, B., Kamath, P., Kumar, R., and Manurangsi, P. Connect the dots: Tighter discrete approximations of privacy loss distributions. In *PETS (to appear)*, 2022.

Dwork, C. and Rothblum, G. N. Concentrated differential privacy. *arXiv:1603.01887*, 2016.

Dwork, C., Kenthapadi, K., McSherry, F., Mironov, I., and Naor, M. Our data, ourselves: Privacy via distributed noise generation. In *EUROCRYPT*, pp. 486–503, 2006a.

Dwork, C., McSherry, F., Nissim, K., and Smith, A. D. Calibrating noise to sensitivity in private data analysis. In *TCC*, pp. 265–284, 2006b.

Dwork, C., Rothblum, G. N., and Vadhan, S. Boosting and differential privacy. In *FOCS*, pp. 51–60, 2010.

Erlingsson, Ú., Pihur, V., and Korolova, A. RAPPOR: Randomized aggregatable privacy-preserving ordinal response. In *CCS*, pp. 1054–1067, 2014.

Google. DP Accounting Library. https://github.com/google/differential-privacy/tree/main/python/dp_accounting, 2020.

Gopi, S., Lee, Y. T., and Wutschitz, L. Numerical composition of differential privacy. In *NeurIPS*, 2021.

Greenberg, A. Apple's "differential privacy" is about collecting your data – but not your data. *Wired, June*, 13, 2016.

Kairouz, P., Oh, S., and Viswanath, P. The composition theorem for differential privacy. In *ICML*, pp. 1376–1385, 2015.

Kenthapadi, K. and Tran, T. T. L. Pripearl: A framework for privacy-preserving analytics and reporting at LinkedIn. In *CIKM*, pp. 2183–2191, 2018.

Koskela, A. and Honkela, A. Computing differential privacy guarantees for heterogeneous compositions using FFT. *arXiv:2102.12412*, 2021.

Koskela, A., Jälkö, J., and Honkela, A. Computing tight differential privacy guarantees using FFT. In *AISTATS*, pp. 2560–2569, 2020.

Koskela, A., Jälkö, J., Prediger, L., and Honkela, A. Tight differential privacy for discrete-valued mechanisms and for the subsampled Gaussian mechanism using FFT. In *AISTATS*, pp. 3358–3366, 2021.

Lukas Prediger, A. K. Code for computing tight guarantees for differential privacy. https://github.com/DPBayes/PLD-Accountant, 2020.

Meiser, S. and Mohammadi, E. Tight on budget? Tight bounds for r-fold approximate differential privacy. In *CCS*, pp. 247–264, 2018.

Microsoft. A fast algorithm to optimally compose privacy guarantees of differentially private (DP) mechanisms to arbitrary accuracy. https://github.com/microsoft/prv_accountant, 2021.

Mironov, I. Rényi differential privacy. In *CSF*, pp. 263–275, 2017.

Murtagh, J. and Vadhan, S. The complexity of computing the optimal composition of differential privacy. In *TCC*, pp. 157–175, 2016.

Rogers, R., Subramaniam, S., Peng, S., Durfee, D., Lee, S., Kancha, S. K., Sahay, S., and Ahammad, P. LinkedIn's audience engagements API: A privacy preserving data analytics system at scale. *Journal of Privacy and Confidentiality*, 11(3), 2021.

Shankland, S. How Google tricks itself to protect Chrome user privacy. *CNET, October*, 2014.

Sommer, D. M., Meiser, S., and Mohammadi, E. Privacy loss classes: The central limit theorem in differential privacy. *PoPETS*, 2019(2):245–269, 2019.

Zhu, Y., Dong, J., and Wang, Y. Optimal accounting of differential privacy via characteristic function. In *AISTATS*, pp. 4782–4817, 2022.

A. Proofs of Coupling Approximation Properties

For sake of completeness, we include proofs of the lemmas we use from Gopi et al. (2021).

