RRHL: A Relational Logic for Rényi Differential Privacy of Private Learning

GILLES BARTHE, Max Planck Institute for Security and Privacy (MPI-SP), Germany and IMDEA Software Institute, Spain
DAVIDE DAVOLI, Max Planck Institute for Security and Privacy (MPI-SP), Germany

Differential privacy is the standard framework for defining privacy-preserving computation and is adopted in large-scale production systems, including industrial-level private learning. The underlying algorithms are based on complex privacy-enhancing mechanisms such as privacy amplification by subsampling and correlated noising. Proving their privacy requires advanced reasoning techniques such as the moments accountant and shifted composition; no existing approach to formal verification supports all of them. In this work, we introduce RRHL: a relational program logic for proving Rényi differential privacy of imperative programs with continuous sampling instructions. Its judgments express bounds on the privacy budgets of instructions as functions of the program states, and its composition of budgets is sharp. We use RRHL to prove differential privacy of the algorithms widely implemented and deployed. These include several variants of DP-SGD and DP-FTRL, and gradient descent with per-record Rényi filters. In several cases the bounds match the best known Rényi differential privacy analyses. The logic, its soundness proof and the case studies are formalized in Lean. To the best of our knowledge, this is the first formal privacy analysis, via a relational program logic, of private learning algorithms with state-of-the-art Rényi differential privacy bounds.

CCS Concepts: • **Theory of computation → Logic and verification**; • **Security and privacy → Logic and verification**.

Additional Key Words and Phrases: probabilistic program logics, relational Hoare logic, Rényi differential privacy, DP-SGD, shifted composition, formal verification, Lean

1 Introduction

Differential privacy [33, 34] is the de facto gold standard for privacy-preserving computations. In a nutshell, a randomized computation is differentially private if its outcome is provably robust to modifications of the values associated with individuals. Thanks to its rigorous guarantees, differential privacy is deployed at scale. Two notable examples of deployment are the 2020 Decennial Census [2] by the U.S. Census Bureau, and the collection of user device statistics, or telemetry, by Google and Apple [7, 36]. Another prominent example of deployment is Google's keyboard Gboard, which predicts next words using private learning [56, 76]. Beyond these examples, many differentially private learning libraries are available for research and deployment [9, 47, 70, 77].

At their heart, private learning algorithms are generally based on DP-SGD [1, 22, 68] and DP-FTRL [49], both private adaptations of classical learning algorithms, respectively SGD (Stochastic Gradient Descent) and FTRL (Follow The Regularized Leader). These algorithms exploit advanced privacy techniques and advanced proof tools, such as *the moments accountant* [1], *privacy amplification by subsampling* [11, 59, 73], *shifted composition* [5], *correlated noising* [27, 32, 49] and *conversion lemmas* from Rényi differential privacy [12, 25, 58]. No existing verification approach supports all of these mechanisms and proof techniques; see Related work (§7).

This paper. The objective of this paper is to prove state-of-the-art Rényi differential privacy bounds for DP-SGD and DP-FTRL. To this end, we introduce a new program logic, called RRHL, to reason about differential privacy of an imperative language with sampling from continuous

Authors' Contact Information: Gilles Barthe, Max Planck Institute for Security and Privacy (MPI-SP), Bochum, Germany and IMDEA Software Institute, Madrid, Spain, gilles.barthe@mpi-sp.org; Davide Davoli, Max Planck Institute for Security and Privacy (MPI-SP), Bochum, Germany.

distributions. Judgments of the logic are of the form

$$\vdash \{\mathcal{R} \mid \phi\}\ c_1 \sim_\alpha c_2\ \{\mathcal{S} \mid \psi\},$$

where the *pre-* and *post-condition* $\mathcal{R}$ and $\mathcal{S}$ are relations over pairs of states, $\alpha \in (1, \infty)$ is a Rényi differential privacy order, and ϕ, ψ are *privacy budgets* that map pairs of states to non-negative extended reals. The use of state-dependent quantitative pre- and post-conditions, inspired by expectation-based Hoare logics [3, 8, 14, 15, 51, 55], enables a precise and compositional accounting of the privacy cost: the *pre-budget* ϕ is an upper bound on the privacy cost and the *post-budget* ψ is an upper bound on what remains of it after executing both programs. Finally, similar to other quantitative relational Hoare logics [20, 21, 64], RRHL's judgments are expressive enough to capture the corresponding notion of differential privacy, here Rényi differential privacy. In order to carry out proofs, we equip RRHL with proof rules for the usual deterministic program constructs, novel rules for random samplings, and two additional ingredients: a syntactic construct to declare local variables and an equational theory which allows one to merge/unmerge samplings akin to the associativity rule for monadic bind. Together, these features make it possible to capture the moments accountant, and to reason about amplification by subsampling, shifted composition, and correlated noising. We briefly recall each one and explain how RRHL captures it:

- The moments accountant is a state-of-the-art *compositional* privacy accounting mechanism whose composition of privacy bounds is sharp. RRHL's rules for sequential composition and loops are compositional and sharp in the same sense, and directly capture the moments accountant.
- Amplification by subsampling consists in running a differentially private algorithm on a randomly sampled subset of its input dataset. RRHL reasons about such programs via the rule for local variables and equivalences that collapse the sequence of dependent samplings into a single sampling, for which the amplified bound can be proved.
- The shifted composition analysis bounds the privacy loss of iterative algorithms by tracking, alongside the divergence between the two executions, the distance between their states. RRHL performs proofs by shifted composition by tracking this distance in its logical pre- and post-conditions.
- Correlated noising enhances privacy by making the noise added to released values (e.g., trained models) dependent on noise used in previous rounds. RRHL can analyse algorithms that use this technique because it keeps track of the privacy budget at each sampling statement, when noise is generated, and not only when data is released.

We use RRHL to prove Rényi differential privacy bounds on several learning algorithms:

- DP-SGD [1, 22, 68], a classic learning algorithm that uses subsampling, and is implemented in several private learning libraries [46, 47, 70, 77]. We prove its privacy bound under the *transcript model*, where the adversary sees every intermediate model, and under the *last-iterate model*, where the adversary only sees the final model. Both proofs account for subsampling, which is essential for precise bounds. The second one uses the shifted composition technique, and yields a budget independent of the number of iterations.
- DP-GD (differentially private gradient descent) with per-record Rényi filters [38]. The algorithm holds a privacy budget for each record, spends part of it at every round the record takes part in, and excludes the record once its budget is exhausted. The proof takes advantage of the quantitative judgments of RRHL to express the privacy budget as a state-dependent invariant.
- DP-FTRL [32, 49], the federated algorithm deployed in Gboard [76]. The algorithm is designed for online training, where the data available at each round is not known in

advance. It achieves its privacy guarantees by adding correlated noise to the released models. We consider two variants of this algorithm: one with subsampling and one without. Without subsampling, our analysis uses state-dependent budgets and encompasses previous results [28, 32, 49]. With subsampling, we prove the bound of Choquette-Choo et al. [27] but with a different subsampling mechanism, for uniformity with the other case studies.

In all these cases, our analyses match the Rényi differential privacy state-of-the-art bounds, except for the last-iterate DP-SGD where we derive the bound of Altschuler and Talwar [6] up to a constant, under additional assumptions due to the different subsampling mechanism we consider. Our proofs apply to algorithmic descriptions written in an idealized probabilistic programming language rather than to real-world implementations. Lifting the guarantees to implementations is future work.

Summary of contributions. We make the following contributions:

- we introduce RRHL and prove its soundness with respect to the denotational semantics of programs. The proof of soundness rests on a dual characterization of Rényi differential privacy based on pairs of tests, i.e., maps from states to non-negative extended reals. The duality is novel in the continuous setting. The test-based interpretation is more general than the two-witness lifting [21, 64], and more convenient to work with. The two coincide when the post-condition is a closed set and the post-budget is 1;
- using RRHL we formally prove privacy of the private learning algorithms behind widely used libraries, namely DP-SGD under the transcript and last-iterate models, DP-GD with per-record Rényi filters, DP-FTRL, and DP-FTRL with subsampling;
- we formalize our results in the Lean proof assistant; throughout the paper, the mark λ (a λ in a blue box) flags each formalized definition and machine-checked result. The formalization is publicly available at https://github.com/davidedavoli/RRHL.

2 Motivating Example: Differentially Private SGD

DP-SGD [1, 22, 68] is a widely deployed variant of SGD that provides differential privacy guarantees. It is implemented in private learning libraries such as Opacus [77] for PyTorch, TensorFlow Privacy [70], and JAX Privacy [47], among others. The interest in DP-SGD is not merely academic: it was used to train deployed models such as the open-weights VaultGemma [67], with 1B parameters.

To make precise the guarantees that this algorithm targets, we first recall the definition of differential privacy. Differential privacy is a formal framework for expressing privacy of randomized algorithms M, modelled as functions from an input space A to probability distributions over a space B. Running M on two adjacent inputs $a_1 \simeq a_2$ (typically two datasets that differ in one record, which is missing in one of the two) must yield two distributions that are similar w.r.t. some notion of similarity. The most popular such notion is (ϵ, δ)-differential privacy, in which the degree of similarity depends on the privacy budgets ϵ, δ, as follows:

Definition 2.1 ((ϵ, δ)-differential privacy [33] λ**).** A randomized algorithm M is (ϵ, δ)-*differentially private* with respect to the adjacency relation $\simeq$ when, for all inputs $a_1 \simeq a_2$ and every event $\mathcal{E} \subseteq B$,

$$\Pr_{M(a_1)}[\mathcal{E}] \;\leq\; \exp(\epsilon)\, \Pr_{M(a_2)}[\mathcal{E}] + \delta\,.$$

Figure 1 shows DP-SGD. We explain how each step contributes to its privacy. Its input D is a dataset of fixed size $|D| \in \mathbb{N}$ with records in $\mathbb{R}^d_\perp \triangleq \mathbb{R}^d \uplus \{\perp\}$, where $\perp$ marks a *missing* record, whose gradient $\nabla\ell$ for the loss function ℓ is 0, so that it contributes nothing to a sum of gradients. The other inputs are the number of learning rounds T; the step size $\eta \in \mathbb{R}_{>0}$; the maximum gradient norm $\Gamma \in \mathbb{R}_{>0}$; the variance $v \in \mathbb{R}_{>0}$ of the noise added at each round; and the subsampling rate $q \in (0, 1)$, i.e., the probability that each record of D participates in a round.

```
1: input dataset D; parameters T, η, Γ, v, q, θ0, Π, Δ
2: θ ← θ0; t ← 0
3: while t < T do
4:     local b, B, c, g in (
5:         Poisq;
6:         g ← Σ_{i∈B} clip_Γ(∇ℓ(θ, D[i]));
7:         w ←$ N(g, v))
8:     θ ← Π(θ − η/(q|D|) w); release w
9:     t ← t + 1
10: od
11: release θ
```

where Pois_q is

```
B ← ∅; c ← 0
while c < |D| do
    b ←$ Bern(q)
    if b then B ← B ∪ {c} else skip fi
    c ← c + 1
od
```

Fig. 1. DP-SGD algorithm.

After initializing the model θ to θ_0 and the round counter t at line 2, each round proceeds as follows. Line 5 draws a batch B of indices by *Poisson subsampling* (the loop Pois_q at the bottom of Figure 1): every index enters B independently with probability q. Subsampling improves the privacy guarantee, as, when running on adjacent datasets, the differing record takes part in any given round with small probability q. Line 6 sums the gradients of the batch, clipping each of them with $\mathrm{clip}_\Gamma(u) = u \cdot \min(1, \Gamma/\|u\|)$, so that its norm is at most Γ. On adjacent datasets and at the same model θ, the two sums thus differ by at most Γ in norm, which caps the impact of the differing record on the model. Line 7 adds Gaussian noise to g, as it samples w from the Gaussian distribution $\mathcal{N}(g, v)$ centred at g of variance v. The variance sets the trade-off between privacy and accuracy: larger values obfuscate the impact of the differing record, giving stronger privacy at the price of accuracy. The block of statements at lines 5–7 is within a `local` construct that makes the declared variables local to that block; this allows us to replace the block by a single sampling statement, which is needed to obtain the right bound (§5.1.1). Line 8 updates the model with step size $\eta/(q|D|)$. The result is then projected by a non-expansive projection Π onto a convex set of diameter Δ, which keeps the models bounded. The `release` statement appends its argument to a *transcript* of released values that represents the adversary's view of the algorithm. Depending on what is released, we capture two adversary models. In the *transcript* model, the boxed statement at line 8 is present, so the transcript is extended with the noisy gradient at every iteration. In the *last-iterate* model, the boxed statement is omitted and the adversary sees only the final model, released at line 11.

The main purpose of this work is to prove formally the privacy guarantees of algorithms such as DP-SGD. There are three main difficulties:

- *Rényi differential privacy analysis.* The privacy bound of DP-SGD is proven by reasoning about *Rényi differential privacy* [1, 58, 59]. The advantage of Rényi differential privacy is that the privacy budget of the loop can be obtained by the composition of the privacy budget of each iteration, without approximation. Said otherwise, Rényi differential privacy naturally captures the moments accountant, a technique used to track bounds precisely.
- *Conversion to (ϵ, δ)-differential privacy [12, 25].* In deployed instances of DP-SGD, the Rényi differential privacy coefficients are converted to (ϵ, δ)-differential privacy coefficients and reported. This conversion is lossy, and many conversion formulas have been proposed. In this work, we formally verify a state-of-the-art conversion result [12, 25], and we use it to convert the coefficients we obtain through RRHL to (ϵ, δ)-differential privacy coefficients.
- *Privacy amplification by subsampling [11, 59, 73].* The Poisson subsampling of line 5 strengthens the differential privacy guarantee of DP-SGD, because the differing record participates in training only with small probability. This amplification is a property of the joint draw of the batch and the noise, not of either draw alone, so accounting for it by dealing with the two sampling instructions separately is suboptimal. The `local` block of Figure 1 makes the

batch variables local, so that program equivalences can merge the Poisson sampling and the noise sampling into a single sampling statement.

§5.1 shows the derivation of the privacy guarantees of DP-SGD in RRHL.

3 Background

This section recalls the measure-theoretic notions used throughout the paper (§3.1, with the lemmas used in the proofs in Appendix A), the Hellinger integral, the divergence on which Rényi differential privacy builds and which serves as the semantic foundation for RRHL (§3.2), and Rényi differential privacy itself (§3.3), together with its individual variant (§3.4).

3.1 Measure-Theoretic Preliminaries

Measurable spaces. A *measurable space* is a set A equipped with a σ-algebra of *measurable* subsets: a collection that contains A and is closed under complements and countable unions. A function between measurable spaces is *measurable* if the preimage of every measurable set is measurable. A *measure* on A is a map μ from the measurable sets of A to $[0,\infty]$ that is *countably additive*: $\mu\big(\bigcup_i X_i\big) = \sum_i \mu(X_i)$ for every countable family $(X_i)_i$ of pairwise disjoint measurable sets, and satisfies $\mu(\emptyset) = 0$. It is *finite* if $\mu(A) < \infty$, a *sub-probability measure* if $\mu(A) \leq 1$ and a *probability measure* if $\mu(A) = 1$. We write $\mathrm{D}(A)$ for the set of sub-probability measures on A, $\mathbf{0} \triangleq E \mapsto 0$ for the zero measure, and δ_a for the *Dirac measure* at a, which gives mass 1 to the sets containing a and 0 to the others. Measures are added and rescaled setwise, $(\mu + \nu)(E) \triangleq \mu(E) + \nu(E)$ and $(c \cdot \mu)(E) \triangleq c\,\mu(E)$; for instance, $(1-q)\,\mu + q\,\nu$ is the law of a draw from ν with probability q and from μ otherwise. The *integral* $\int f \,\mathrm{d}\mu$ of a measurable $f : A \to [0,\infty]$ against μ, also written $\mathbb{E}_\mu[f]$ or $\mathbb{E}_{a\leftarrow\mu}[f(a)]$, is defined as usual (Definition A.7) and is linear and monotone in f. A set is *μ-null* if it is contained in a measurable set of measure 0, and a property holds *μ-almost everywhere* if it fails only on a μ-null set. The *pushforward* of μ along a measurable $f : A \to B$ is the measure $f^\sharp\,\mu \triangleq E \mapsto \mu(f^{-1}(E))$ on B; for μ on $\mathbb{R}^d$ and $a \in \mathbb{R}^d$, we write $\mu + a$ for the pushforward of μ along the translation $x \mapsto x + a$.

Sub-Giry monad. Sub-probability measures form a monad, the *sub-Giry monad* [40], which is the standard structure for interpreting probabilistic programs in the continuous setting. A *sub-Markov kernel* from A to B is a map $M : A \to \mathrm{D}(B)$ such that $a \mapsto M(a)(E)$ is measurable for every measurable $E \subseteq B$; it is a *Markov kernel* when every $M(a)$ is a probability measure. The *unit* of the monad sends $a \in A$ to δ_a, and the *bind* of $\mu \in \mathrm{D}(A)$ with a sub-Markov kernel M from A to B is the sub-probability measure on B defined by:

$$\big(\mathbf{dlet}\ a \leftarrow \mu\ \mathbf{in}\ M(a)\big)(E) \ \triangleq\ \int M(a)(E)\,\mathrm{d}\mu(a)\,.$$

Here the measurability condition on M makes the integral well defined. Unit and bind satisfy the monad laws, and binding a kernel with a kernel yields a kernel (Proposition A.17).

Density. Densities are used to define the Hellinger integral in §3.2: given a measure ϱ on A and a measurable $f : A \to [0,\infty]$, the map $E \mapsto \int f \cdot \mathbf{1}_E \,\mathrm{d}\varrho$, where $\mathbf{1}_E$ is the indicator function of E, is again a measure on A, and f is called its *density* with respect to ϱ. Given two finite measures μ and ϱ over the same space A, μ is *absolutely continuous* with respect to ϱ, written $\mu \ll \varrho$, when every ϱ-null set is μ-null. In that case μ has a density with respect to ϱ, written $\frac{\mathrm{d}\mu}{\mathrm{d}\varrho}$, which is unique up to a ϱ-null set (Theorem A.19).

3.2 Hellinger Integral

The Hellinger integral quantifies how far apart two measures are. We define it directly in its weighted form, which is the one used by RRHL's judgments.

Definition 3.1 (Weighted Hellinger integral 🅛). Let $\alpha > 1$ and let μ and ν be finite measures on a measurable space A. The *Hellinger integral* of order α between μ and ν, weighted by a measurable $w : A \to [0, \infty]$, is

$$\mathrm{H}_\alpha^w(\mu \,\|\, \nu) \triangleq \int w \cdot \left(\frac{\mathrm{d}\mu}{\mathrm{d}\nu}\right)^\alpha \mathrm{d}\nu$$

if $\mu \ll \nu$, and ∞ otherwise. We write $\mathrm{H}_\alpha(\mu \,\|\, \nu)$ for the integral at $w = 1$.