Proof of Lemma 2.6(2). There exists couplings (X, Y) and (Y, Z) such that $\Pr[|X - Y| \geq h_1] \leq \eta_1$ and $\Pr[|Y - Z| \geq h_2] \leq \eta_2$. From these two couplings, we can construct a coupling between (X, Z): sample X, sample Y from $Y|X$ (given by coupling (X, Y)) and finally sample Z from $Z|Y$ (given by coupling (Y, Z)). Therefore for this coupling, we have:

$$\begin{aligned}\Pr[|X - Z| \geq h_1 + h_2] &\leq \Pr[|(X - Y) + (Y - Z) \geq h_1 + h_2] \\ &\leq \Pr[|X - Y| + |Y - Z| \geq h_1 + h_2] \\ &\leq \Pr[|X - Y| \geq h_1] + \Pr[|Y - Z| \geq h_2] \\ &\leq \eta_1 + \eta_2 \,.\end{aligned}$$

□

Proof of Lemma 2.6(3). The first part follows from the fact that there exists a coupling between X and Y such that $\Pr[X \neq Y] = d_{\mathrm{TV}}(X, Y)$. The second part follows from the first part and the fact that $d_{\mathrm{TV}}(X^{\oplus k}, Y^{\oplus k}) \leq k \cdot d_{\mathrm{TV}}(X, Y)$. □

Proof of Lemma 2.6(4). Let $X = Y - \widetilde{Y}$ where $(Y, \widetilde{Y})$ are coupled such that $|Y - \widetilde{Y} - \mu| \leq h$ with probability 1. Then $\mathbb{E}[X] = 0$ and $X \in [\mu - h, \mu + h]$ w.p. 1 and hence by Hoeffding's inequality,

$$\Pr\left[|X^{\oplus k}| \geq t\right] \leq 2\exp\left(-\frac{2t^2}{k(2h)^2}\right) = \eta\,, \qquad \text{for } t = h\sqrt{2k\log\frac{2}{\eta}}\,.$$

□

B. Two-Stage Heterogeneous Composition

We can handle composition of k different PRVs $Y^1, \ldots, Y^k$ with a slight modification to TwoStageSelfComposePRV as given in Algorithm 4. The approximation analysis remains similar as before. The main difference is that L_1 and L_2 are to be chosen as

$$\begin{aligned}L_1 &\geq O\left(\max\left\{\max_i\left\{\varepsilon_{Y^i}\left(\frac{\varepsilon_{\mathrm{err}}\delta_{\mathrm{err}}}{16k_1k_2^{1.5}}\right)\right\}\,,\ \max_t\left\{\varepsilon_{Y^{tk_1+1:(t+1)k_1}}\left(\frac{\varepsilon_{\mathrm{err}}\delta_{\mathrm{err}}}{64k_2^{1.5}}\right)\right\}\right\} + \frac{\varepsilon_{\mathrm{err}}}{\sqrt{k_2}}\right), \\ L_2 &\geq O\left(\max\left\{\varepsilon_{Y^{1:k}}\left(\frac{\varepsilon_{\mathrm{err}}\delta_{\mathrm{err}}}{16}\right) + \varepsilon_{\mathrm{err}}, L_1\right\}\right),\end{aligned}$$

where we denote $Y^{i:j} = Y^i \oplus \cdots \oplus Y^j$. In the case of $k_1 = k_2 = \sqrt{k}$ (assumed to be an integer) and $r = 0$, this leads to a final running time of

$$\widetilde{O}\left(\frac{\varepsilon_{\mathrm{up}} \cdot k^{1.25} \cdot \sqrt{\log(k/\delta_{\mathrm{err}})}}{\varepsilon_{\mathrm{err}}}\right),$$

where $\varepsilon_{\mathrm{up}}$ can be bounded as

$$\max\left\{\varepsilon_{Y^{1:k}}\left(\frac{\varepsilon_{\mathrm{err}}\delta_{\mathrm{err}}}{16}\right),\ \sqrt[4]{k}\cdot\max_t\,\varepsilon_{Y^{t\sqrt{k}+1:(t+1)\sqrt{k}}}\left(\frac{\varepsilon_{\mathrm{err}}\delta_{\mathrm{err}}}{64k^{0.75}}\right),\ \sqrt[4]{k}\cdot\max_i\varepsilon_{Y^i}\left(\frac{\varepsilon_{\mathrm{err}}\delta_{\mathrm{err}}}{16k^{1.25}}\right)\right\}\ +\ \varepsilon_{\mathrm{err}}.$$

C. Analysis of RecursiveSelfComposePRV

We establish coupling approximations between consecutive random variables in the sequence:

$$Y^{\oplus 2^t},\ Y_0^{\oplus 2^t},\ \widetilde{Y}_0^{\oplus 2^t},\ Y_1^{\oplus 2^{t-1}},\ \widetilde{Y}_1^{\oplus 2^{t-1}},\ Y_2^{\oplus 2^{t-2}},\ \ldots,\ Y_{t-1}^{\oplus 2},\ \widetilde{Y}_{t-1}^{\oplus 2},\ Y_t\,,$$

Coupling $Y^{\oplus 2^t}$ and $Y_0^{\oplus 2^t}$. Since $d_{\mathrm{TV}}(Y, Y_0) = \Pr[|Y| > L_1] =: \delta_0$, we have from Lemma 2.6(3) that

$$|Y^{\oplus 2^t} - Y_0^{\oplus 2^t}| \leq_{2^t\delta_0} 0\,. \tag{10}$$

Algorithm 4 TwoStageComposePRV for heterogeneous compositionss

Input: PRVs $Y^1, \ldots, Y^k$, number of compositions $k = k_1 \cdot k_2 + r$, $r < k_1$, mesh sizes $h_1 \le h_2$, truncation parameters $L_1 \le L_2$, where each $L_i \in h_i \cdot (\frac{1}{2} + \mathbb{Z}_{>0})$.

Output: PDF of an approximation $\widehat{Y}_2$ for $Y^1 \oplus \ldots \oplus Y^k$. $\widehat{Y}_2$ will be supported on $\mu + (h_2\mathbb{Z} \cap [-L_2, L_2])$ for some $\mu \in \left[-\frac{h_2}{2}, \frac{h_2}{2}\right]$.

for $i = 1$ to $k_1 k_2$ **do**
  $Y_0^i \leftarrow Y^i|_{|Y^i| \le L_1}$ ▷ Y^i conditioned on $|Y^i| \le L_1$
  $\widetilde{Y}_0^i \leftarrow \mathsf{DiscretizeRV}(Y_0^i, h_1, L_1)$
end for
for $i = k_1 k_2 + 1$ to k **do**
  $Y_0^i \leftarrow Y^i|_{|Y^i| \le L_2}$ ▷ Y^i conditioned on $|Y^i| \le L_2$
  $\widetilde{Y}_0^i \leftarrow \mathsf{DiscretizeRV}(Y_0^i, h_2, L_2)$
end for

for $t = 0$ to $k_2 - 1$ **do**
  $Y_1^t \leftarrow \widetilde{Y}_0^{tk_1+1} \oplus_{L_1} \cdots \oplus_{L_1} \widetilde{Y}_0^{(t+1)k_1}$ ▷ k_1-fold FFT convolution
  $\widetilde{Y}_1^t \leftarrow \mathsf{DiscretizeRV}(Y_1^t, h_2, L_2)$
end for

$Y_2 \leftarrow \widetilde{Y}_1^1 \oplus_{L_2} \cdots \oplus_{L_2} \widetilde{Y}_1^{k_2}$ ▷ k_2-fold FFT convolution
return $Y_2 \oplus_{L_2} \widetilde{Y}_0^{k_1 \cdot k_2 + 1} \oplus_{L_2} \cdots \oplus_{L_2} \widetilde{Y}_0^k$

Coupling $Y_i^{\oplus 2^{t-i}}$ and $\widetilde{Y}_i^{\oplus 2^{t-i}}$. We have from Proposition 2.7 that $\mathbb{E}[Y_i] = \mathbb{E}[\widetilde{Y}_i]$ and that $|Y_i - \widetilde{Y}_i - \mu_i| \le_0 \frac{h_{i+1}}{2}$ for some μ_i satisfying $|\mu_i| \le \frac{h_{i+1}}{2}$. Thus, applying Lemma 2.6(4), we have (for η to be chosen later) that

$$\left|Y_i^{\oplus 2^{t-i}} - \widetilde{Y}_i^{\oplus 2^{t-i}}\right| \le_\eta h_{i+1}\sqrt{2^{t-i-1}\log\frac{2}{\eta}} =: \widetilde{h}_{i+1}. \tag{11}$$

Coupling $\widetilde{Y}_i^{\oplus 2^{t-i}}$ and $Y_{i+1}^{\oplus 2^{t-i-1}}$. Since $d_{\mathrm{TV}}(\widetilde{Y}_i \oplus \widetilde{Y}_i, \widetilde{Y}_i \oplus_{L_{i+1}} \widetilde{Y}_i) \le \Pr[|\widetilde{Y}_i \oplus \widetilde{Y}_i| > L_{i+1}] =: \delta_{i+1}$, it holds via Lemma 2.6(3) that

$$|\widetilde{Y}_i^{\oplus 2^{t-i}} - Y_{i+1}^{\oplus 2^{t-i-1}}| \le_{2^{t-i-1}\delta_{i+1}} 0\,. \tag{12}$$