In this definition and throughout the paper, arithmetic $[0, \infty]$ follows the usual conventions $0 \cdot \infty = 0$, $0^{1-\alpha} = \infty$ and $\infty^{1-\alpha} = 0$, as $1 - \alpha < 0$.

The Hellinger integral also admits a dual characterization which replaces reasoning about densities by reasoning about expectations of test functions. The judgments of RRHL are interpreted through this characterization (§4.2)

Theorem 3.2 (Dual characterization 🅛). *Let $\alpha > 1$, let μ and ν be finite measures on a measurable space A, and let $w : A \to [0, \infty]$ be measurable, everywhere non-zero and everywhere finite. Then*

$$\mathrm{H}_\alpha^w(\mu \,\|\, \nu) = \sup_{(k_1, k_2)} \big(\mathbb{E}_\mu[k_1]\big)^\alpha \big(\mathbb{E}_\nu[k_2]\big)^{1-\alpha},$$

where the supremum ranges over the pairs of tests $k_1, k_2 : A \to [0, \infty]$ such that $k_1(a)^\alpha k_2(a)^{1-\alpha} \leq w(a)$ for every $a \in A$.

Crucially, the Hellinger integral composes along the bind of the sub-Giry monad: the integral between **dlet** $a \leftarrow \mu$ **in** $M(a)$ and **dlet** $a \leftarrow \nu$ **in** $N(a)$ is bounded by the integral between μ and ν, weighted by those between $M(a)$ and $N(a)$:

Theorem 3.3 (Composition 🅛). *Let $\alpha > 1$, let μ and ν be sub-probability measures on a measurable space A, and let M and N be sub-Markov kernels from A to a measurable space B. Let $w : B \to [0, \infty]$ be measurable, everywhere non-zero and bounded, and let $w' : A \to [0, \infty]$ be measurable and such that $\mathrm{H}_\alpha^w(M(a) \,\|\, N(a)) \leq w'(a)$ for every $a \in A$. Then*

$$\mathrm{H}_\alpha^w\big(\textbf{dlet } a \leftarrow \mu \textbf{ in } M(a) \,\big\|\, \textbf{dlet } a \leftarrow \nu \textbf{ in } N(a)\big) \leq \mathrm{H}_\alpha^{w'}(\mu \,\|\, \nu)\ .$$

This result underlies the moments accountant, which composes the same bounds on a logarithmic scale, and RRHL's sequencing rule (§4.3). The bound is attained on product measures [72, Theorem 28]: in this sense RRHL's sequencing rule is sharp.

3.3 Rényi Differential Privacy

Like (ϵ, δ)-differential privacy, Rényi differential privacy bounds the *dissimilarity* of the outputs of a randomized algorithm on two adjacent inputs, measured by the Rényi divergence:

Definition 3.4 (Rényi divergence [61], Rényi differential privacy [58] 🅛). Let $\alpha > 1$ and let μ and ν be finite measures on a measurable space A. The *Rényi divergence* of order α between μ and ν is

$$\mathrm{R}_\alpha(\mu \,\|\, \nu) \triangleq \frac{1}{\alpha - 1} \log \mathrm{H}_\alpha(\mu \,\|\, \nu)\,.$$

A sub-Markov kernel M is (α, ρ)-*Rényi differentially private* with respect to an adjacency relation $\simeq$, or (α, ρ)-*RDP*, when $\mathrm{R}_\alpha(M(a_1) \,\|\, M(a_2)) \leq \rho$ for every pair of inputs $a_1 \simeq a_2$.

Note that in the previous definition, the algorithm is a sub-Markov kernel and is therefore allowed to diverge with non-zero probability. This definition departs from the standard one, and coincides with it for programs that terminate with probability 1. Ensuring this condition is orthogonal to the scope of this work, and is typically not addressed by other probabilistic relational Hoare logics [15, 19]. All the case studies of §5 trivially always terminate, as their loops are bounded.

As recalled in §2, Rényi differential analyses are meant to provide (ϵ, δ)-differential privacy bounds, that are obtained with a conversion to (ϵ, δ) coefficients (Definition 2.1). Although the conversion is not tight, in practice gives better coefficients than composing (ϵ, δ) guarantees [1, 58]. We use the following:

Proposition 3.5 (Conversion [12, Theorem 21] **Λ**). *If M is (α, ρ)-RDP, then for every $\delta \in (0, 1)$ it is $\left(\epsilon_{\alpha,\rho}(\delta),\ \delta\right)$-differentially private, where $\epsilon_{\alpha,\rho}(\delta) = \rho + \log \frac{\alpha-1}{\alpha} - \frac{\log \delta + \log \alpha}{\alpha-1}$.*

This result is used by deployed privacy accounting mechanisms [9, 23, 41, 46, 52, 70, 71, 77]; Appendix F gives the full list.

3.4 Individual Rényi Differential Privacy

In standard Rényi differential privacy, the privacy budget is a *worst-case* bound, measuring the maximum privacy loss across all adjacent input pairs. However, some applications, such as the Rényi filters of §5.2, benefit from *per-individual* accounting. This allows users to track their own budget consumption individually and enables the algorithm to query average individuals more frequently, as they incur less privacy degradation, improving the overall privacy/utility trade-off. To capture these guarantees, Feldman and Zrnic [38] introduced *individual* Rényi differential privacy:

Definition 3.6 (Individual Rényi differential privacy [38] **Λ**). Let $(\simeq_k)_k$ be a family of adjacency relations, indexed by the individuals, where $a_1 \simeq_k a_2$ relates inputs that differ only in the data of k. A sub-Markov kernel M is (α, ρ)-*individually Rényi differentially private* for a family of budgets $\rho = (\rho_k)_k$ when it is (α, ρ_k)-RDP with respect to $\simeq_k$ for every individual k.

From individual Rényi differential privacy guarantees, it is always possible to derive *global* ones:

Lemma 3.7 (From individual to global Rényi differential privacy **Λ**). *Let M be $(\alpha, (\rho_k)_k)$-individually Rényi differentially private for a family $(\simeq_k)_k$. Then M is $\left(\alpha, \sup_k \rho_k\right)$-RDP with respect to $\bigcup_k \simeq_k$.*

We use this lemma in our analyses (§5.1 and Appendix E.5), where adjacency at a fixed record simplifies the proofs: the index j of the differing record is known, so the invariants can refer to it.

4 The Logic RRHL

We now introduce the *Rényi Relational Hoare Logic* RRHL. Its judgments pair Boolean pre- and post-conditions relating the two runs with budgets that track the privacy cost. §4.1 presents the language, §4.2 the judgments, §4.3 the proof system, and §4.4 soundness and the link to Rényi DP.

4.1 Programs

Let Var $= \{x, y, z, \dots\}$ be a countable set of *variables*. Statements Stmt are given by:

$$\begin{aligned} c, c' \ ::= \ & \mid \texttt{skip} \mid x \leftarrow E \mid x \overset{\$}{\leftarrow} S \mid \texttt{release}\ E \mid \texttt{local}\ x\ \texttt{in}\ c \\ & \mid \texttt{if}\ P\ \texttt{then}\ c\ \texttt{else}\ c'\ \texttt{fi} \mid \texttt{while}\ P\ \texttt{do}\ c\ \texttt{od} \mid c; c' \,. \end{aligned}$$

Here $E \in$ Expr, $P \in$ BExpr and $S \in$ SExpr range over *expressions*, *Boolean expressions* and *sampling expressions*. We leave these languages abstract for generality. In the following, we use conventional mathematical notation to represent them.

$$
\begin{aligned}
[\![\texttt{skip}]\!](m,\tau) &\triangleq \delta_{(m,\tau)}\\
[\![x \leftarrow E]\!](m,\tau) &\triangleq \delta_{(m[x\mapsto [\![E]\!](m)],\tau)}\\
[\![x \xleftarrow{\$} S]\!](m,\tau) &\triangleq \mathbf{dlet}\ v \leftarrow [\![S]\!](m)\ \mathbf{in}\ \delta_{(m[x\mapsto v],\tau)}\\
[\![\texttt{release}\ E]\!](m,\tau) &\triangleq \delta_{(m,\tau\cdot[\![E]\!](m))}\\
[\![\texttt{local}\ x\ \texttt{in}\ c]\!](m,\tau) &\triangleq [\![x \leftarrow d_x; c; x \leftarrow d_x]\!](m,\tau)\\
[\![c;c']\!](m,\tau) &\triangleq \mathbf{dlet}\ (m',\tau') \leftarrow [\![c]\!](m,\tau)\ \mathbf{in}\ [\![c']\!](m',\tau')\\
[\![\texttt{if}\ P\ \texttt{then}\ c\ \texttt{else}\ c'\ \texttt{fi}]\!](m,\tau) &\triangleq \begin{cases} [\![c]\!](m,\tau) & \text{if } [\![P]\!](m),\\ [\![c']\!](m,\tau) & \text{otherwise}\end{cases}\\
[\![\texttt{while}\ P\ \texttt{do}\ c\ \texttt{od}]\!](m,\tau) &\triangleq \sup_n W_n(m,\tau)\ \text{ where:}
\end{aligned}
$$

$$
W_0(m,\tau) \triangleq \mathbf{0} \qquad W_{n+1}(m,\tau) \triangleq \begin{cases} \mathbf{dlet}\ (m',\tau') \leftarrow [\![c]\!](m,\tau)\ \mathbf{in}\ W_n(m',\tau') & \text{if } [\![P]\!](m),\\ \delta_{(m,\tau)} & \text{otherwise.}\end{cases}
$$

Fig. 2. Semantics of statements.

Semantics. To model sampling from continuous distributions, programs c are interpreted as sub-Markov kernels $[\![c]\!] : \mathrm{St} \to \mathrm{D}(\mathrm{St})$ on states, where the mass missing from $[\![c]\!](s)$ is the probability of non-termination on the state s. In turn, a state $s = (m,\tau)$ consists of a *memory* $m \in \mathrm{Mem}$, modelled as a total map $\mathrm{Var} \to \mathrm{Val}$, and a *transcript* $\tau \in \mathrm{Tr} \triangleq \bigcup_n \mathrm{Val}^n$, representing the sequence of the values released so far, which models what the adversary learns. We write $\langle\rangle$ for the empty transcript, $\tau \cdot v$ for τ extended with v, and $m[x \mapsto v]$ for the memory that gives x the value v and agrees with m elsewhere. Figure 2 defines $[\![c]\!]$. The definition uses the expression semantics, which we leave abstract. For well-definedness, we only require $[\![E]\!] : \mathrm{Mem} \to \mathrm{Val}$ and $[\![P]\!] : \mathrm{Mem} \to \mathbb{B}$, with $\mathbb{B} \triangleq \{0,1\}$, to be measurable and $[\![S]\!] : \mathrm{Mem} \to \mathrm{D}(\mathrm{Val})$ to be a sub-Markov kernel.

The statement `skip` leaves the state unchanged; $x \leftarrow E$ updates the memory at x with $[\![E]\!](m)$; $x \xleftarrow{\$} S$ draws v from $[\![S]\!](m)$ and updates the memory at x with it; `release` E appends $[\![E]\!](m)$ to the transcript. The statement `local` x `in` c runs c with x at its default value $d_x \in \mathrm{Val}$ and resets x to d_x afterwards, so that the values x takes in c are not visible outside. Sequential composition binds the kernel of the second statement to the output measure of the first; the conditional runs the branch selected by $[\![P]\!](m)$; the loop is the supremum of its n-step unrollings W_n. As $\{W_n\}_{n\in\mathbb{N}}$ is an increasing sequence, its supremum exists and is again a sub-probability measure (Lemma A.8).

4.2 Judgments

RRHL's judgments have the following shape:

$$\{\mathcal{R} \mid \phi\}\ c_1 \sim_\alpha c_2\ \{\mathcal{S} \mid \psi\},$$

where c_1 and c_2 are programs, the *pre-* and *post-conditions* $\mathcal{R}, \mathcal{S} \subseteq \mathrm{St} \times \mathrm{St}$ are relations on pairs of states, the *pre-* and *post-budgets* are functions $\phi, \psi : \mathrm{St} \times \mathrm{St} \to [0,\infty]$, and $\alpha > 1$ is the judgment's *order*. Intuitively, if the initial states are related by $\mathcal{R}$, the output distributions are related by an *approximate* lifting of $\mathcal{S}$, whose degree of approximation depends on the budgets. When $\mathcal{S}$ identifies the two final states, the pre-budget ϕ is an upper bound on the Hellinger integral of order α between the two output distributions, weighted by ψ, as in Theorem 3.2. This interpretation makes RRHL suitable for establishing Rényi differential privacy guarantees (Proposition 4.4). For a general $\mathcal{S}$, validity relaxes the admissibility of tests in Theorem 3.2 from pairs of equal states to $\mathcal{S}$-related pairs, so that ϕ bounds a generalization of that integral. Formally, our interpretation is the following one:

Definition 4.1 (Validity). Let $\alpha > 1$. The judgment $\{\mathcal{R} \mid \phi\}\ c_1 \sim_\alpha c_2\ \{\mathcal{S} \mid \psi\}$ is *valid*, written $\models \{\mathcal{R} \mid \phi\}\ c_1 \sim_\alpha c_2\ \{\mathcal{S} \mid \psi\}$, if for every $s = (s_1, s_2) \in \mathcal{R}$ and every pair of tests (k_1, k_2) that is admissible for $\mathcal{S}$ and ψ, we have:

$$\left(\mathbb{E}_{[\![c_1]\!](s_1)}[k_1]\right)^\alpha \left(\mathbb{E}_{[\![c_2]\!](s_2)}[k_2]\right)^{1-\alpha} \ \le\ \phi(s)\,.$$

In turn, a *test* is a measurable function from states to $[0, \infty]$, and a pair of tests (k_1, k_2) is *admissible* for a post-condition $\mathcal{S}$ and a post-budget ψ if

$$k_1(s_1)^\alpha\, k_2(s_2)^{1-\alpha} \ \le\ \psi(s_1, s_2) \qquad \text{for every } (s_1, s_2) \in \mathcal{S}\,,$$

with the conventions of Definition 3.1 for $0 \cdot \infty$ and $0^{1-\alpha}$.

Our notion of validity differs from the one adopted by other relational logics for differential privacy [20, 21, 64]. There, validity on $\mathcal{R}$-related input states s_1, s_2 requires the existence of two *witness* distributions $\mu_1, \mu_2 \in \mathsf{D}(\mathrm{St} \times \mathrm{St})$ where the relation $\mathcal{S}$ holds μ_1- and μ_2-almost everywhere, and whose projections are the two program semantics: $\pi_1^\sharp\, \mu_1 = [\![c_1]\!](s_1)$ and $\pi_2^\sharp\, \mu_2 = [\![c_2]\!](s_2)$. On the quantitative side, validity asks that the privacy budget bounds a divergence between μ_1 and μ_2 (e.g., a skewed distance or a Rényi divergence, depending on the target differential privacy notion). With that interpretation, proving soundness of the traditional sequencing rule

$$\frac{\vdash \{\mathcal{R} \mid \phi\}\ c_1 \sim_\alpha c_2\ \{\mathcal{T} \mid \chi\} \qquad \vdash \{\mathcal{T} \mid \chi\}\ c_1' \sim_\alpha c_2'\ \{\mathcal{S} \mid \psi\}}{\vdash \{\mathcal{R} \mid \phi\}\ c_1; c_1' \sim_\alpha c_2; c_2'\ \{\mathcal{S} \mid \psi\}}$$

in the continuous setting requires combining the two initial witnesses with a pair of witnesses for every intermediate state, and the construction only works if the latter are chosen measurably in the intermediate states [64]. This *measurable selection* assumption must either be built into the interpretation or threaded through the soundness proof. Our interpretation quantifies over no witnesses, so our soundness proofs need no measurable selection principle. Our notion of validity is more general than the witness-based one, as expressed below:

Proposition 4.2 (Weak duality). *Let $\alpha > 1$. Assume that for every $s = (s_1, s_2) \in \mathcal{R}$ there exist $\mu_1, \mu_2 \in \mathsf{D}(\mathrm{St}\times\mathrm{St})$ such that $\mathcal{S}$ holds μ_1-almost everywhere and μ_2-almost everywhere, $\pi_1^\sharp\, \mu_1 = [\![c_1]\!](s_1)$, $\pi_2^\sharp\, \mu_2 = [\![c_2]\!](s_2)$ and $\mathrm{H}_\alpha^\psi(\mu_1 \,\|\, \mu_2) \le \phi(s)$. Then $\models \{\mathcal{R} \mid \phi\}\ c_1 \sim_\alpha c_2\ \{\mathcal{S} \mid \psi\}$.*

The converse holds at post-budget 1 when $\mathcal{S}$ is a closed set (see Appendix D).

Notation. Assertions are represented as first-order formulas over program and logical variables; budgets are $[0, \infty]$-valued terms over the same vocabulary. In assertions, the dedicated variable τ denotes the transcript. Because assertions relate two states, program variables and transcripts are tagged with $\cdot^{\langle 1\rangle}$ when they are evaluated in the left state, and $\cdot^{\langle 2\rangle}$ when they are evaluated in the right state. Logical variables are untagged, implicitly universally quantified, and denote identical values on both sides. Following standard conventions, we write $s \models \mathcal{R}$ for $s \in \mathcal{R}$; we write $\mathcal{R} \models \mathcal{T}$ when $(s_1, s_2) \models \mathcal{T}$ for every $(s_1, s_2) \in \mathcal{R}$. Substitution in an assertion is interpreted so that $\mathcal{S}[x{\mapsto}E^{\langle 1\rangle}]$ holds at $((m_1, \tau_1), s_2)$ when $\mathcal{S}$ holds at $((m_1[x \mapsto [\![E]\!](m_1)], \tau_1), s_2)$; similarly, $\mathcal{S}[\tau_1{\mapsto}\tau_1 \cdot E^{\langle 1\rangle}]$ holds at $((m_1, \tau_1), s_2)$ when $\mathcal{S}$ holds at $((m_1, \tau_1 \cdot [\![E]\!](m_1)), s_2)$. Budgets are substituted in the same way: $\phi[x{\mapsto}E^{\langle 1\rangle}]$ is the budget whose value at $((m_1, \tau_1), s_2)$ is $\phi((m_1[x \mapsto [\![E]\!](m_1)], \tau_1), s_2)$. Substitutions on the right state are defined analogously.

4.3 Deductive System

Figure 3 gives the rules of RRHL: core rules in blue, rules derived from them in green. A judgment is *derivable*, written with $\vdash$, when it is the conclusion of a derivation built from the rules of the logic.

Structural rules

$$\frac{}{\vdash \{\mathcal{R} \mid \phi\}\ \texttt{skip} \sim_\alpha \texttt{skip}\ \{\mathcal{R} \mid \phi\}}\ [\textsc{Skip}] \qquad \frac{\mathrm{upd} = [x{\mapsto}E_1\langle 1\rangle,\ y{\mapsto}E_2\langle 2\rangle]}{\vdash \{\mathcal{S}[\mathrm{upd}] \mid \psi[\mathrm{upd}]\}\ x \leftarrow E_1 \sim_\alpha y \leftarrow E_2\ \{\mathcal{S} \mid \psi\}}\ [\textsc{Assn}]$$

$$\frac{\mathrm{upd} = [\tau_1{\mapsto}\tau_1 \cdot E_1\langle 1\rangle,\ \tau_2{\mapsto}\tau_2 \cdot E_2\langle 2\rangle]}{\vdash \{\mathcal{S}[\mathrm{upd}] \mid \psi[\mathrm{upd}]\}\ \texttt{release}\ E_1 \sim_\alpha \texttt{release}\ E_2\ \{\mathcal{S} \mid \psi\}}\ [\textsc{Release}] \qquad \frac{\vdash \{\mathcal{R} \mid \phi\}\ c_1 \sim_\alpha c_2\ \{\mathcal{T} \mid \chi\} \qquad \vdash \{\mathcal{T} \mid \chi\}\ c_1' \sim_\alpha c_2'\ \{\mathcal{S} \mid \psi\}}{\vdash \{\mathcal{R} \mid \phi\}\ c_1; c_1' \sim_\alpha c_2; c_2'\ \{\mathcal{S} \mid \psi\}}\ [\textsc{Seq}]$$

$$\frac{\mathrm{rst} = [x{\mapsto}d_x,\ y{\mapsto}d_y] \qquad \vdash \{\mathcal{R} \mid \phi\}\ c_1 \sim_\alpha c_2\ \{\mathcal{S}[\mathrm{rst}] \mid \psi[\mathrm{rst}]\}}{\vdash \{\mathcal{R}[\mathrm{rst}] \mid \phi[\mathrm{rst}]\}\ \texttt{local}\ x\ \texttt{in}\ c_1 \sim_\alpha \texttt{local}\ y\ \texttt{in}\ c_2\ \{\mathcal{S} \mid \psi\}}\ [\textsc{Local}]$$

$$\frac{\mathcal{R} \models P_1\langle 1\rangle = P_2\langle 2\rangle \qquad \vdash \{\mathcal{R} \wedge P_1\langle 1\rangle \mid \phi\}\ c_1 \sim_\alpha c_2\ \{\mathcal{S} \mid \psi\} \qquad \vdash \{\mathcal{R} \wedge \neg P_1\langle 1\rangle \mid \phi\}\ c_1' \sim_\alpha c_2'\ \{\mathcal{S} \mid \psi\}}{\vdash \{\mathcal{R} \mid \phi\}\ \texttt{if}\ P_1\ \texttt{then}\ c_1\ \texttt{else}\ c_1'\ \texttt{fi} \sim_\alpha \texttt{if}\ P_2\ \texttt{then}\ c_2\ \texttt{else}\ c_2'\ \texttt{fi}\ \{\mathcal{S} \mid \psi\}}\ [\textsc{If}]$$

$$\frac{\begin{array}{c}\forall k.\ \mathcal{R}_k \models P_1\langle 1\rangle = P_2\langle 2\rangle \qquad \forall k.\ \mathcal{R}_k \wedge \neg P_1\langle 1\rangle \models \mathcal{S} \wedge \psi \le \phi_k \\ \forall k.\ \vdash \{\mathcal{R}_k \wedge P_1\langle 1\rangle \wedge P_2\langle 2\rangle \mid \phi_k\}\ c_1 \sim_\alpha c_2\ \{\mathcal{R}_{k+1} \mid \phi_{k+1}\}\end{array}}{\vdash \{\mathcal{R}_0 \mid \phi_0\}\ \texttt{while}\ P_1\ \texttt{do}\ c_1\ \texttt{od} \sim_\alpha \texttt{while}\ P_2\ \texttt{do}\ c_2\ \texttt{od}\ \{\mathcal{S} \wedge \neg P_1\langle 1\rangle \wedge \neg P_2\langle 2\rangle \mid \psi\}}\ [\textsc{While}]$$

$$\frac{\models \{\mathcal{R} \mid \phi\}\ x \xleftarrow{\$} S_1 \sim_\alpha y \xleftarrow{\$} S_2\ \{\mathcal{S} \mid \psi\}}{\vdash \{\mathcal{R} \mid \phi\}\ x \xleftarrow{\$} S_1 \sim_\alpha y \xleftarrow{\$} S_2\ \{\mathcal{S} \mid \psi\}}\ [\textsc{Samp-Lift}]$$

Logical rules

$$\frac{\mathcal{R} \models \mathcal{R}_0 \quad \mathcal{R} \models \phi_0 \le \phi \quad \mathcal{S}_0 \models \mathcal{S} \quad \mathcal{S}_0 \models \psi \le \psi_0 \quad \vdash \{\mathcal{R}_0 \mid \phi_0\}\ c_1 \sim_\alpha c_2\ \{\mathcal{S}_0 \mid \psi_0\}}{\vdash \{\mathcal{R} \mid \phi\}\ c_1 \sim_\alpha c_2\ \{\mathcal{S} \mid \psi\}}\ [\textsc{Conseq}] \qquad \frac{0 < c < \infty \quad \vdash \{\mathcal{R} \mid \phi\}\ c_1 \sim_\alpha c_2\ \{\mathcal{S} \mid \psi\}}{\vdash \{\mathcal{R} \mid c \cdot \phi\}\ c_1 \sim_\alpha c_2\ \{\mathcal{S} \mid c \cdot \psi\}}\ [\textsc{Scale}]$$

$$\frac{\mathcal{R} \models \mathcal{P}_1\langle 1\rangle \wedge \mathcal{P}_2\langle 2\rangle \quad \vdash c_1 \equiv_{\mathcal{P}_1} c_1' \quad \vdash c_2 \equiv_{\mathcal{P}_2} c_2' \quad \vdash \{\mathcal{R} \mid \phi\}\ c_1' \sim_\alpha c_2'\ \{\mathcal{S} \mid \psi\}}{\vdash \{\mathcal{R} \mid \phi\}\ c_1 \sim_\alpha c_2\ \{\mathcal{S} \mid \psi\}}\ [\textsc{Equiv}]$$

Derived rules

$$\frac{\begin{array}{c}\forall s = ((m_1,\tau_1),(m_2,\tau_2)) \in \mathcal{R}.\ \mathcal{S}[x{\mapsto}v, y{\mapsto}v - G(s)](s) \text{ for } (\llbracket S_2 \rrbracket(m_2) + G(s))\text{-almost every } v \\ \forall s = ((m_1,\tau_1),(m_2,\tau_2)) \in \mathcal{R}.\ \mathrm{H}_\alpha^{\psi_s}(\llbracket S_1 \rrbracket(m_1) \parallel \llbracket S_2 \rrbracket(m_2) + G(s)) \le \phi(s) \quad \text{where } \psi_s(v) \triangleq \psi[x{\mapsto}v, y{\mapsto}v - G(s)](s)\end{array}}{\vdash \{\mathcal{R} \mid \phi\}\ x \xleftarrow{\$} S_1 \sim_\alpha y \xleftarrow{\$} S_2\ \{\mathcal{S} \mid \psi\}}\ [\textsc{Samp-Gap}]$$

$$\frac{\begin{array}{c}\forall s = ((m_1,\tau_1),(m_2,\tau_2)) \in \mathcal{R}.\ \mathcal{S}[x{\mapsto}v, y{\mapsto}v](s) \text{ for } \llbracket S_2 \rrbracket(m_2)\text{-almost every } v \\ \forall s = ((m_1,\tau_1),(m_2,\tau_2)) \in \mathcal{R}.\ \mathrm{H}_\alpha^{\psi_s}(\llbracket S_1 \rrbracket(m_1) \parallel \llbracket S_2 \rrbracket(m_2)) \le \phi(s) \quad \text{where } \psi_s(v) \triangleq \psi[x{\mapsto}v, y{\mapsto}v](s)\end{array}}{\vdash \{\mathcal{R} \mid \phi\}\ x \xleftarrow{\$} S_1 \sim_\alpha y \xleftarrow{\$} S_2\ \{\mathcal{S} \mid \psi\}}\ [\textsc{Samp}]$$

$$\frac{\mathcal{R}_n \models \neg P_1\langle 1\rangle \wedge \neg P_2\langle 2\rangle \quad \forall k < n.\ \mathcal{R}_k \models P_1\langle 1\rangle \wedge P_2\langle 2\rangle \quad \forall k < n.\ 0 < \phi_k < \infty \quad \forall k < n.\ \vdash \{\mathcal{R}_k \mid \phi_k\}\ c_1 \sim_\alpha c_2\ \{\mathcal{R}_{k+1} \mid 1\}}{\vdash \{\mathcal{R}_0 \mid \prod_{k<n} \phi_k\}\ \texttt{while}\ P_1\ \texttt{do}\ c_1\ \texttt{od} \sim_\alpha \texttt{while}\ P_2\ \texttt{do}\ c_2\ \texttt{od}\ \{\mathcal{R}_n \mid 1\}}\ [\textsc{While-Prod}]$$

Fig. 3. Rules of RRHL: core rules (blue) and derived rules (green). ∎

Structural rules. Rule [Skip] leaves conditions and budget unchanged. Rule [Assn] is the relational assignment rule; as in Hoare logic, the post-condition and the post-budget are evaluated under the substitution of the two assigned expressions. Rule [Release] behaves similarly to [Assn], but substitutes the transcript instead of a program variable. Rule [Seq] composes two judgments as in standard relational Hoare logics, through a shared intermediate assertion and a shared intermediate budget. On the quantitative judgments, it embodies Theorem 3.3, so composing with [Seq] reproduces the moments accountant's accounting exactly. Rule [Local] reflects the initialization of local variables at the level of the logic; it can be understood as chaining the derivation for the body with two applications of the assignment rule on the initialization and reset statements. By combining this rule with [Equiv], RRHL can substitute the body with a single sampling statement, to obtain state-of-the-art privacy bounds for subsampling (§5.1.1). Rule [If] performs a synchronous analysis of the conditional statements: the premise $\mathcal{R} \models P_1\langle 1\rangle = P_2\langle 2\rangle$ forces

the two guards to agree. The two judgments in its premises relate the two branches at the same post-budget and post-condition, after internalizing the value of the guard in the pre-condition. Rule [While] is the synchronous loop rule of relational Hoare logic, with invariant and budget taken as *families* indexed by the number $k \in \mathbb{N}$ of completed iterations, so that the budget can change along the iterations. Its logical premises require (i) that the guards evaluate to the same Boolean value, to enforce synchronous execution; and (ii) that at every index at which the loops may exit — that is, under $\mathcal{R}_k \wedge \neg P_1^{\langle 1 \rangle}$ — the Boolean post-condition holds and the post-budget is below ϕ_k. The premise on the loop bodies ensures that each iteration transforms the invariant $\mathcal{R}_k$ into $\mathcal{R}_{k+1}$ and the budget ϕ_k into ϕ_{k+1}. Notably, the rule does not require the loops to terminate and does not bound the number of iterations. Soundness of [While] is proved by bounding every finite unrolling of the loops and passing to the limit by monotone convergence (Lemma A.8 and Theorem A.9). Rule [Samp-Lift] makes any valid judgment on sampling statements derivable. It is the only core rule whose premise needs probabilistic reasoning, done in the meta-logic. Our case studies only use sampling rules derived from it, whose premises are facts about the sampled distributions, following the convention of similar relational Hoare logics [15, 19].

One-sided rules. The structural rules above are synchronous, as they relate two structurally identical statements. One-sided rules, instead, act on one program at a time, leaving the other unchanged. RRHL supports one-sided rules similar to those of other relational probabilistic program logics [8, 19]. For instance, the one-sided *left* rule for assignment is the following one:

$$\text{[Assn-L]}\quad \frac{\mathsf{upd} = [x \mapsto E_1\langle 1 \rangle]}{\vdash \{\mathcal{S}[\mathsf{upd}] \mid \psi[\mathsf{upd}]\}\ x \leftarrow E_1 \sim_\alpha \mathtt{skip}\ \{\mathcal{S} \mid \psi\}}$$

It is analogous to [Assn], except that it only updates the variable assigned on the left. Similar rules can be defined for all the other statements of our programming language. Since none of our case studies relies on them, they are deferred to Appendix B.

Logical rules. Unlike structural rules, which are driven by the syntax of the program, logical rules act on the judgment by modifying its pre- and post-conditions, or the programs it involves. Rule [Conseq] allows for the monotonic replacement of the logical conditions and budgets. Rule [Scale] multiplies pre-budget and post-budget by a constant. Rule [Equiv] replaces each program by one that is equivalent on the states that satisfy the pre-condition. Two of its premises are equivalences $\vdash c \equiv_{\mathcal{P}} c'$ (c and c' agree on inputs satisfying $\mathcal{P}$), derived in the calculus of Appendix C. We use this rule to derive state-of-the-art RDP bounds for amplification by subsampling (§5.1.1) and to express shifted composition proofs (§5.1.3).

Derived rules. The core rules follow the conventional presentation of program logics but can be tedious to use in practice; e.g., for sampling statements, rules with simpler syntactic premises are often more convenient. Figure 3 lists such *derived rules*, which follow from the core ones. Rule [Samp-Gap] sets the two sampled values at distance $G(s)$ from each other. Its privacy cost is the Hellinger integral between the left sampling distribution and the shifted right one, where $[\![S_2]\!](m_2) + G(s)$ is the pushforward of $[\![S_2]\!](m_2)$ along $v \mapsto v + G(s)$. Its first premise asks the post-condition to hold for almost every value in $[\![S_2]\!](m_2) + G(s)$. Under these conditions, the rule moves the sampled values close to each other in the logical post-condition, and we use this technique to perform shifted composition proofs (§5.1.3). Rule [Samp] equates the two sampled values at the cost of the Hellinger integral between the two distributions; its first premise is read as for [Samp-Gap], at $G(s) = 0$. The rule is pervasive in the privacy analyses of §5. For instance, the

following judgment

$$\vdash \{\mathcal{R} \mid \exp\big(\alpha(\alpha-1)\|g^{\langle 1\rangle} - g^{\langle 2\rangle}\|^2/2v\big)\}\ w \xleftarrow{\$} \mathcal{N}(g,v) \sim_\alpha w \xleftarrow{\$} \mathcal{N}(g,v)\ \{\mathcal{R} \wedge w^{\langle 1\rangle} = w^{\langle 2\rangle} \mid 1\}$$

equates the values sampled from two Gaussian distributions in the post-condition at privacy cost given by the distance of their means:

$$\mathrm{H}_\alpha\Big(\mathcal{N}(g^{\langle 1\rangle}, v) \Big\| \mathcal{N}(g^{\langle 2\rangle}, v)\Big) \;=\; \exp\Big(\alpha(\alpha-1)\,\|g^{\langle 1\rangle} - g^{\langle 2\rangle}\|^2/2v\Big)\,. \tag{1}$$

Finally, Rule [While-Prod] is derived from [While], using the family of state-independent budgets $\prod_{k\le j<n}\phi_j$, which express the pre-budget after k iterations, and from Rule [Scale], which normalizes the post-budget to 1.

4.4 Metatheoretical Properties

The main metatheoretical property of RRHL is *soundness*, stating validity of all derivable judgments.

Theorem 4.3 (Soundness). *For every judgment $\{\mathcal{R} \mid \phi\}\ c_1 \sim_\alpha c_2\ \{\mathcal{S} \mid \psi\}$, we have:*

$$\vdash \{\mathcal{R} \mid \phi\}\ c_1 \sim_\alpha c_2\ \{\mathcal{S} \mid \psi\} \qquad \Rightarrow \qquad \vDash \{\mathcal{R} \mid \phi\}\ c_1 \sim_\alpha c_2\ \{\mathcal{S} \mid \psi\}.$$

Most importantly, RRHL proves Rényi differential privacy of programs:

Proposition 4.4 (From judgments to Rényi differential privacy). *Let c be a program, $\mathcal{R}$ a state relation, and ϕ a budget. Assume*

$$\vDash \{\mathcal{R} \wedge \tau^{\langle 1\rangle} = \tau^{\langle 2\rangle} \mid \phi\}\ c \sim_\alpha c\ \{\tau^{\langle 1\rangle} = \tau^{\langle 2\rangle} \mid 1\}\,.$$

- *If ϕ is a constant, the transcript of c is $\big(\alpha, \frac{\log\phi}{\alpha-1}\big)$-RDP on $\mathcal{R}$-related inputs.*
- *If $\mathcal{R}$ and ϕ are families $(\simeq_k)_k$ and $(\phi_k)_k$, and the judgment holds for every k, the transcript of c is $\big(\alpha, \big(\frac{\log\phi_k}{\alpha-1}\big)_k\big)$-individually RDP for the family $(\simeq_k)_k$.*

5 Examples

In this section, we use RRHL to derive Rényi differential privacy bounds for three classes of training algorithms, exercising the privacy enhancement and analysis techniques mentioned in §1. In §5.1 we continue the analysis of our motivating example, DP-SGD, under the *transcript* and the *last-iterate* models, exercising sharp budget composition (as in the moments accountant), amplification by subsampling and shifted composition. We obtain the bounds reported by DP-SGD implementations. In §5.2 we consider a variant of gradient descent with per-record Rényi filters [38], which replaces worst-case accounting by per-record accounting: the budget of each individual is tracked in the program state and decreased whenever the individual takes part in a round, and the guarantee obtained is individual RDP. In §5.3 we consider DP-FTRL [32, 49], a learning algorithm designed for federated learning. Its privacy comes from correlated noising, which RRHL accounts for by charging the budget at each sampling statement. A variant with subsampling is treated in Appendix E.5.

5.1 DP-SGD

In this section, we analyse the privacy guarantees of DP-SGD under two distinct adversary models. The *transcript* model [1, 58, 59] captures what happens in online training, where updates travel through the network, so at each training step the noisy gradient is released to the adversary and the privacy budget is linear in the number of steps. In the *last-iterate* model, only the final model is released, and the privacy loss stops growing after a finite number of steps [6, 37]. These case studies are entirely machine-checked, and to our knowledge they are the first formal proofs of the amplified (ϵ, δ) bounds that RDP-based accountants of DP-SGD implementations report. To the

best of our knowledge, the only other mechanization of DP-SGD is by Sato [63], cut only covers the transcript model and proves a looser bound (§7).