Putting things together. Thus, combining Equations (11) and (12) for $i \in \{0, \ldots, t-1\}$, using Lemma 2.6(3), we have that

$$\left|Y_0^{\oplus 2^t} - Y_t\right| \le_{\delta^*} \varepsilon^*, \tag{13}$$

$$\text{where} \quad \delta^* := \delta_t^* := t\eta + \textstyle\sum_{j=1}^t 2^{t-j}\delta_j\,, \tag{14}$$

$$\text{and} \quad \varepsilon^* := \varepsilon_t^* := \textstyle\sum_{j=1}^t h_j\sqrt{2^{t-j}\log\frac{2}{\eta}}.$$

More generally, the same analysis shows that for any $1 \le i \le t$,

$$\left|Y_0^{\oplus 2^i} - Y_i\right| \le_{\delta_i^*} \varepsilon_i^*,$$

$$\text{where} \quad \delta_i^* := i\eta + \textstyle\sum_{j=1}^i 2^{i-j}\delta_j\,, \tag{15}$$

$$\text{and} \quad \varepsilon_i^* := \textstyle\sum_{j=1}^i h_j\sqrt{2^{i-j}\log\frac{2}{\eta}}\,.$$

To simplify our analysis going forward, we fix the choice of h_i's that we will use, namely, $h_i = \frac{\varepsilon^*}{t\sqrt{2^{t-i}\log\frac{2}{\eta}}}$ (for η that will be chosen later). This implies that

$$\varepsilon_i^* \;=\; \sum_{j=1}^{i} h_j\sqrt{2^{i-j}\log\frac{2}{\eta}} \;=\; i\cdot\frac{\varepsilon^*}{t\sqrt{2^{t-i}}} \;=\; h_{i+1}\cdot i\sqrt{\frac{1}{2}\log\frac{2}{\eta}},$$

where the last step uses that $i \le t$.

In order to get our final bound, we need to bound δ_i's in terms of η, the mesh sizes h_i's, and truncation parameters L_i's. For ease of notation, we let $\delta_0^* = 0$. We have for $0 \le i < t$ that

$$\begin{aligned}
\delta_{i+1} \;&=\; \Pr\left[\left|\widetilde{Y}_i \oplus \widetilde{Y}_i\right| > L_{i+1}\right] \\
&\le\; \Pr\left[|Y_i \oplus Y_i| > L_{i+1} - 2h_{i+1}\right] \qquad \left(\text{since, } \left|Y_i - \widetilde{Y}_i\right| \le h_{i+1} \text{ w.p. } 1\right) \\
&\le\; 2\Pr\left[\left|Y_i - Y_0^{\oplus 2^i}\right| > \varepsilon_i^*\right] + \Pr\left[\left|Y_0^{\oplus 2^{i+1}}\right| > L_{i+1} - 2h_{i+1} - 2\varepsilon_i^*\right] \\
&\le\; 2\delta_i^* + \Pr\left[\left|Y^{\oplus 2^{i+1}}\right| > L_{i+1} - 2h_{i+1} - 2\varepsilon_i^*\right] \\
\delta_{i+1} \;&\le\; 2\delta_i^* + \frac{4}{\alpha_{i+1}}\delta_{Y^{\oplus 2^{i+1}}}(\widetilde{L}_{i+1}), \qquad (16)
\end{aligned}$$

where in the penultimate step we use that the tails of $Y_0^{\oplus 2^i}$ are no larger than tails of $Y^{\oplus 2^i}$ since Y_0 is a truncation of Y, and in the last step we use Lemma 2.4 with $\widetilde{L}_{i+1} := L_{i+1} - 2h_{i+1}(1 + i\sqrt{\frac{1}{2}\log\frac{2}{\eta}}) - \alpha_{i+1}$ (eventually we set $\alpha_{i+1} = h_{i+1}$).

We show using an inductive argument that for $C_i = 8^i$,

$$\delta_i \;\le\; 2C_i\cdot\left(\eta + \textstyle\sum_{j=1}^{i}\frac{4}{\alpha_j}\delta_{Y^{\oplus 2^j}}\left(\widetilde{L}_j\right)\right), \qquad (17)$$

$$\text{and} \qquad \delta_i^* \;\le\; C_{i+1}\cdot\left(\eta + \textstyle\sum_{j=1}^{i}\frac{4}{\alpha_j}\delta_{Y^{\oplus 2^j}}\left(\widetilde{L}_j\right)\right). \qquad (18)$$