5.1.1 Transcript Model. Under the transcript model, we derive that the DP-SGD algorithm of Figure 1 is $\left(\alpha,\ T\,\frac{\log \Upsilon_\alpha(q,\Gamma,v)}{\alpha-1}\right)$-RDP, where $\Upsilon_\alpha(q,\Gamma,v)$ is a shorthand for the Hellinger integral:

$$\Upsilon_\alpha(q,\Gamma,v) \triangleq \mathrm{H}_\alpha((1-q)\,\mathcal{N}(0,v) + q\,\mathcal{N}(\Gamma,v) \,\|\, \mathcal{N}(0,v)) \ . \tag{2}$$

This is the privacy bound reported by deployments such as Opacus [77], TensorFlow Privacy [70] and FedML [46]. The deployments evaluate Υ via the methods of Mironov et al. [59] (which we also formalized, see Proposition E.2) and convert it to (ϵ,δ)-differential privacy coefficients with the identities of Proposition 3.5. Every step of this pipeline is machine-checked, except for the numerical evaluation of the actual bound Υ. Appendix F provides an overview of these deployments and of the accountant libraries implementing the same pipeline [9, 23, 41, 46, 52].

We first establish *individual* RDP, which lets the proof fix the index j of the differing record; Lemma 3.7 then derives global RDP from this bound. By Proposition 4.4, it suffices to derive:

$$\vdash \{D^{\langle 1\rangle} \simeq_j D^{\langle 2\rangle} \wedge \tau^{\langle 1\rangle} = \tau^{\langle 2\rangle} \mid \Upsilon_\alpha(q,\Gamma,v)^T\}\ \mathrm{dpsgd} \sim_\alpha \mathrm{dpsgd}\ \{\theta^{\langle 1\rangle} = \theta^{\langle 2\rangle} \wedge \tau^{\langle 1\rangle} = \tau^{\langle 2\rangle} \mid 1\} \tag{3}$$

Here dpsgd is the program of Figure 1; the pre-condition asks that the initial datasets differ at an arbitrary index j, as defined by the adjacency $D \simeq_j D' \triangleq (\forall i \neq j.\ D[i] = D'[i]) \wedge (D[j] = \bot \vee D'[j] = \bot)$. The pre-budget is the T-th power of $\Upsilon_\alpha(q,\Gamma,v)$, which is the budget of a learning round. The complete derivation is in Appendix E.1; here we only give an high-level overview.

In deriving the judgment, the crucial step is to replace the body of the `local` construct, via [Equiv] and [Local], with the equivalent single sampling statement

$$\mathrm{smp} \triangleq w \xleftarrow{\$} \int \left[\, (1-q)\,\mathcal{N}(g(B),\, v) \ +\ q\,\mathcal{N}(g(B)+u,\, v) \,\right] d\mathrm{Btc}_q(B)\,, \tag{4}$$

where $u \triangleq \mathrm{clip}_\Gamma(\nabla\ell(\theta, D[j]))$ is the contribution of the differing element j, B ranges over the subsets of the indices other than the differing one j, under their law Btc_q, in which each index enters independently with probability q, and $g(B) \triangleq \sum_{i\in B} \mathrm{clip}_\Gamma(\nabla\ell(\theta, D[i]))$ is the gradient sum on batch B. The sampling expression is the average over all batches B sampled from Btc_q of two cases: if the j-th element is not included (with probability $1-q$), due to $\nabla\ell(\theta,\bot) = 0$, the noise has mean equal to the sum of all the other elements of the batch: $g(B)$. Otherwise (with probability q) the j-th element is sampled, so its contribution u is added to the sum, and the noise has mean $g(B)+u$. This substitution enables deriving state-of-the-art privacy bounds for subsampling, as described in §5.1.2 below. We then apply Rule [Samp], obtaining:

$$\vdash \{\mathcal{I}_k \mid \Upsilon_\alpha(q,\Gamma,v)\}\ \mathrm{smp} \sim_\alpha \mathrm{smp}\ \{\mathcal{I}_k \wedge w^{\langle 1\rangle} = w^{\langle 2\rangle} \mid 1\} \tag{5}$$

where the invariant $\mathcal{I}_k$ asserts that t evaluates to k, that the datasets are adjacent at j, and that the models and the transcripts are identical. Here, the pre-budget is obtained by case analysis: when $D^{\langle 2\rangle}[j] = \bot$, we have:

$$\mathrm{H}_\alpha\Big(\int (1-q)\,\mathcal{N}(g(B),\, v) + q\,\mathcal{N}(g(B)+u,\, v)\ d\mathrm{Btc}_q(B) \,\Big\|\, \int \mathcal{N}(g(B),\, v)\ d\mathrm{Btc}_q(B)\Big) \ \leq\ \Upsilon_\alpha(q,\Gamma,v)\,,$$

which holds by standard properties on Gaussian distributions (Proposition A.24 ⚠). When $D^{\langle 1\rangle}[j] = \bot$, we reason identically, and we conclude using:

$$\Lambda_\alpha(q,\Gamma,v) \triangleq \mathrm{H}_\alpha(\mathcal{N}(0,v) \,\|\, (1-q)\,\mathcal{N}(0,v) + q\,\mathcal{N}(\Gamma,v)) \ \leq\ \Upsilon_\alpha(q,\Gamma,v)\,,$$

due to Mironov et al. [59, Corollary 7], i.e., our Lemma E.1 ⚠. Intuitively, in (5) the noise counteracts the drift of the models due to the differing record: it equates the noised values in the logical post-condition *despite their different means*, at cost $\Upsilon_\alpha(q,\Gamma,v)$. The rest of the derivation is by

syntax-driven application of the rules. Given (3), Proposition 4.4 and Lemma 3.7 entail that DP-SGD is $\left(\alpha,\ T \log \Upsilon_\alpha(q,\Gamma,v)/\alpha-1\right)$-RDP in the transcript model for the standard adjacency $\simeq \triangleq \bigcup_j \simeq_j$.

5.1.2 Subsampling Bound Comparison. To derive the pre-budget $\Upsilon_\alpha(q,\Gamma,v)$ at each round, our derivation replaces the subsampling loop and the noising by a single sampling. Without it, the batch and the noise are drawn by distinct sampling statements, and Rule [Seq] composes them through a shared intermediate assertion and a shared intermediate budget. By using [Samp] on the noising instruction, we can impose identity of the models at a batch-dependent cost: 1 if $j \notin B$ and $\exp(\alpha(\alpha-1)\Gamma^2/2v)$ if $j \in B$, by (1). By [Seq], this budget has to be used as post-condition of the sub-sampling loop, where by applying [Samp] on all sampling instructions, we derive the pre-budget $(1-q) + q\exp\left(\alpha(\alpha-1)\Gamma^2/2v\right) \geq \Upsilon_\alpha(q,\Gamma,v)$. This cost is the average of the per-batch costs over the coin draws, whereas $\Upsilon_\alpha(q,\Gamma,v)$ is the cost of the averaged distribution, which is smaller due to joint convexity of the Hellinger integral. Merging the two draws into the single sampling statement smp removes the need for an intermediate budget and the averaging. Figure 4 compares the two bounds after composition over T rounds and conversion to (ϵ,δ)-differential privacy. The upper curve is the unamplified Gaussian mechanism, which ignores subsampling; the one in the middle is the bound of this section, which coincides with the one by Sato [63, Lemma 7.2] on the same algorithm, under a replace-one adjacency relation. Numerical accountants that compose privacy loss distributions directly [42] give tighter (ϵ,δ) guarantees than any RDP analysis followed by conversion; verifying them is outside the scope of this work.

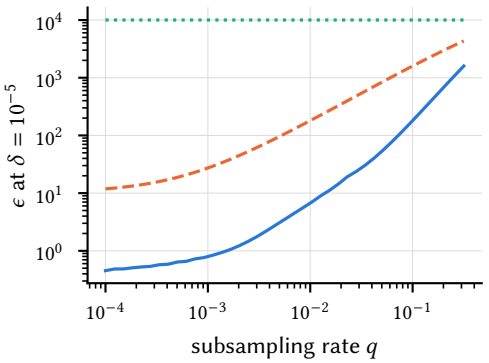


Fig. 4. Bounds of §5.1.1 (——) and §5.1.2 (– –), and the unamplified DP-SGD (i.e., the bounds of §5.1.1 when $q = 1$) (····), over $T = 10^4$ rounds and converted by Proposition 3.5 at $\delta = 10^{-5}$, with noise variance $v = \Gamma^2$.

5.1.3 Last-Iterate Model. We now turn to the last-iterate model. For uniformity, we reuse the same algorithm as in §5.1.1, without the release at line 8, and we call it lastIter. It differs from Altschuler and Talwar [6]'s algorithm, for the use of Poisson subsampling instead of fixed-size minibatch subsampling see Appendix E.2.

Comparison with the transcript model. Unlike the transcript analysis, which ignores the projection[1], this proofs needs needs models to stay within distance Δ to obtain a bound independent of the number of rounds. The proof mimics the shifted composition technique [5, 37] for an arbitrary number of rounds $R \leq T$. It proceeds in two phases: first, the two models drift apart for the initial $T-R$ rounds, maintaining a distance of at most Δ via the projection Π at no privacy cost. Second, during the final R rounds, the models are brought closer by a fixed step—while tracking their distance in the logical invariant—until they coincide, so that Proposition 4.4 applies. The second phase starts from the distance left by the first one, which is at most Δ by the projection.

In the last R rounds, the Gaussian noise thus has two effects that the logic has to account for: counteracting the drift of the models due to the differing record, as in §5.1.1, and reducing their worst-case distance. To charge each effect to its own rule, we express the variance as $v = \sigma_1^2 + \sigma_2^2$ for

[1]There the invariant keeps the two models equal, so Π is applied to equal arguments and plays no part in the analysis; the algorithm of Abadi et al. [1], and its Rényi analysis by Mironov et al. [59], have no projection step at all.

two suitable $\sigma_1, \sigma_2 > 0$, and we replace the sampling statement at line 7 by two via Rule [Equiv]:

$$\texttt{local } z \texttt{ in } w \xleftarrow{\$} \mathcal{N}(g,\, \sigma_1^2 + \sigma_2^2) \equiv_{\top} \texttt{local } z \texttt{ in } (z \xleftarrow{\$} \mathcal{N}(g,\, \sigma_2^2);\, w \xleftarrow{\$} \mathcal{N}(z,\, \sigma_1^2)) \quad (6)$$

Reducing the worst case distance of the models also requires non-expansiveness of the update and of the projection: for every batch B, every dataset D and all θ_1, θ_2,

$$\big\|(\theta_1 - \gamma\, g(B, D, \theta_1)) - (\theta_2 - \gamma\, g(B, D, \theta_2))\big\| \;\le\; \|\theta_1 - \theta_2\| \quad \text{and} \quad \|\Pi(\theta_1) - \Pi(\theta_2)\| \;\le\; \|\theta_1 - \theta_2\|\,, \quad (7)$$

where $g(B, D, \theta) \triangleq \sum_{i \in B} \mathrm{clip}_{\Gamma}(\nabla \ell(\theta, D[i]))$ is the clipped gradient sum of line 6 and $\gamma \triangleq \eta/q|D|$ is the step applied to it at line 8. The first condition holds for $\eta \le 2q/\beta$ under the hypotheses of [6]: convex, β-smooth, Γ-Lipschitz losses (proved in Lean); the second condition is assumed.

Derivation. Leveraging Proposition 4.4, we derive *individual* Rényi differential privacy of DP-SGD under the last-iterate model by deriving the following family of judgments, one per $1 \le R \le T$:

$$\vdash \{D\langle 1\rangle \simeq_j D\langle 2\rangle \wedge \tau\langle 1\rangle = \tau\langle 2\rangle \mid \Omega_\alpha(R)\}\ \mathsf{lastIter} \sim_\alpha \mathsf{lastIter}\ \{\theta\langle 1\rangle = \theta\langle 2\rangle \wedge \tau\langle 1\rangle = \tau\langle 2\rangle \mid 1\}\,. \quad (8)$$

Here, the adjacency relation is the one of §5.1.1, and

$$\Omega_\alpha(R) \triangleq \left(\Upsilon_\alpha(q, \Gamma, \sigma_2^2) \cdot \exp\!\left(\frac{\alpha(\alpha-1)\,\Delta^2}{2R^2\gamma^2\sigma_1^2}\right)\right)^R \quad (9)$$

is the cost of the last R rounds, since the first $T - R$ rounds incur no privacy cost. The left (red) factor is the cost of adding Gaussian noise with variance σ_2^2 to the Poisson-subsampled minibatch. It is the budget of one application of [Samp] to the first (red) statement in (6), after merging it with the subsampling loop as in §5.1.1. The right (blue) factor is the budget of one application of [Samp-Gap] to the second (blue) statement. The logical invariant keeps track of the distance between the two models, which is at most Δ when the last R rounds begin. At each of these rounds, we set the gap so that the noises compensate a shift of Δ/R in the models, which the update scales by γ. The budget is thus the Hellinger integral (1) of two Gaussians of variance σ_1^2 whose means are $\Delta/R\gamma$ apart. After R rounds the distance is 0, the two models coincide, and Proposition 4.4 applies. The complete derivation is in Appendix E.2. Lemma 3.7 lifts the individual guarantee to *global* Rényi differential privacy: for every real $\alpha > 1$, lastIter is $\big(\alpha,\ \min_{R \le T} \log \Omega_\alpha(R)/\alpha - 1\big)$-RDP under the last-iterate model. Moreover, $\frac{\log \Omega_\alpha(R)}{\alpha - 1} = RL + \frac{K}{R}$, where $L \triangleq \log \Upsilon_\alpha(q,\Gamma,\sigma_2^2)/\alpha - 1$ and $K \triangleq \alpha\,\Delta^2/2\gamma^2\sigma_1^2$ (cf. [6, Equations 3.4 and 3.5]). By a routine computation, the real minimizer is $R = \sqrt{K/L}$, and rounding it up loses at most L. Therefore, if $L > 0$ and $T \ge \lceil\sqrt{K/L}\rceil$, we have $\min_{1 \le R \le T}(RL + K/R) \;\le\; 2\sqrt{KL} + L$ (cf. [6, Equation 3.6]), i.e., the bound does not depend on T. From our derivation, a bound of the form of Altschuler and Talwar [6, Theorem 3.1] follows as a corollary, under additional assumptions due to the different subsampling mechanism (Appendix E.2).

5.2 Noisy Gradient Descent with Per-Record Rényi Filters

Standard privacy accounting techniques, like those of §5.1, provide *worst-case* bounds on the privacy budget. This form of accounting is too conservative for *most* records: a record whose gradient is small barely affects the final model. *Rényi filters* [38] are a technique for obtaining per-record guarantees instead: they associate a per-individual privacy budget with each record, spend a part of it at every round the record participates in, and *filter* out the record from the learning step when its budget is exhausted. We verify a variant of the Noisy Gradient Descent algorithm with Rényi filters of Feldman and Zrnic [38, Algorithm 3], and obtain the bound of their Proposition 5.1.

The filtered DP-GD algorithm. The algorithm is in Figure 5. Its inputs are the dataset D, a collection of elements from $\mathbb{R}^d_\perp$, as in §5.1; the number of rounds T; the step size $\eta \in \mathbb{R}_{>0}$; the norm $\Gamma \in \mathbb{R}_{>0}$ to which gradients are clipped; the per-individual budget $B \in \mathbb{R}_{\ge 0}$, and

the noise variance $v \in \mathbb{R}_{>0}$. Line 2 initializes the round counter t, the model θ, and an array s which records the budget consumed by each record. Each of the T rounds runs an inner loop over D that accumulates the clipped gradients in acc and their squared norms in s. The statement at line 6 computes the gradient of the i-th entry, and clips it at $r_i(s) \triangleq \min\{\Gamma, \sqrt{\max(0, B - s[i])}\}$. Here, minimizing against Γ guarantees the boundedness of the gradient, and minimizing against $\sqrt{\max(0, B - s[i])}$ ensures that an element contributes 0 once it has exhausted the budget B, filtering it out from the training set. The statement at line 10 adds noise to the gradient, and line 11 performs the descent step and releases the model. Compared to [38, Algorithm 3], ours adds Gaussian noise once to the sum of the gradients, rather than adding the same nois to each of the $|D|$ gradients. We believe this is the intended reading of their Algorithm 3, which they base on DP-GD of Abadi et al. [1], where noise goes to the aggregate gradient.

```
1:  input dataset D; parameters T, η, Γ, B, v
2:  s ← 0; θ ← 0; t ← 0
3:  while t < T do
4:      i ← 0; acc ← 0
5:      while i < |D| do
6:          acc ← acc + clip_{r_i(s)}(∇ℓ(θ, D[i]))
7:          s[i] ← s[i] + ‖clip_{r_i(s)}(∇ℓ(θ, D[i]))‖²
8:          i ← i + 1
9:      od
10:     z ←$ N(0, v);
11:     θ ← θ − η/|D| (acc + z); release θ; t ← t + 1
12: od
```

Fig. 5. Noisy gradient descent with Rényi filters.

Derivation. We now show that the algorithm of Figure 5 is $(\alpha, {}^{\alpha B}/_{2v})$-individually RDP for every element of the dataset [38, Proposition 5.1], by deriving the following judgment

$$\vdash \{D^{\langle 1\rangle} \simeq_j D^{\langle 2\rangle} \wedge \tau^{\langle 1\rangle} = \tau^{\langle 2\rangle} \mid \exp\big((\alpha-1)\tfrac{\alpha B}{2v}\big)\} \text{ filter } \sim_\alpha \text{ filter } \{\theta^{\langle 1\rangle} = \theta^{\langle 2\rangle} \wedge \tau^{\langle 1\rangle} = \tau^{\langle 2\rangle} \mid 1\}\,, \quad (10)$$

where $j < |D|$ is the (meta-level) index of an arbitrary element of D. The key step is deriving the [While] premise:

$$\vdash \{\mathcal{R}_j \wedge t^{\langle 1\rangle} = t^{\langle 2\rangle} = k \mid \omega\} \text{ round } \sim_\alpha \text{ round } \{\mathcal{R}_j \wedge t^{\langle 1\rangle} = t^{\langle 2\rangle} = k+1 \mid \omega\} \quad (11)$$

for every $k < T$, where round is the body of the main loop of Figure 5. In the judgment, the invariant

$$\begin{aligned} \mathcal{R}_j \triangleq\ & D^{\langle 1\rangle} \simeq_j D^{\langle 2\rangle} \wedge \theta^{\langle 1\rangle} = \theta^{\langle 2\rangle} \wedge \tau^{\langle 1\rangle} = \tau^{\langle 2\rangle} \\ & \wedge (\forall l \neq j.\ s^{\langle 1\rangle}[l] = s^{\langle 2\rangle}[l]) \wedge 0 \le s^{\langle 1\rangle}[j] + s^{\langle 2\rangle}[j] \le B \wedge \textstyle\bigwedge_{i\in\{1,2\}} (D^{\langle i\rangle}[j] = \perp \Rightarrow s^{\langle i\rangle}[j] = 0) \end{aligned} \quad (12)$$

asserts adjacency of the datasets, identity of the models and transcripts, identity of the arrays $s^{\langle 1\rangle}$ and $s^{\langle 2\rangle}$ at positions $l \neq j$, and boundedness of $s[j]$. Finally, it requires that the consumed budget $s[j]$ of the run without the record j is 0. The quantitative invariant $\omega = \exp\big({}^{(\alpha-1)\alpha\,(B - s^{\langle 1\rangle}[j] - s^{\langle 2\rangle}[j])}/_{2v}\big)$ corresponds to the privacy budget that the record j has left in the run i where it is included in the dataset. Before any of its budget is spent, we have $s[j]^{\langle i\rangle} = 0$ in both runs, and the invariant is equal to the pre-budget of (10); it decreases to 1 when the budget is exhausted, that is, when $s[j]^{\langle i\rangle} = B$, matching the judgment's post-budget. Intuitively, ω is built up by applying [Samp-Gap] at each learning round, which multiplies the budget by the Hellinger integral (1) of two Gaussians with means at distance $\text{clip}_{r_j(s^{\langle i\rangle})}(\nabla\ell(\theta, D[j]))$, where i is the index of the run that holds the differing record j. As usual, the distance is the contribution of the differing record to the round's gradient sum. Note that the invariant ω is expressible in RRHL because budgets are functions of the states, not constants as in previous works [18, 20, 21, 62]. The derivation is in Appendix E.3. By Proposition 4.4, the algorithm is $(\alpha, {}^{\alpha B}/_{2v})$-individually RDP.

5.3 DP-FTRL

Differentially Private Follow-The-Regularized-Leader (DP-FTRL) is a family of algorithms designed for federated learning systems, where each round trains on the data available at that time, so

the server cannot subsample from the whole dataset, and amplification by subsampling is unavailable [49]. To cope with this, DP-FTRL keeps a *history* of past gradient sums, reused in later rounds, and adds noise to linear combinations of those sums rather than to every model, which yields convenient privacy/utility trade-offs [49]. DP-FTRL is used to train word-prediction models deployed in Gboard [76], and in this section we derive the RDP bounds of Choquette-Choo et al. [28] for a variant of DP-FTRL with *multiple participation* (meaning that data can participate in multiple rounds), which generalizes those studied in prior work [32, 49]. We also formalized a variant of the Amplified Banded Matrix Factorization algorithm of Choquette-Choo et al. [27], whose analysis is deferred to Appendix E.5.

The DP-FTRL algorithm. In Figure 6 we show an implementation of DP-FTRL under *multiple participation*, in which each record may take part in several rounds [28]. Its inputs are: the dataset D of elements of $\mathbb{R}^d_\perp$; a factorization $A = BC$ of the lower-triangular matrix of ones A; the number of rounds T and of aggregates N; the clipping norm $\Gamma \in \mathbb{R}_{>0}$; the noise parameter $\sigma \in \mathbb{R}_{>0}$; the regularization parameter $\lambda \in \mathbb{R}_{>0}$; and the participation schedule part. DP-FTRL maintains a *history* $H \in \mathbb{R}^{T\times d}$, whose row $H[t]$ (written at line 4) holds the sum of the gradients at round t, each clipped at norm Γ and weighted by the participation $\mathsf{part}(i,t) \in \mathbb{R}$ of its record at that round. Given the history of gradients, the model of round t is $-1/\lambda\,(A \cdot H)_t$, where $(A \cdot H)_t = \sum_{s\le t} H[s]$. Rather than computing $A \cdot H$ directly, DP-FTRL factorizes $A = BC$ into an *encoder* $C \in \mathbb{R}^{N\times T}$ and a *decoder* $B \in \mathbb{R}^{T\times N}$. The factorization gives freedom in how noise is spread across rounds. Leveraging the identity $(A \cdot H)_t = (BC \cdot H)_t$, the algorithm first computes an array $Y[r] \triangleq (C \cdot H)_r + z$ whose entries (which we call *aggregates*) are linear combinations of gradient sums perturbed with fresh Gaussian noise z of variance σ^2. Aggregates are computed in the inner loop. Its body (line 8) generates the r-th aggregate at the round $\mathsf{done}(r) \triangleq \sup\{s : C_{r,s} \neq 0\}$, which is the round wqhen the last gradient sum it reads is computed. Finally, at lines 11–12, the decoder recombines the aggregates into the model $-1/\lambda(B \cdot Y)_t$, which is then released.

```
1:  input dataset D; factorization A = BC; participation
    part; parameters T, N, λ, Γ, σ
2:  θ ← 0; H ← 0; Y ← 0; t ← 0
3:  while t < T do
4:      H[t] ← Σ_{i<|D|} part(i, t) clip_Γ(∇ℓ(θ, D[i]))
5:      r ← 0
6:      while r < N do
7:          if done(r) = t then
8:              z ←$ N(0, σ²); Y[r] ← (C · H)_r + z
9:          r ← r + 1
10:     od
11:     ŝ ← (B · Y)_t
12:     θ ← −(1/λ) ŝ; release θ; t ← t + 1
13: od
```

Fig. 6. DP-FTRL with a matrix mechanism.

Derivation. We want to prove individual RDP of DP-FTRL, and its privacy bound rests on the following observation. On adjacent datasets, the histories of two runs differ only by the contributions of the record j, of norm at most $\Gamma_{j,t} \triangleq |\mathsf{part}(j,t)|\,\Gamma$ in the round t. So the difference $H^{\langle 2\rangle} - H^{\langle 1\rangle}$ lies in the set $\mathcal{U}_j$ of the $U \in \mathbb{R}^{T\times d}$ with $\|U_t\| \le \Gamma_{j,t}$ for all t, and the largest squared distance between the noiseless aggregates $C \cdot H$ of the two runs is $\mathsf{sens}_j \triangleq \sup_{U\in\mathcal{U}_j} \sum_r \|(C \cdot U)_r\|^2$. This is the drift caused by the differing record that the Gaussian noise has to mask, and, as for DP-SGD in §5.1.1, the RDP level is linear in it. Guided by this observation, we prove that the algorithm of Figure 6 is $\big(\alpha,\ \alpha\,\mathsf{sens}_j/2\sigma^2\big)$-individually RDP for every record j by deriving the judgment

$$\vdash \{D^{\langle 1\rangle} \simeq_j D^{\langle 2\rangle} \wedge \tau^{\langle 1\rangle} = \tau^{\langle 2\rangle} \mid \exp\big(\alpha(\alpha-1)\,\mathsf{sens}_j/2\sigma^2\big)\}\ \mathsf{dpftrl} \sim_\alpha \mathsf{dpftrl}\ \{\tau^{\langle 1\rangle} = \tau^{\langle 2\rangle} \mid 1\}\,, \qquad (13)$$

where $\simeq_j$ is as in §5.1.1. The key step is deriving the [While] premise

$$\vdash \{\mathcal{R}_j \wedge t^{\langle 1\rangle} = t^{\langle 2\rangle} = k \mid \Phi_k\}\ \mathsf{round} \sim_\alpha \mathsf{round}\ \{\mathcal{R}_j \wedge t^{\langle 1\rangle} = t^{\langle 2\rangle} = k+1 \mid \Phi_{k+1}\} \qquad (14)$$

for every $k < T$, where round is the body of the main loop of Figure 6. The invariant $\mathcal{R}_j$ asserts adjacency of the datasets, equality of the models, aggregates and transcripts, and $H^{\langle 2\rangle} - H^{\langle 1\rangle} \in \mathcal{U}_j$. The quantitative invariant refines the supremum above along the run. At the start of the round k, the slots $t < k$ are written, so the history difference ranges over $\mathcal{U}_j^k \triangleq \{U \in \mathcal{U}_j : U_t = H^{\langle 2\rangle}[t] - H^{\langle 1\rangle}[t] \text{ for } t < k\}$, and only the aggregates with $\mathrm{done}(r) \geq k$ remain to be released. The budget is the worst-case cost of these releases:

$$\Phi_k \triangleq \sup_{U \in \mathcal{U}_j^k} \prod_{r:\mathrm{done}(r)\geq k} \exp\big(\alpha(\alpha-1)\|(C \cdot U)_r\|^2/2\sigma^2\big).$$

When the aggregate r is computed, $\Delta_r = (C \cdot (H^{\langle 2\rangle} - H^{\langle 1\rangle}))_r$ is the distance that [Samp-Gap] has to close on line 8 to equate it along the two runs, at the usual cost $\exp(\alpha(\alpha-1)\|\Delta_r\|^2/2\sigma^2)$. At each round the budget is $\prod_{r:\mathrm{done}(r)=k} \exp(\alpha(\alpha-1)\|\Delta_r\|^2/2\sigma^2)$, i.e. the product of the contributions of all aggregates. Preservation of the invariant follows by: $\prod_{r:\mathrm{done}(r)=k} \exp(\alpha(\alpha-1)\|\Delta_r\|^2/2\sigma^2) \cdot \Phi_{k+1} \leq \Phi_k$. The rest of the derivation is in Appendix E.4, which also derives the bounds in [28, 32, 49].

6 The Formalization

Overview and availability. The theory and case studies of this work are formalized in Lean 4 (v4.33.1) [31] over Mathlib [54], covering our language and semantics (§4.1), logic (Sections 4.2 to 4.4), and case studies (§5). The formalization is publicly available at https://github.com/davidedavoli/RRHL.

Effort. The development comprises over 23k lines of Lean, docstrings excluded, as outlined in Table 1. For the case studies, the table separates four kinds of lines: the RRHL derivations (including programs, assertions, budgets, and proof trees); the derivations of the equivalence premises of [Equiv]; side conditions of the rules: entailments between assertions, inequalities between budgets, the Hellinger bounds for the sampling rules; and the rest, which mainly consists of proofs of privacy bounds from the derivations. Case studies required the largest proof effort, mostly in RRHL derivations, and equivalence proofs. Deriving privacy bounds from our derivation typically requires little effort, except for DP-FTRL without subsampling, where several published bounds [27, 28, 32, 49], are obtained from the same derivation. Besides case studies, general results of measure-theory for the logic's soundness, duality and deriving side conditions are the largest row, which amounts at about 5k lines of code.

Contributions of independent interest. The formalization also contains results of independent interest: (i) results on divergences, Hellinger integrals, and closed forms of Gaussian error functions not provided by Mathlib; (ii) our probabilistic while language with continuous sampling and its equivalence calculus, which can serve as a foundation for other relational program logics.

AI Use. The proofs of the artifact were generated by large language models (mainly Claude Opus 5.5 and Fable 5.1). To ensure that the formalization captures the claims of the paper, the authors manually inspected the definitions and the statements of the main theorems during the development and before submission. The absence of additional axioms and of `sorry` statements in proofs is checked at compilation time: the artifact lists, for each main theorem, the axioms it relies on, which are only those of Lean's library: `propext`, `Classical.choice` and `Quot.sound`. To facilitate the inspection of the artifact, we use *deep embeddings*. In particular, derivability notions are defined as inductive types whose constructors are the corresponding rules. Hence checking that a case study is not proven semantically side-stepping the logic, amounts to inspecting that the case study derives a term of that inductive type, however the proofs were produced.

Part	RRHL	Equivalences	Side conditions	Other	Total
Language and semantics (§4.1)	–	–	–	627	627
Measures and divergences (§3.2, App. A)	–	–	–	4,715	4,715
Program equivalences (App. C)	–	–	–	1,722	1,722
Logic (§4.2–4.4)	–	–	–	1,421	1,421
Witness duality (§4.2, App. D)	–	–	–	3,026	3,026
Case studies (§5)	5,005	1,532	3,292	2,376	12,205
DP-SGD, transcript model (§5.1.1)	199	–	57	85	341
DP-SGD, last-iterate model (§5.1.3)	943	–	93	705	1,741
DP-SGD, shared by both models (§5.1)	515	385	1,203	234	2,337
Rényi filters (§5.2)	762	–	389	123	1,274
DP-FTRL without subsampling (§5.3)	733	–	233	602	1,568
DP-FTRL with subsampling (App. E.5)	1,804	1,147	1,206	332	4,489
Shared by all case studies (§5)	49	–	111	295	455
Total	**5,005**	**1,532**	**3,292**	**13,887**	**23,716**

Table 1. Proof effort in lines of Lean, docstrings excluded.

7 Related Work

Program Logics for Differential Privacy. Relational program logics are the closest to our work. They prove privacy by constructing approximate couplings of two executions. The first such logic, apRHL [20], was mechanized in Rocq. The logic was later refined, for instance by Barthe et al. [18]. Approxis [45] brings approximate couplings to higher-order programs within the Iris-based family of probabilistic separation logics [4, 13, 43, 48, 69], and Clutch-DP [44] uses them to verify differential privacy of higher-order programs with local state. Avanzini et al. [8] give a relational logic whose pre- and post-conditions are, like our budgets, state-dependent quantities, but for discrete distributions. Most of these logics establish (ϵ, δ)-differential privacy, whose composition theorems are loose, unlike the exact composition of Rényi divergences. Only two logics go beyond (ϵ, δ)-differential privacy. The logic of Barthe and Olmedo [21] bounds f-divergences between discrete distributions. The logic of Sato et al. [64] covers Rényi, zero-concentrated and truncated concentrated differential privacy over continuous distributions; it composes constant budgets and is not mechanized. None of these logics supports amplification by subsampling or shifted composition.

There are alternative approaches to the formal verification of differential privacy. Type systems track the sensitivity of programs and compose privacy budgets along their typing derivations, in linear and dependent types [39, 60] and in higher-order relational refinement types [17]. Automated verifiers work by pairing each noise sample drawn on one database with a shifted sample on its neighbor, so that both runs produce the same output, and tracking the privacy budget of each shift. The shifts are either annotated by the user and checked by a type system [75, 78], or inferred [74]. A third approach is semantic. Sato [63] formalizes Rényi differential privacy in Isabelle/HOL, with no language or program logic.

On the Choice of Couplings. Relational logics for differential privacy interpret a judgment by an approximate lifting of its postcondition to output distributions. The definitions in the literature differ in the number of witness distributions that they require, and we call them 0-, 1- and 2-couplings accordingly. A 2-coupling consists of two witness distributions over pairs of states related by the postcondition, one for each program, whose divergence is bounded by the budget. It is used by Barthe and Olmedo [21] for discrete distributions and by Sato et al. [64] for continuous

distributions and a family of divergences that includes the Rényi divergence. The latter require that the witnesses depend measurably on the initial memories. A 1-coupling consists of a single witness distribution whose marginals approximate the two output distributions. It is used in apRHL [20]. A 0-coupling is witness-free and bounds the divergence by quantifying over tests. It is used by Sato [62] for (ϵ, δ)-differential privacy over continuous distributions and by Sato and Katsumata [65] for arbitrary divergences. Several results relate these different styles of couplings. For (ϵ, δ)-differential privacy over discrete distributions, Barthe et al. [16] prove a duality theorem based on an approximate form of Strassen's theorem. They show that 0-couplings and 1-couplings coincide with their $\star$-liftings. Sato and Katsumata [65] extend one direction of this duality to arbitrary divergences in the discrete setting. Proposition 4.2 and Theorem D.1 give both directions between 0-couplings and 2-couplings for the Rényi divergence and form, to our knowledge, the first duality theorem stated in the continuous setting. The witness-free side of our duality is the codensity lifting of Sato and Katsumata [65] instantiated with tests into the one-point space.

Analyses of DP-SGD and Its Variants. The privacy analysis of DP-SGD has been refined several times. The first analyses made a single pass over disjoint batches [68], accounted for a privacy budget per parameter [66], or composed per-step (ϵ, δ) guarantees with advanced composition [22, 35]. The moments accountant [1] and Rényi differential privacy [58] replaced them with an exact composition of Rényi divergences, combined with amplification by subsampling [11, 59, 73] and with an improved conversion to (ϵ, δ) guarantees [12, 25]. Numerical accountants [42] compose the privacy loss distributions directly and obtain tighter bounds compared to RDP accounting techniques. For the last iterate of convex problems, amplification by iteration [37] gives bounds that improve with the number of steps taken after a record is used. Its refinements [6, 29] give bounds that do not grow with the number of iterations, and Altschuler and Chewi [5] abstract the technique into the shifted composition rule. The RDP bounds of Rényi filters [38] and of DP-FTRL [27, 28, 32, 49], reproduced in Sections 5.2 and 5.3 and Appendix E.5, had not been formalized before this work, to our knowledge. Sato [63] formalizes DP-SGD in Isabelle/HOL under Hamming-distance adjacency, with a weaker bound than ours.

Implementations: Challenges and Opportunities. An important direction for future work is to bring our guarantees to implementations written in mainstream languages such as Rust.

On the challenge side, deployed systems may leak privacy through implementation specifics, including floating-point arithmetic and timing. Mironov [57] shows that the textbook implementation of the Laplace mechanism in floating-point arithmetic is not differentially private. Casacuberta et al. [26] show that several differentially private libraries underestimate the sensitivity of summations, because floating-point addition is not associative and can overflow, and that the resulting releases violate the claimed guarantees. On the opportunity side, several works provide a starting point to verify implementations. SampCert [30] provides verified samplers for the discrete Laplace and Gaussian distributions. Refining our algorithms with verified samplers in the style of SampCert would be a first step towards verified implementations. Alerus [10] is a framework for verifying probabilistic Rust programs. It has been used to prove samplers of differentially private algorithms correct, but not their privacy. Combining the two would yield privacy guarantees for Rust implementations, with Alerus proving that an implementation refines its core-language model and RRHL proving that the model is private.

8 Conclusion and Future Work

We introduced RRHL, the first mechanized quantitative relational Hoare logic for proving Rényi differential privacy. Its quantitative judgments bound the Hellinger integral between pairs of output distributions, and its proof system embodies the composition theorem of the Hellinger integral,

which is sharp, to support compositional privacy analyses. RRHL is formalized in Lean, and we use it to verify Rényi differential privacy guarantees of private learning algorithms (including DP-SGD and DP-FTRL). The guarantees are formalized in an idealized language, and the actual numerical evaluation of the bounds is not formalized. Our next step is to bring our guarantees to implementations written in a real-world language. Another direction is to prove utility guarantees for the same algorithms, such as bounds on the error of the prefix sums released by DP-FTRL and on the excess risk of DP-SGD for convex losses.

References

[1] Martín Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep Learning with Differential Privacy. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 308–318.

[2] John M. Abowd. 2018. The U.S. Census Bureau Adopts Differential Privacy. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*. 2867.