The base case holds since $\delta_1 \le \frac{4}{\alpha_1}\delta_{Y^{\oplus 2}}(\widetilde{L}_1)$; note $C_1 > 1$. From (15), we have

$$\begin{aligned}
\delta_i^* \;&\le\; i\eta + \textstyle\sum_{j=1}^{i} 2^{i-j}\delta_j \\
&\le\; i\eta + \textstyle\sum_{j=1}^{i} 2^{i-j}\left(2C_j\cdot\left(\eta + \sum_{\ell=1}^{j}\frac{4}{\alpha_\ell}\delta_{Y^{\oplus 2^\ell}}\left(\widetilde{L}_\ell\right)\right)\right) \\
&\le\; i\eta + \left(\textstyle\sum_{j=1}^{i} 2^{i-j}\cdot 2C_j\right)\cdot\left(\eta + \sum_{j=1}^{i}\frac{4}{\alpha_j}\delta_{Y^{\oplus 2^j}}\left(\widetilde{L}_j\right)\right) \\
&\le\; i\eta + (4C_i)\cdot\left(\eta + \textstyle\sum_{j=1}^{i}\frac{4}{\alpha_j}\delta_{Y^{\oplus 2^j}}\left(\widetilde{L}_j\right)\right) \\
&\le\; C_{i+1}\cdot\left(\eta + \textstyle\sum_{j=1}^{i}\frac{4}{\alpha_j}\delta_{Y^{\oplus 2^j}}\left(\widetilde{L}_j\right)\right).
\end{aligned}$$

This completes the inductive step (18). Finally, (16) immediately implies the inductive step (17).

Putting this together in (14), and setting $\alpha_i = h_i$, we get

$$\delta_t^* \;\le\; \frac{1}{\varepsilon^*}\cdot 2^{O(t)}\left(\eta + \sum_{i=1}^{t}\delta_{Y^{\oplus i}}(\widetilde{L}_i)\right).$$

Finally, combining (13) with (10) using Lemma 2.6(2), we get

$$\begin{aligned}
&\left|Y^{\oplus 2^t} - Y_t\right| \;\le_{\delta_{\mathrm{err}}}\; \varepsilon_{\mathrm{err}} \\
\text{where} \qquad &\delta_{\mathrm{err}} \;:=\; \tfrac{1}{\varepsilon^*}\cdot\left(2^{O(t)}\left(\eta + \textstyle\sum_{i=1}^{t}\delta_{Y^{\oplus 2^i}}(\widetilde{L}_i)\right) + 2^{O(t)}\delta_Y(\widetilde{L}_1)\right), \\
\text{and} \qquad &\varepsilon_{\mathrm{err}} \;:=\; \varepsilon^*.
\end{aligned}$$

Thus, we get the desired approximation result for the following choice of parameters

$$\begin{aligned}
\eta &= \tfrac{\delta_{\mathrm{err}}}{2^{O(t)}}, \\
h_i &= \tfrac{\varepsilon_{\mathrm{err}}}{t\sqrt{2^{t-i}\log\frac{2}{\eta}}} = \Omega\left(\tfrac{\varepsilon_{\mathrm{err}}}{t^{1.5}\sqrt{2^{t-i}\log\frac{1}{\delta_{\mathrm{err}}}}}\right), \\
L_i &\geq \max\left\{\varepsilon_{Y\oplus 2^i}\left(\tfrac{\varepsilon_{\mathrm{err}}\delta_{\mathrm{err}}}{2^{O(t)}}\right) + h_i\cdot\left(3+2i\sqrt{\tfrac{1}{2}\log\tfrac{2}{\eta}}\right), L_{i-1}\right\}.
\end{aligned}$$

Thus, the overall running time is at most

$$\begin{aligned}
\widetilde{O}\left(\sum_i \frac{L_i}{h_i}\right) &= \widetilde{O}\left(\frac{\varepsilon_{\mathrm{up}}\cdot t^{2.5}\sqrt{\log\frac{1}{\delta_{\mathrm{err}}}}}{\varepsilon_{\mathrm{err}}}\right), \\
\text{where} \qquad \varepsilon_{\mathrm{up}} &:= \max_i\left(\sqrt{2^{t-i}}\cdot\varepsilon_{Y\oplus 2^i}\left(\tfrac{\varepsilon_{\mathrm{err}}\delta_{\mathrm{err}}}{2^{O(t)}}\right)\right).
\end{aligned}$$

Extensions of RecursiveSelfComposePRV. Similar to Appendix B, it is possible to extend RecursiveSelfComposePRV to handle the case where the number of compositions k is not a power of 2, with a similar asymptotic runtime. It can then be extended to handle heterogeneous compositions of k different mechanisms.