[3] Alejandro Aguirre, Gilles Barthe, Justin Hsu, Benjamin Lucien Kaminski, Joost-Pieter Katoen, and Christoph Matheja. 2021. A Pre-Expectation Calculus for Probabilistic Sensitivity. *Proc. ACM Program. Lang.* 5, POPL (2021), 52:1–52:28. doi:10.1145/3434333

[4] Alejandro Aguirre, Philipp G. Haselwarter, Kwing Hei Li, Simon Oddershede Gregersen, Joseph Tassarotti, and Lars Birkedal. 2024. Error Credits: Resourceful Reasoning about Error Bounds for Higher-Order Probabilistic Programs. *Proc. ACM Program. Lang.* 8, ICFP (2024). doi:10.1145/3674635

[5] Jason M. Altschuler and Sinho Chewi. 2023. Shifted Composition I: Harnack and Reverse Transport Inequalities. *CoRR* abs/2311.14520 (2023). https://arxiv.org/abs/2311.14520

[6] Jason M. Altschuler and Kunal Talwar. 2022. Privacy of Noisy Stochastic Gradient Descent: More Iterations without More Privacy Loss. In *Advances in Neural Information Processing Systems 35 (NeurIPS)*.

[7] Apple Differential Privacy Team. 2017. Learning with Privacy at Scale. *Apple Machine Learning Journal* 1, 8 (2017).

[8] Martin Avanzini, Gilles Barthe, Davide Davoli, and Benjamin Grégoire. 2025. A Quantitative Probabilistic Relational Hoare Logic. *Proc. ACM Program. Lang.* 9, POPL (2025), 1167–1195.

[9] AWS Labs. 2025. `fast-differential-privacy`. Software library. https://github.com/awslabs/fast-differential-privacy

[10] Alexander Y. Bai and Joseph Tassarotti. 2026. Verifying Probabilistic Programs in Rust. arXiv:2607.12282 [cs.PL] https://arxiv.org/abs/2607.12282

[11] Borja Balle, Gilles Barthe, and Marco Gaboardi. 2018. Privacy Amplification by Subsampling: Tight Analyses via Couplings and Divergences. In *Advances in Neural Information Processing Systems 31 (NeurIPS)*. 6280–6290.

[12] Borja Balle, Gilles Barthe, Marco Gaboardi, Justin Hsu, and Tetsuya Sato. 2020. Hypothesis Testing Interpretations and Rényi Differential Privacy. In *International Conference on Artificial Intelligence and Statistics (AISTATS) (Proceedings of Machine Learning Research, Vol. 108)*. 2496–2506. Theorem 21; arXiv:1905.09982.

[13] Jialu Bao, Emanuele D'Osualdo, and Azadeh Farzan. 2025. Bluebell: An Alliance of Relational Lifting and Independence for Probabilistic Reasoning. *Proc. ACM Program. Lang.* 9, POPL (2025), 58:1–58:31. doi:10.1145/3704894

[14] Gilles Barthe, Thomas Espitau, Marco Gaboardi, Benjamin Grégoire, Justin Hsu, and Pierre-Yves Strub. 2018. An Assertion-Based Program Logic for Probabilistic Programs. In *Programming Languages and Systems (ESOP) (Lecture Notes in Computer Science, Vol. 10801)*. Springer, Cham, 117–144. doi:10.1007/978-3-319-89884-1_5

[15] Gilles Barthe, Thomas Espitau, Benjamin Grégoire, Justin Hsu, and Pierre-Yves Strub. 2018. Proving Expected Sensitivity of Probabilistic Programs. *Proc. ACM Program. Lang.* 2, POPL (2018), 57:1–57:29. doi:10.1145/3158145

[16] Gilles Barthe, Thomas Espitau, Justin Hsu, Tetsuya Sato, and Pierre-Yves Strub. 2019. Relational ⋆-Liftings for Differential Privacy. *Logical Methods in Computer Science* 15, 4 (2019), 18:1–18:32. doi:10.23638/LMCS-15(4:18)2019

[17] Gilles Barthe, Marco Gaboardi, Emilio Jesús Gallego Arias, Justin Hsu, Aaron Roth, and Pierre-Yves Strub. 2015. Higher-Order Approximate Relational Refinement Types for Mechanism Design and Differential Privacy. In *POPL*.

[18] Gilles Barthe, Marco Gaboardi, Benjamin Grégoire, Justin Hsu, and Pierre-Yves Strub. 2016. Proving Differential Privacy via Probabilistic Couplings. In *LICS*.

[19] Gilles Barthe, Benjamin Grégoire, and Santiago Zanella-Béguelin. 2009. Formal certification of code-based cryptographic proofs. In *POPL 2009*. ACM, 90–101.

[20] Gilles Barthe, Boris Köpf, Federico Olmedo, and Santiago Zanella-Béguelin. 2012. Probabilistic relational reasoning for differential privacy. In *POPL 2012*. ACM, 97–110.

[21] Gilles Barthe and Federico Olmedo. 2013. Beyond Differential Privacy: Composition Theorems and Relational Logic for f-divergences between Probabilistic Programs. In *Automata, Languages, and Programming – ICALP 2013, Part II*

(LNCS, Vol. 7966). Springer, 49–60.
[22] Raef Bassily, Adam Smith, and Abhradeep Thakurta. 2014. Private Empirical Risk Minimization: Efficient Algorithms and Tight Error Bounds. In *55th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 464–473.
[23] Daniel J. Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Javier Fernandez-Marques, Yan Gao, Lorenzo Sani, Kwing Hei Li, Titouan Parcollet, Pedro Porto Buarque de Gusmão, and Nicholas D. Lane. 2020. Flower: A Friendly Federated Learning Research Framework. *CoRR* abs/2007.14390 (2020).
[24] Patrick Billingsley. 1995. *Probability and Measure* (3rd ed.). Wiley.
[25] Clément L. Canonne, Gautam Kamath, and Thomas Steinke. 2020. The Discrete Gaussian for Differential Privacy. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*.
[26] Sílvia Casacuberta, Michael Shoemate, Salil Vadhan, and Connor Wagaman. 2022. Widespread Underestimation of Sensitivity in Differentially Private Libraries and How to Fix It. In *CCS*. 471–484.
[27] Christopher A. Choquette-Choo, Arun Ganesh, Ryan McKenna, H. Brendan McMahan, J. Keith Rush, Abhradeep Guha Thakurta, and Xinyi Zhang. 2023. (Amplified) Banded Matrix Factorization: A unified approach to private training. In *Advances in Neural Information Processing Systems 36 (NeurIPS)*.
[28] Christopher A. Choquette-Choo, H. Brendan McMahan, J. Keith Rush, and Abhradeep Guha Thakurta. 2023. Multi-Epoch Matrix Factorization Mechanisms for Private Machine Learning. In *ICML*. 5924–5963.
[29] Rishav Chourasia, Jiayuan Ye, and Reza Shokri. 2021. Differential Privacy Dynamics of Langevin Diffusion and Noisy Gradient Descent. In *Advances in Neural Information Processing Systems 34 (NeurIPS 2021)*. 14771–14781.
[30] Markus de Medeiros, Muhammad Naveed, Tancrède Lepoint, Temesghen Kahsai, Tristan Ravitch, Stefan Zetzsche, Anjali Joshi, Joseph Tassarotti, Aws Albarghouthi, and Jean-Baptiste Tristan. 2025. Verified Foundations for Differential Privacy. *Proc. ACM Program. Lang.* 9, PLDI (2025). doi:10.1145/3729294
[31] Leonardo de Moura and Sebastian Ullrich. 2021. The Lean 4 Theorem Prover and Programming Language. In *Automated Deduction – CADE 28 (LNCS, Vol. 12699)*. Springer, 625–635.
[32] Sergey Denisov, H. Brendan McMahan, Keith Rush, Adam Smith, and Abhradeep Guha Thakurta. 2022. Improved Differential Privacy for SGD via Optimal Private Linear Operators on Adaptive Streams. In *Advances in Neural Information Processing Systems 35 (NeurIPS)*.
[33] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating Noise to Sensitivity in Private Data Analysis. In *Theory of Cryptography (TCC) (LNCS, Vol. 3876)*. Springer, 265–284.
[34] Cynthia Dwork and Aaron Roth. 2014. The Algorithmic Foundations of Differential Privacy. *Foundations and Trends in Theoretical Computer Science* 9, 3–4 (2014), 211–407.
[35] Cynthia Dwork, Guy N. Rothblum, and Salil Vadhan. 2010. Boosting and Differential Privacy. In *51st IEEE Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 51–60.
[36] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security (CCS)*. 1054–1067.
[37] Vitaly Feldman, Ilya Mironov, Kunal Talwar, and Abhradeep Thakurta. 2018. Privacy Amplification by Iteration. In *59th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 521–532.
[38] Vitaly Feldman and Tijana Zrnic. 2021. Individual Privacy Accounting via a Rényi Filter. In *NeurIPS*.
[39] Marco Gaboardi, Andreas Haeberlen, Justin Hsu, Arjun Narayan, and Benjamin C. Pierce. 2013. Linear Dependent Types for Differential Privacy. In *POPL*.
[40] Michèle Giry. 1982. A Categorical Approach to Probability Theory. In *Categorical Aspects of Topology and Analysis*. Lecture Notes in Mathematics, Vol. 915. Springer, 68–85. doi:10.1007/BFb0092872
[41] Google Differential Privacy contributors. 2025. `dp_accounting`: Differential Privacy Accounting Library. Software library. https://github.com/google/differential-privacy/tree/main/python/dp_accounting
[42] Sivakanth Gopi, Yin Tat Lee, and Lukas Wutschitz. 2021. Numerical Composition of Differential Privacy. In *Advances in Neural Information Processing Systems 34 (NeurIPS)*. 11631–11642.
[43] Simon Oddershede Gregersen, Alejandro Aguirre, Philipp G. Haselwarter, Joseph Tassarotti, and Lars Birkedal. 2024. Asynchronous Probabilistic Couplings in Higher-Order Separation Logic. *Proc. ACM Program. Lang.* 8, POPL (2024), 26:1–26:32.
[44] Philipp G. Haselwarter, Alejandro Aguirre, Simon Oddershede Gregersen, Kwing Hei Li, Joseph Tassarotti, and Lars Birkedal. 2026. Modular Verification of Differential Privacy in Probabilistic Higher-Order Separation Logic. *Proc. ACM Program. Lang.* 10, PLDI (2026). doi:10.1145/3808311
[45] Philipp G. Haselwarter, Kwing Hei Li, Alejandro Aguirre, Simon Oddershede Gregersen, Joseph Tassarotti, and Lars Birkedal. 2025. Approximate Relational Reasoning for Higher-Order Probabilistic Programs. *Proc. ACM Program. Lang.* 9, POPL (2025).
[46] Chaoyang He, Songze Li, Jinhyun So, Xiao Zeng, Mi Zhang, Hongyi Wang, Xiaoyang Wang, Praneeth Vepakomma, Abhishek Singh, Hang Qiu, Xinghua Zhu, Jianzong Wang, Li Shen, Peilin Zhao, Yan Kang, Yang Liu, Ramesh Raskar,

Qiang Yang, Murali Annavaram, and Salman Avestimehr. 2020. FedML: A Research Library and Benchmark for Federated Machine Learning. *CoRR* abs/2007.13518 (2020).

[47] JAX Privacy contributors. 2025. JAX Privacy: Algorithms for Privacy-Preserving Machine Learning in JAX. Software library. https://github.com/google-deepmind/jax_privacy

[48] Ralf Jung, Robbert Krebbers, Jacques-Henri Jourdan, Aleš Bizjak, Lars Birkedal, and Derek Dreyer. 2018. Iris from the ground up: A modular foundation for higher-order concurrent separation logic. *Journal of Functional Programming* 28 (2018), e20.

[49] Peter Kairouz, Brendan McMahan, Shuang Song, Om Thakkar, Abhradeep Thakurta, and Zheng Xu. 2021. Practical and Private (Deep) Learning Without Sampling or Shuffling. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*.

[50] Olav Kallenberg. 1997. *Foundations of Modern Probability*. Springer.

[51] Dexter Kozen. 1985. A Probabilistic PDL. *J. Comput. Syst. Sci.* 30, 2 (1985), 162–178. doi:10.1016/0022-0000(85)90012-1

[52] Xuechen Li, Florian Tramèr, Percy Liang, and Tatsunori Hashimoto. 2022. Large Language Models Can Be Strong Differentially Private Learners. In *10th International Conference on Learning Representations (ICLR)*.

[53] Friedrich Liese and Igor Vajda. 1987. *Convex Statistical Distances*. Teubner-Texte zur Mathematik, Vol. 95. Teubner, Leipzig.

[54] The mathlib Community. 2020. The Lean mathematical library. In *CPP 2020*. ACM, 367–381.

[55] Annabelle McIver and Carroll Morgan. 2005. *Abstraction, Refinement and Proof for Probabilistic Systems*. Springer, New York. doi:10.1007/b138392

[56] H. Brendan McMahan, Zheng Xu, and Yanxiang Zhang. 2024. A Hassle-free Algorithm for Private Learning in Practice: Don't Use Tree Aggregation, Use BLTs. In *EMNLP*.

[57] Ilya Mironov. 2012. On Significance of the Least Significant Bits for Differential Privacy. In *CCS*.

[58] Ilya Mironov. 2017. Rényi Differential Privacy. In *30th IEEE Computer Security Foundations Symposium (CSF)*. IEEE, 263–275.

[59] Ilya Mironov, Kunal Talwar, and Li Zhang. 2019. Rényi Differential Privacy of the Sampled Gaussian Mechanism. *CoRR* abs/1908.10530 (2019). https://arxiv.org/abs/1908.10530

[60] Joseph P. Near, David Darais, Chike Abuah, Tim Stevens, et al. 2019. Duet: An Expressive Higher-Order Language and Linear Type System for Statically Enforcing Differential Privacy. *Proc. ACM Program. Lang.* 3, OOPSLA (2019).

[61] Alfréd Rényi. 1961. On Measures of Entropy and Information. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1*. University of California Press, 547–561.

[62] Tetsuya Sato. 2016. Approximate Relational Hoare Logic for Continuous Random Samplings. *Electronic Notes in Theoretical Computer Science* 325 (2016), 277–298. Proceedings of MFPS XXXII.

[63] Tetsuya Sato. 2026. Formalization of Rényi Differential Privacy in Isabelle/HOL. *CoRR* abs/2610.25636 (2026). https://arxiv.org/abs/2610.25636

[64] Tetsuya Sato, Gilles Barthe, Marco Gaboardi, Justin Hsu, and Shin-ya Katsumata. 2019. Approximate Span Liftings: Compositional Semantics for Relaxations of Differential Privacy. In *34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. IEEE.

[65] Tetsuya Sato and Shin-ya Katsumata. 2023. Divergences on monads for relational program logics. *Mathematical Structures in Computer Science* 33, 4-5 (2023), 427–485. doi:10.1017/S0960129523000245

[66] Reza Shokri and Vitaly Shmatikov. 2015. Privacy-Preserving Deep Learning. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 1310–1321.

[67] Amer Sinha, Thomas Mesnard, Ryan McKenna, Daogao Liu, Christopher A. Choquette-Choo, Yangsibo Huang, Da Yu, George Kaissis, Zachary Charles, Ruibo Liu, Lynn Chua, Pritish Kamath, Pasin Manurangsi, Steve He, Chiyuan Zhang, Badih Ghazi, Borja De Balle Pigem, Prem Eruvbetine, Tris Warkentin, Armand Joulin, and Ravi Kumar. 2025. VaultGemma: A Differentially Private Gemma Model. *CoRR* abs/2510.15001 (2025).

[68] Shuang Song, Kamalika Chaudhuri, and Anand D. Sarwate. 2013. Stochastic Gradient Descent with Differentially Private Updates. In *IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. IEEE, 245–248.

[69] Joseph Tassarotti and Robert Harper. 2019. A Separation Logic for Concurrent Randomized Programs. *Proc. ACM Program. Lang.* 3, POPL (2019), 64:1–64:30.

[70] TensorFlow Privacy contributors. 2025. TensorFlow Privacy. Software library. https://github.com/tensorflow/privacy

[71] The OpenDP Project. 2025. OpenDP Library, version 0.14. Software library. https://opendp.org

[72] Tim van Erven and Peter Harremoës. 2014. Rényi Divergence and Kullback–Leibler Divergence. *IEEE Transactions on Information Theory* 60, 7 (2014), 3797–3820.

[73] Yu-Xiang Wang, Borja Balle, and Shiva Prasad Kasiviswanathan. 2019. Subsampled Rényi Differential Privacy and Analytical Moments Accountant. In *22nd International Conference on Artificial Intelligence and Statistics (AISTATS) (Proceedings of Machine Learning Research, Vol. 89)*. PMLR, 1226–1235.

[74] Yuxin Wang, Zeyu Ding, Daniel Kifer, and Danfeng Zhang. 2020. CheckDP: An Automated and Integrated Approach for Proving Differential Privacy or Finding Precise Counterexamples. In *CCS*.
[75] Yuxin Wang, Zeyu Ding, Guanhong Wang, Daniel Kifer, and Danfeng Zhang. 2019. Proving Differential Privacy with Shadow Execution. In *PLDI*.
[76] Zheng Xu, Yanxiang Zhang, Galen Andrew, Christopher A. Choquette-Choo, Peter Kairouz, H. Brendan McMahan, Jesse Rosenstock, and Yuanbo Zhang. 2023. Federated Learning of Gboard Language Models with Differential Privacy. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics: Industry Track (ACL)*. 629–639.
[77] Ashkan Yousefpour, Igor Shilov, Alexandre Sablayrolles, Davide Testuggine, Karthik Prasad, Mani Malek, John Nguyen, Sayan Ghosh, Akash Bharadwaj, Jessica Zhao, Graham Cormode, and Ilya Mironov. 2021. Opacus: User-Friendly Differential Privacy Library in PyTorch. *CoRR* abs/2109.12298 (2021).
[78] Danfeng Zhang and Daniel Kifer. 2017. LightDP: Towards Automating Differential Privacy Proofs. In *POPL*.

A Measure-Theoretic Background

This appendix introduces the notions and the notation of measure theory that are necessary to follow the technical development of the paper. The reader interested in a deeper and more complete introduction to measure and probability theory can consult Billingsley's "Probability and Measure" [24] or Kallenberg's "Foundations of Modern Probability" [50]; for the theory of f-divergences, of which the Hellinger integral is an instance, we refer to Liese and Vajda [53].

A.1 Measurable Spaces

σ-algebras are a central concept of measure theory, as they describe those collections of sets that can be measured:

Definition A.1 (σ-algebra, measurable space)**.** A *σ-algebra* on a set A is a collection $\Sigma \subseteq \mathcal{P}(A)$ such that:

- $A \in \Sigma$;
- if $X \in \Sigma$, then $A \setminus X \in \Sigma$;
- if $X_1, X_2, \ldots \in \Sigma$, then $\bigcup_{i \in \mathbb{N}} X_i \in \Sigma$.

A *measurable space* is a pair $\langle A, \Sigma \rangle$ of a set and a σ-algebra on it, whose members are called the *measurable sets*. We keep Σ implicit whenever it is clear from the context.

We say that a σ-algebra is generated by a collection of sets $\mathcal{G} \subseteq \mathcal{P}(A)$ whenever it is the smallest σ-algebra that contains all those sets.

Functions between measurable spaces are required to respect the σ-algebra.

Definition A.2 (Measurable function)**.** A function $f : A \to B$ between measurable spaces is *measurable* if and only if $f^{-1}(E)$ is a measurable set of A for every measurable set E of B.

Measurable spaces are closed under products and countable disjoint unions. Hence the space of states used to interpret our programs is a measurable space.

Definition A.3 (Product and disjoint union)**.** The *product* $\prod_{i \in I} B_i$ of a family of measurable spaces carries the σ-algebra generated by the sets $\pi_i^{-1}(E)$, for $i \in I$ and E measurable in B_i, where π_i is the i-th coordinate projection. The *disjoint union* $\bigsqcup_{i \in I} B_i$ of a countable family carries the σ-algebra of the sets X such that $\iota_i^{-1}(X)$ is measurable in B_i for every i, where ι_i is the i-th injection; this is a σ-algebra because each ι_i^{-1} commutes with complements and with countable unions.

Maps into a product are measurable as soon as their components are. As a consequence, the operations on memories and states of §4.1 are measurable without further assumptions.

Lemma A.4 (Maps into a product [50])**.** *A map $g : A \to \prod_{i \in I} B_i$ into a product of measurable spaces is measurable if and only if $\pi_i \circ g$ is measurable for every $i \in I$, where π_i is the i-th coordinate*

projection. Consequently, replacing one coordinate of a point of the product by a measurable function of that point is a measurable operation.

Proof. The sets $\pi_i^{-1}(E)$, for $i \in I$ and E measurable in B_i, generate the σ-algebra of the product, and a map is measurable as soon as the preimages of a generating family are measurable. The claim follows from $g^{-1}(\pi_i^{-1}(E)) = (\pi_i \circ g)^{-1}(E)$, the converse direction being the measurability of the projections themselves. □

Lemma A.5 (Maps out of a disjoint union)**.** *A map $f : \bigsqcup_{i\in I} B_i \to A$ out of a countable disjoint union of measurable spaces is measurable if and only if $f \circ \iota_i$ is measurable for every $i \in I$.*

Proof. For E measurable in A we have $\iota_i^{-1}(f^{-1}(E)) = (f \circ \iota_i)^{-1}(E)$, and by Definition A.3 the set $f^{-1}(E)$ is measurable exactly when all of these preimages are. □

A.2 Measures and Integrals

A measure assigns a mass to each measurable set, in a countably additive way.

Definition A.6 (Measure)**.** A *measure* on a measurable space A is a function μ from the measurable sets of A to $[0, \infty]$ such that:

- $\mu(\emptyset) = 0$;
- for every countable collection $\{X_i\}_{i\in\mathbb{N}}$ of pairwise disjoint measurable sets, $\mu\big(\bigcup_i X_i\big) = \sum_i \mu(X_i)$.

The measure μ is *finite* if $\mu(A) < \infty$, a *probability measure* if $\mu(A) = 1$ and a *sub-probability measure* if $\mu(A) \leq 1$. We write $\mathbf{0}$ for the zero measure and δ_a for the *Dirac measure* at a, which assigns 1 to the sets containing a and 0 to the others. We write $\mathrm{D}(A)$ for the set of sub-probability measures on A. Measures are added and rescaled setwise: $(\mu + \nu)(E) \triangleq \mu(E) + \nu(E)$ and $(c \cdot \mu)(E) \triangleq c\,\mu(E)$ for $c \in [0, \infty]$.

Measures provide the foundation for probability theory, as they make it possible to define central concepts such as expected values and, on top of the latter, the probability of events. The expected value of a measurable $f : A \to [0, \infty]$, in turn, is defined in terms of integration with respect to a measure.

Definition A.7 (Integral)**.** A *simple function* is a function of the form $s = \sum_{i=1}^n c_i \cdot \mathbf{1}_{E_i}$, where $c_1, \ldots, c_n \in [0, \infty)$, the sets $E_1, \ldots, E_n$ are measurable and pairwise disjoint, and $\mathbf{1}_E$ denotes the indicator function of E; its integral against μ is $\int s \,\mathrm{d}\mu \triangleq \sum_{i=1}^n c_i\,\mu(E_i)$. The *integral* of a measurable $f : A \to [0, \infty]$, also written $\mathbb{E}_\mu[f]$, is

$$\int f \,\mathrm{d}\mu \;\triangleq\; \sup\left\{ \int s \,\mathrm{d}\mu \;\middle|\; s \text{ simple},\ s \leq f \right\} \;\in\; [0, \infty]\,,$$

with the convention $0 \cdot \infty = 0$. When it helps to name the integration variable, we write $\int f(a)\,\mathrm{d}\mu(a)$ for $\int f \,\mathrm{d}\mu$. For a measurable $E \subseteq A$, the integral of f *over* E is $\int_E f \,\mathrm{d}\mu \triangleq \int f \cdot \mathbf{1}_E \,\mathrm{d}\mu$.

The integral is linear and monotone in both of its arguments.

One of the crucial properties needed to interpret probabilistic programs on top of measures is their continuity:

Lemma A.8 (Continuity of measure [50, Lemma 1.15])**.** *If $\mu_0 \leq \mu_1 \leq \cdots$ is an increasing sequence of measures on a measurable space, then its setwise supremum $E \mapsto \sup_n \mu_n(E)$ is again a measure.*

Continuity makes it possible to interpret iteration: the loop semantics is defined as the supremum of the increasing sequence of measures given by the loop unrollings. By Lemma A.8 this supremum is a measure, and it is a sub-probability measure, every unrolling having mass at most 1.

Our logic rests on a second property: a quantity integrated against the approximants lifts to the limit. The soundness of the loop rule relies on it, since the budget of the loop is obtained from the budgets of its finite unrollings. We write $f_n \uparrow f$ pointwise when $f_n(a) \le f_{n+1}(a)$ for every n and $f_n(a) \to f(a)$ for every a.

Theorem A.9 (Monotone convergence, Levi [50, Theorem 1.19])**.** *If $f_n \uparrow f$ pointwise, then $\int f_n \,\mathrm{d}\mu \uparrow \int f \,\mathrm{d}\mu$.*

Integration also runs backwards, turning a function into a measure. This is the construction the Gaussian measures of Appendix A.5 are defined by, and the one the densities of Appendix A.4 invert.

Definition A.10 (Measure with a density)**.** Let μ be a measure on A and let $f : A \to [0,\infty]$ be measurable. The map $E \mapsto \int_E f \,\mathrm{d}\mu$ is again a measure on A, written $f \cdot \mu$; countable additivity follows from Theorem A.9. It is a probability measure exactly when $\int f \,\mathrm{d}\mu = 1$.

Definition A.11 (Null set, almost everywhere, concentration)**.** A set $N \subseteq A$ is *μ-null* if $N \subseteq E$ for some measurable E with $\mu(E) = 0$. A property Q holds *μ-almost everywhere* (a.e.) if the set $\{a \in A \mid \neg Q(a)\}$ is μ-null. A measurable set S *carries* μ, equivalently μ is *concentrated* on S, if $\mu(A \setminus S) = 0$.

Measures can be transported along measurable functions, combined into products, and averaged into mixtures. These three constructions, together with the sub-Markov kernels of the next subsection, generate every measure that appears in the paper.

Definition A.12 (Pushforward)**.** The *pushforward* of μ under a measurable $f : A \to B$ is the measure $f^\sharp \mu \triangleq E \mapsto \mu(f^{-1}(E))$. It satisfies the change-of-variables formula $\int g \,\mathrm{d}(f^\sharp \mu) = \int g \circ f \,\mathrm{d}\mu$ [50, Lemma 1.22].

Definition A.13 (Product, marginals, convolution)**.** Finite measures μ on A and ν on B have a unique *product* $\mu \otimes \nu$ on $A \times B$ with $(\mu \otimes \nu)(E \times F) = \mu(E)\,\nu(F)$, and integration against it can be iterated in either order (Tonelli's theorem, [50, Theorem 1.27]). The *marginals* of a measure on a product are its pushforwards under the two projections. The *convolution* of two finite measures on $\mathbb{R}^d$ is the pushforward of their product under addition, $\mu * \nu \triangleq +^\sharp (\mu \otimes \nu)$ [50, Lemma 1.28].

Definition A.14 (Mixture)**.** Given a measurable space I, a probability measure P on it, and a family $(\mu_i)_{i \in I}$ of measures for which $i \mapsto \mu_i(E)$ is measurable for every measurable E, the *mixture* $\int \mu_i \,\mathrm{d}P(i)$ is the measure $E \mapsto \int \mu_i(E) \,\mathrm{d}P(i)$.

A.3 Sub-Markov Kernels

Probabilistic programs are modelled as functions mapping initial states to sub-probability measures over final states. In addition, the mass assigned to each fixed measurable set varies measurably with the input. Maps of this kind are called *sub-Markov kernels*, and this additional measurability condition makes them a monad suitable for defining probabilistic program semantics.

Definition A.15 (Sub-Markov kernel, Markov kernel)**.** A *sub-Markov kernel* from A to B is a map $M : A \to \mathrm{D}(B)$ such that, for every measurable set $E \subseteq B$, the mass $a \mapsto M(a)(E)$ is a measurable function of a. Its values are sub-probability measures, hence the name *sub-Markov*; it is a *Markov kernel* when every $M(a)$ is a probability measure.

Definition A.16 (Unit and bind)**.** The *unit* sends a value $a \in A$ to the Dirac measure $\delta_a \in \mathsf{D}(A)$ of Definition A.6. The *bind* of $\mu \in \mathsf{D}(A)$ with a sub-Markov kernel M from A to B is the measure

$$\big(\mathbf{dlet}\ a \leftarrow \mu\ \mathbf{in}\ M(a)\big)(E) \triangleq \int M(a)(E)\,\mathrm{d}\mu(a) \qquad \text{for } E \subseteq B \text{ measurable.}$$

The bind is well defined because of the measurability condition on sub-Markov kernels. The quantity being integrated is the function $a \mapsto M(a)(E)$, and by Definition A.7 only measurable functions can be integrated against μ.

The following proposition collects exactly the facts that make the semantics of §4.1 well defined and compositional.

Proposition A.17. *Let $\mu \in \mathsf{D}(A)$, let M be a sub-Markov kernel from A to B and let N be a sub-Markov kernel from B to C.*

(1) $\mathbf{dlet}\ a \leftarrow \mu\ \mathbf{in}\ M(a)$ *is again a sub-probability measure, so* $\mathbf{dlet}\ a \leftarrow \mu\ \mathbf{in}\ M(a) \in \mathsf{D}(B)$.
(2) *For every measurable $f : B \to [0,\infty]$, the map $a \mapsto \int f\,\mathrm{d}M(a)$ is measurable, and* $\int f\,\mathrm{d}\big(\mathbf{dlet}\ a \leftarrow \mu\ \mathbf{in}\ M(a)\big) = \int \big(\int f\,\mathrm{d}M(a)\big)\mathrm{d}\mu(a)$.
(3) $\mathbf{dlet}\ a \leftarrow \delta_{a_0}\ \mathbf{in}\ M(a) = M(a_0)$ *for every $a_0 \in A$.*
(4) $\mathbf{dlet}\ a \leftarrow \mu\ \mathbf{in}\ \delta_a = \mu$.
(5) $\mathbf{dlet}\ b \leftarrow \big(\mathbf{dlet}\ a \leftarrow \mu\ \mathbf{in}\ M(a)\big)\ \mathbf{in}\ N(b) = \mathbf{dlet}\ a \leftarrow \mu\ \mathbf{in}\ \big(\mathbf{dlet}\ b \leftarrow M(a)\ \mathbf{in}\ N(b)\big)$.
(6) *For every measurable $f : A \to B$,* $f^\sharp\,\mu = \mathbf{dlet}\ a \leftarrow \mu\ \mathbf{in}\ \delta_{f(a)}$.

Proof. Item 1 is [50, Lemma 1.38(iii)], the bound on the total mass being immediate. Item 2 is [50, Lemma 1.38(i)] for the measurability; the identity is an instance of [50, Theorem 1.27]. (These results are stated there for probability kernels; the sub-probability case is identical.) Items 3–5 are the monad laws for sub-probability measures [40], associativity being again the iterated-integral identity above. Item 6 is immediate from $\delta_{f(a)}(E) = \mathbf{1}_{f^{-1}(E)}(a)$, and is the kernel form of [50, Lemma 1.22]. □

A.4 Densities, f-Divergences and the Hellinger Integral

Two measures on the same space can be compared by asking whether one can be reconstructed from the other by reweighting. The Radon–Nikodym theorem asserts that, given two measures $\mu \in \mathsf{D}(A)$ and $\nu \in \mathsf{D}(A)$ on the same measurable space A, and provided ν dominates μ in the sense of the following definition, there exists a measurable function $\frac{\mathrm{d}\mu}{\mathrm{d}\nu} : A \to [0,\infty)$ which expresses, pointwise, how much ν has to be rescaled to get μ.

Definition A.18 (Absolute continuity)**.** μ is *absolutely continuous* with respect to ν, written $\mu \ll \nu$, if and only if every ν-null set is μ-null.

Theorem A.19 (Radon–Nikodym [50, Theorem A1.3])**.** *If μ and ν are finite measures with $\mu \ll \nu$, then there is a measurable $\frac{\mathrm{d}\mu}{\mathrm{d}\nu} : A \to [0,\infty)$, called the* density *of μ with respect to ν, such that $\mu(E) = \int_E \frac{\mathrm{d}\mu}{\mathrm{d}\nu}\,\mathrm{d}\nu$ for every measurable E — that is, $\mu = \frac{\mathrm{d}\mu}{\mathrm{d}\nu} \cdot \nu$ in the notation of Definition A.10. Any two such functions are equal ν-almost everywhere in the sense of Definition A.11, so the density is determined up to a ν-null set.*

Since integrals do not see changes on null sets, the choice of the density is irrelevant in everything that follows. Densities are at the base of the definition of f-divergences: a class of distances between probability measures, expressed, for a convex $f : [0,\infty) \to [0,\infty]$ and $\mu \ll \nu$, as

$$\mathrm{D}_f(\mu \,\|\, \nu) \triangleq \mathbb{E}_{a \leftarrow \nu}\big[\, f\big(\tfrac{\mathrm{d}\mu}{\mathrm{d}\nu}(a)\big)\big] = \int f\Big(\frac{\mathrm{d}\mu}{\mathrm{d}\nu}\Big)\,\mathrm{d}\nu\,,$$

and as ∞ when μ is not absolutely continuous with respect to ν. When $f(\cdot) \triangleq \cdot^{\alpha}$ with $\alpha > 1$, the display above is the quantity whose logarithm, rescaled by $\frac{1}{\alpha-1}$, is the Rényi divergence of order α:

$$\mathrm{R}_\alpha(\mu \,\|\, \nu) \;=\; \frac{1}{\alpha-1} \log \mathbb{E}_{a \leftarrow \nu}\big[\, \big(\tfrac{\mathrm{d}\mu}{\mathrm{d}\nu}(a)\big)^{\alpha} \big] \;=\; \frac{1}{\alpha-1} \log \int \Big(\frac{\mathrm{d}\mu}{\mathrm{d}\nu}\Big)^{\alpha} \mathrm{d}\nu \,.$$

In this formula, the term $\int (\frac{\mathrm{d}\mu}{\mathrm{d}\nu})^{\alpha} \,\mathrm{d}\nu$ is the *Hellinger integral of order α* between the two measures, written $\mathrm{H}_\alpha(\mu \,\|\, \nu)$, and it is the quantity our logic proves bounds on. It is the instance at weight $w = 1$ of the weighted Hellinger integral of Definition 3.1, which is the primitive of the development: the weight is carried under the integral, and the value is ∞ when μ is not absolutely continuous with respect to ν, as above.

The following proposition collects the properties of the Hellinger integral that the paper uses and that can be read off at this point; throughout, $\alpha > 1$.

Proposition A.20.

(1) Data processing. *For every sub-Markov kernel M,*

$$\mathrm{H}_\alpha\big(\mathbf{dlet}\ a \leftarrow \mu\ \mathbf{in}\ M(a) \,\big\|\, \mathbf{dlet}\ a \leftarrow \nu\ \mathbf{in}\ M(a)\big) \;\le\; \mathrm{H}_\alpha(\mu \,\|\, \nu) \;.$$

(2) Pushforwards. *For every measurable f, $\mathrm{H}_\alpha\big(f^\sharp \mu \,\big\|\, f^\sharp \nu\big) \le \mathrm{H}_\alpha(\mu \,\|\, \nu)$, with equality when f is injective with measurable inverse on its image.*

(3) Joint convexity. *For every measurable space I, every probability measure P on it and all families $(\mu_i)_{i\in I}$ and $(\nu_i)_{i\in I}$ whose mixtures are defined,*

$$\mathrm{H}_\alpha\big(\textstyle\int \mu_i \,\mathrm{d}P \,\big\|\, \int \nu_i \,\mathrm{d}P\big) \;\le\; \displaystyle\int \mathrm{H}_\alpha(\mu_i \,\|\, \nu_i)\ \mathrm{d}P \;\le\; \sup_i\, \mathrm{H}_\alpha(\mu_i \,\|\, \nu_i) \;.$$

(4) Products. $\mathrm{H}_\alpha(\mu_1 \otimes \mu_2 \,\|\, \nu_1 \otimes \nu_2) = \mathrm{H}_\alpha(\mu_1 \,\|\, \nu_1) \cdot \mathrm{H}_\alpha(\mu_2 \,\|\, \nu_2)$.

(5) Convolutions. *For finite measures on $\mathbb{R}^d$, $\mathrm{H}_\alpha(\mu_1 * \mu_2 \,\|\, \nu_1 * \nu_2) \le \mathrm{H}_\alpha(\mu_1 \,\|\, \nu_1) \cdot \mathrm{H}_\alpha(\mu_2 \,\|\, \nu_2)$.*

Proof. The Hellinger integral is the f-divergence of the convex $f(\cdot) = \cdot^{\alpha}$, so items 1 and 3 are the data-processing inequality and the joint convexity of f-divergences [53]. Item 2 is item 1 for the sub-Markov kernel $a \mapsto \delta_{f(a)}$ (Proposition A.17, item 6). Item 4 is Tonelli's theorem [50, Theorem 1.27] applied to the product density $\frac{\mathrm{d}(\mu_1\otimes\mu_2)}{\mathrm{d}(\nu_1\otimes\nu_2)}(a_1, a_2) = \frac{\mathrm{d}\mu_1}{\mathrm{d}\nu_1}(a_1) \cdot \frac{\mathrm{d}\mu_2}{\mathrm{d}\nu_2}(a_2)$, which factors the integral; the non-absolutely-continuous case is ∞ on both sides. Item 5 follows, convolution being the pushforward of the product under $+$: item 2 applied to $+$, then item 4. □

A.5 Gaussian Measures

Definition A.21 (Lebesgue measure)**.** The *Lebesgue measure* λ_d on $\mathbb{R}^d$ is the unique measure on the Borel sets that is invariant under every translation and gives the unit cube $[0,1]^d$ measure 1 [24].

Definition A.22 (Gaussian measure)**.** The *Gaussian measure* of mean $m \in \mathbb{R}^d$ and covariance $\sigma^2 I$, with $\sigma > 0$, is the measure with density

$$p_{m,\sigma}(x) \;\triangleq\; (2\pi\sigma^2)^{-d/2}\, \exp(-\|x - m\|^2 / 2\sigma^2)$$

with respect to λ_d, in the sense of Definition A.10: $\mathcal{N}(m, \sigma^2 I) \triangleq p_{m,\sigma} \cdot \lambda_d$. It is a probability measure, since $\int p_{m,\sigma} \,\mathrm{d}\lambda_d$ factors by Tonelli into d copies of $\int_{\mathbb{R}} (2\pi\sigma^2)^{-1/2} \exp(-t^2/2\sigma^2) \,\mathrm{d}t = 1$.

Proposition A.23.

(1) *For every affine isometry $T : x \mapsto Ux + c$ of $\mathbb{R}^d$, with U a linear isometry, $T^\sharp\, \mathcal{N}(m, \sigma^2 I) = \mathcal{N}(Um + c, \sigma^2 I)$. Translations ($U = I$) shift the mean and linear isometries ($c = 0$) rotate it.*

(2) $\mathcal{N}(m_1, \sigma_1^2 I) * \mathcal{N}(m_2, \sigma_2^2 I) = \mathcal{N}(m_1 + m_2, (\sigma_1^2 + \sigma_2^2) I)$.

Proposition A.24. *Let $\alpha > 1$.*

(1) Gaussians. $\mathrm{H}_\alpha\big(\mathcal{N}(m_1, \sigma^2 I) \,\big\|\, \mathcal{N}(m_2, \sigma^2 I)\big) = \exp\big(\alpha(\alpha-1)\,\|m_1 - m_2\|^2 / 2\sigma^2\big)$.

(2) Monotonicity in the mean. *For fixed $q \in [0,1]$ and $v > 0$, the map*

$$c \;\mapsto\; \mathrm{H}_\alpha((1-q)\mathcal{N}(0,v) + q\mathcal{N}(c,v) \,\|\, \mathcal{N}(0,v))$$

is nondecreasing on $[0, \infty)$.

B One-Sided Rules

Figure 7 gives one-sided rules, which act on one program at a time, leaving the other unchanged. The rules for assignment, release and conditionals are similar in spirit to the one-sided rules of apRHL [20]. However, due to the intrinsic asymmetry of the definition of the Hellinger integral, the rules for sampling and loops are not symmetric. To explain why, observe that by soundness of Rule [Samp-L], ϕ has to be an upper bound on $\big(\int k_1 \,\mathrm{d}\mu\big)^\alpha k_2(s_2)^{1-\alpha}$, where $\mu = [\![S_1]\!](m_1)$ is the output distribution of the left program. Its mass m is at most 1, and Jensen's inequality for the probability measure μ/m gives

$$\Big(\int k_1 \,\mathrm{d}\mu\Big)^\alpha \;\le\; m^{\alpha-1} \int k_1^\alpha \,\mathrm{d}\mu \;\le\; \int k_1^\alpha \,\mathrm{d}\mu\,,$$

so the bound is at most $\int k_1^\alpha \, k_2(s_2)^{1-\alpha} \,\mathrm{d}\mu$, which admissibility and the rule's premise bound by $\mathbb{E}_{v\sim\mu}\big[\psi[x\mapsto v](s)\big]$. Symmetrically, for [Samp-R], validity requires bounding $k_1(s_1)^\alpha \big(\int k_2 \,\mathrm{d}\mu\big)^{1-\alpha}$ with $\mu = [\![S_2]\!](m_2)$. Since $1-\alpha < 0$, the map $t \mapsto t^{1-\alpha}$ is convex and decreasing, and Jensen's inequality gives:

$$\Big(\int k_2 \,\mathrm{d}\mu\Big)^{1-\alpha} \;=\; m^{1-\alpha}\Big(\int k_2 \,\mathrm{d}\tfrac{\mu}{m}\Big)^{1-\alpha} \;\le\; m^{-\alpha} \int k_2^{1-\alpha} \,\mathrm{d}\mu\,.$$

Compared to [Samp-L], the inequality gains a factor $m^{-\alpha} \ge 1$. For this reason, [Samp-R] requires the sampling expression to be a Markov kernel, i.e., $m = 1$. The loop rules follow the same pattern, because non-termination loses mass. [While-L] does not require termination, whereas [While-R] requires the loop to exit within n iterations.

C Program Equivalence Calculus

This appendix provides a calculus for deriving the $c \equiv_{\mathcal{P}} c'$ premises of [Equiv]. The judgments of the calculus have the following shape:

$$\vdash c \equiv_{\mathcal{P}} c'$$

which expresses the equivalence of programs c and c' on states satisfying the predicate $\mathcal{P}$, as made formal by the following interpretation:

$$\models c \equiv_{\mathcal{P}} c' \triangleq \forall (m,\tau) \in \mathrm{St}.\ (m,\tau) \models \mathcal{P} \Rightarrow [\![c]\!](m,\tau) = [\![c']\!](m,\tau)$$

The basic equations of the derivation system are shown in Figure 8. The rules in Figure 9 express closure properties of the calculus. The interpretation of each of the rules is self-evident. The calculus is sound:

Theorem C.1 (Soundness of the equivalence calculus **Λ**)**.**

$$\vdash c \equiv_{\mathcal{P}} c' \implies \models c \equiv_{\mathcal{P}} c'.$$

One-sided rules

$$\text{[Assn-L]}\quad \frac{\mathrm{upd} = [x \mapsto E_1\langle 1\rangle]}{\vdash \{\mathcal{S}[\mathrm{upd}] \mid \psi[\mathrm{upd}]\}\ x \leftarrow E_1 \sim_\alpha \texttt{skip}\ \{\mathcal{S} \mid \psi\}}$$

$$\text{[Release-L]}\quad \frac{\mathrm{upd} = [\tau_1 \mapsto \tau_1 \cdot E_1\langle 1\rangle]}{\vdash \{\mathcal{S}[\mathrm{upd}] \mid \psi[\mathrm{upd}]\}\ \texttt{release}\ E_1 \sim_\alpha \texttt{skip}\ \{\mathcal{S} \mid \psi\}}$$

$$\text{[If-L]}\quad \frac{\vdash \{\mathcal{R} \wedge P_1\langle 1\rangle \mid \phi\}\ c_1 \sim_\alpha c_2\ \{\mathcal{S} \mid \psi\} \qquad \vdash \{\mathcal{R} \wedge \neg P_1\langle 1\rangle \mid \phi\}\ c_1' \sim_\alpha c_2\ \{\mathcal{S} \mid \psi\}}{\vdash \{\mathcal{R} \mid \phi\}\ \texttt{if}\ P_1\ \texttt{then}\ c_1\ \texttt{else}\ c_1'\ \texttt{fi} \sim_\alpha c_2\ \{\mathcal{S} \mid \psi\}}$$

$$\text{[Samp-L]}\quad \frac{\begin{array}{c}\forall s = ((m_1, \tau_1), s_2) \in \mathcal{R}.\ \mathcal{S}[x \mapsto v](s) \text{ for } [\![S_1]\!](m_1)\text{-almost every } v \\ \forall s = ((m_1, \tau_1), s_2) \in \mathcal{R}.\ \mathbb{E}_{v \sim [\![S_1]\!](m_1)}\left[\psi[x \mapsto v](s)\right] \le \phi(s)\end{array}}{\vdash \{\mathcal{R} \mid \phi\}\ x \xleftarrow{\$} S_1 \sim_\alpha \texttt{skip}\ \{\mathcal{S} \mid \psi\}}$$

$$\text{[Samp-R]}\quad \frac{\begin{array}{c}[\![S_2]\!] \text{ is a Markov kernel} \\ \forall s = (s_1, (m_2, \tau_2)) \in \mathcal{R}.\ \mathcal{S}[y \mapsto v](s) \text{ for } [\![S_2]\!](m_2)\text{-almost every } v \\ \forall s = (s_1, (m_2, \tau_2)) \in \mathcal{R}.\ \mathbb{E}_{v \sim [\![S_2]\!](m_2)}\left[\psi[y \mapsto v](s)\right] \le \phi(s)\end{array}}{\vdash \{\mathcal{R} \mid \phi\}\ \texttt{skip} \sim_\alpha y \xleftarrow{\$} S_2\ \{\mathcal{S} \mid \psi\}}$$

$$\text{[While-L]}\quad \frac{\forall k.\ \mathcal{R}_k \wedge \neg P_1\langle 1\rangle \models \mathcal{S} \wedge \psi \le \phi_k \qquad \forall k.\ \vdash \{\mathcal{R}_k \wedge P_1\langle 1\rangle \mid \phi_k\}\ c_1 \sim_\alpha \texttt{skip}\ \{\mathcal{R}_{k+1} \mid \phi_{k+1}\}}{\vdash \{\mathcal{R}_0 \mid \phi_0\}\ \texttt{while}\ P_1\ \texttt{do}\ c_1\ \texttt{od} \sim_\alpha \texttt{skip}\ \{\mathcal{S} \wedge \neg P_1\langle 1\rangle \mid \psi\}}$$

$$\text{[While-R]}\quad \frac{\begin{array}{c}\mathcal{R}_n \models \neg P_2\langle 2\rangle \qquad \forall k \le n.\ \mathcal{R}_k \wedge \neg P_2\langle 2\rangle \models \mathcal{S} \wedge \psi \le \phi_k \\ \forall k < n.\ \vdash \{\mathcal{R}_k \wedge P_2\langle 2\rangle \mid \phi_k\}\ \texttt{skip} \sim_\alpha c_2\ \{\mathcal{R}_{k+1} \mid \phi_{k+1}\}\end{array}}{\vdash \{\mathcal{R}_0 \mid \phi_0\}\ \texttt{skip} \sim_\alpha \texttt{while}\ P_2\ \texttt{do}\ c_2\ \texttt{od}\ \{\mathcal{S} \wedge \neg P_2\langle 2\rangle \mid \psi\}}$$

Fig. 7. One-sided rules of RRHL.

D Witness-Based Validity

This appendix gives the converse of Proposition 4.2: under conditions on the post-condition, a valid judgment has witnesses.

Theorem D.1 (Strong duality). *Let $\alpha > 1$. Assume that* St *carries a Polish topology whose Borel sets are its measurable sets, that $\mathcal{S}$ is closed in* $\mathrm{St} \times \mathrm{St}$*, and that for every $(s_1, s_2) \in \mathcal{R}$, for $[\![c_1]\!](s_1)$-almost every a there is b with $(a, b) \in \mathcal{S}$, and for $[\![c_2]\!](s_2)$-almost every b there is a with $(a, b) \in \mathcal{S}$. If $\models \{\mathcal{R} \mid \phi\}\ c_1 \sim_\alpha c_2\ \{\mathcal{S} \mid 1\}$, then for every $s = (s_1, s_2) \in \mathcal{R}$ there are $\mu_1, \mu_2 \in \mathrm{D}(\mathrm{St} \times \mathrm{St})$ such that*

- $\mu_1(\overline{\mathcal{S}}) = \mu_2(\overline{\mathcal{S}}) = 0$;
- $\pi_1^\sharp\, \mu_1 = [\![c_1]\!](s_1)$;
- $\pi_2^\sharp\, \mu_2 = [\![c_2]\!](s_2)$, *where* $\pi_1, \pi_2 : \mathrm{St} \times \mathrm{St} \to \mathrm{St}$ *are the projections, and*
- $\mathrm{H}_\alpha(\mu_1 \,\|\, \mu_2) \le \phi(s)$.

If Val carries a Polish topology whose Borel sets are its measurable sets, then so does $\mathrm{St} = \mathrm{Val}^{\mathrm{Var}} \times \bigcup_n \mathrm{Val}^n$, since Var is countable and countable products and countable disjoint unions of Polish spaces are Polish. For this topology, the closedness of $\mathcal{S}$ holds for all the post-conditions of §5, which are conjunctions of equalities and non-strict inequalities between continuous functions of the state. These post-conditions contain the diagonal, so the last hypothesis of the theorem holds with $b = a$ and $a = b$.

E Detailed Case Studies

This appendix complements the case studies of §5 with detailed derivations and comparisons with other analyses from the literature. All parameters are finite reals, and the hypotheses stated for each case study imply those of its Lean theorem.

Control flow and sequencing

$$\frac{}{\texttt{while } P \texttt{ do } c \texttt{ od} \equiv_{\mathcal{P}} \texttt{if } P \texttt{ then } (c; \texttt{while } P \texttt{ do } c \texttt{ od}) \texttt{ else skip fi}}\ [\text{E-Unroll}] \qquad \frac{}{(c_1; c_2); c_3 \equiv_{\mathcal{P}} c_1; (c_2; c_3)}\ [\text{E-Assoc}]$$

$$\frac{}{c; \texttt{skip} \equiv_{\mathcal{P}} c}\ [\text{E-SeqSkip}] \qquad \frac{}{\texttt{skip}; c \equiv_{\mathcal{P}} c}\ [\text{E-SkipSeq}]$$

Locality and framing

$$\frac{}{\texttt{local } x \texttt{ in } c \equiv_{\mathcal{P}} x \leftarrow d_x; (c; x \leftarrow d_x)}\ [\text{E-Loc-D}] \qquad \frac{x \text{ does not occur in } c}{x \leftarrow v; c \equiv_{\mathcal{P}} c; x \leftarrow v}\ [\text{E-Frame}]$$

Sampling

$$\frac{x \text{ does not occur in } c}{c; x \xleftarrow{\$} \mu \equiv_{\mathcal{P}} x \xleftarrow{\$} \mu; c}\ [\text{E-Defer}] \qquad \frac{x \neq y \quad E \text{ does not read } y}{x \leftarrow E; y \xleftarrow{\$} S \equiv_{\mathcal{P}} y \xleftarrow{\$} S[E/x]; x \leftarrow E}\ [\text{E-Comm}]$$

$$\frac{\kappa \text{ of total mass one} \quad P \text{ does not read } y \quad y \text{ does not occur in } c_2}{\texttt{if } P \texttt{ then local } y \texttt{ in } (y \xleftarrow{\$} \kappa; c_1) \texttt{ else } c_2 \texttt{ fi} \equiv_{y=d_y} \texttt{local } y \texttt{ in } (y \xleftarrow{\$} \kappa; \texttt{if } P \texttt{ then } c_1 \texttt{ else } c_2 \texttt{ fi})}\ [\text{E-Defer-C}]$$

$$\frac{x \neq b}{\texttt{local } b \texttt{ in } (b \xleftarrow{\$} \kappa; x \xleftarrow{\$} S) \equiv_{\mathcal{P}} b \leftarrow d_b; x \xleftarrow{\$} (\textbf{dlet } b \leftarrow \kappa \textbf{ in } S)}\ [\text{E-Merge}]$$

$$\frac{x \neq b}{\texttt{local } b \texttt{ in } (b \xleftarrow{\$} \kappa; x \leftarrow E) \equiv_{\mathcal{P}} b \leftarrow d_b; x \xleftarrow{\$} (\textbf{dlet } b \leftarrow \kappa \textbf{ in } \delta_E)}\ [\text{E-Push}]$$

Writes to one variable

$$\frac{}{\texttt{if } P \texttt{ then } x \leftarrow E \texttt{ else skip fi} \equiv_{\mathcal{P}} x \leftarrow (\textit{if } P \textit{ then } E \textit{ else } x)}\ [\text{E-Cond-W}] \qquad \frac{}{x \leftarrow E_1; x \leftarrow E_2 \equiv_{\mathcal{P}} x \leftarrow E_2[E_1/x]}\ [\text{E-AA}]$$

$$\frac{}{x \leftarrow E; x \xleftarrow{\$} S \equiv_{\mathcal{P}} x \xleftarrow{\$} S[E/x]}\ [\text{E-AS}] \qquad \frac{}{x \xleftarrow{\$} S; x \leftarrow E \equiv_{\mathcal{P}} x \xleftarrow{\$} (\textbf{dlet } v \leftarrow S \textbf{ in } \delta_{E[v/x]})}\ [\text{E-SA}]$$

$$\frac{}{x \xleftarrow{\$} S_1; x \xleftarrow{\$} S_2 \equiv_{\mathcal{P}} x \xleftarrow{\$} (\textbf{dlet } v \leftarrow S_1 \textbf{ in } S_2[v/x])}\ [\text{E-SS}]$$

Fig. 8. The equivalence equations.

E.1 DP-SGD, Transcript Model

Under the transcript model, we derive that the DP-SGD algorithm of Figure 1 is $\left(\alpha,\ T\,\frac{\log \Upsilon_\alpha(q,\Gamma,v)}{\alpha-1}\right)$-RDP, where the cost of a single round is

$$\Upsilon_\alpha(q,\Gamma,v) \ \triangleq\ \mathrm{H}_\alpha((1-q)\,\mathcal{N}(0,v) + q\,\mathcal{N}(\Gamma,v) \,\|\, \mathcal{N}(0,v))\,.$$

The judgment. With Proposition 4.4 at hand, the Rényi bound reduces to deriving the following judgment:

$$\vdash \{D^{\langle 1\rangle} \simeq_j D^{\langle 2\rangle} \wedge \tau^{\langle 1\rangle} = \tau^{\langle 2\rangle} \mid \Upsilon_\alpha(q,\Gamma,v)^T\}\ \text{dpsgd} \sim_\alpha \text{dpsgd}\ \{\theta^{\langle 1\rangle} = \theta^{\langle 2\rangle} \wedge \tau^{\langle 1\rangle} = \tau^{\langle 2\rangle} \mid 1\}\,. \quad (15)$$

Here dpsgd is the program of Figure 1. The pre-condition asks the two runs to start on adjacent datasets differing at an arbitrary index j, as defined by the adjacency $D \simeq_j D' \triangleq (\forall i \neq j.\ D[i] = D'[i]) \wedge (D[j] = \bot \vee D'[j] = \bot)$; the post-condition imposes that they end on the same model and have released the same transcript. The pre-budget is the T-th power of $\Upsilon_\alpha(q,\Gamma,v)$, the cost of a single round.

The proof starts by combining [Release] with an application of [While-Prod] to the following judgment:

$$\vdash \{\mathcal{I}_k \mid \Upsilon_\alpha(q,\Gamma,v)\}\ \text{body} \sim_\alpha \text{body}\ \{\mathcal{I}_{k+1} \mid 1\}, \quad (16)$$

Equivalence and restriction

$$\frac{}{c \equiv_{\mathcal{P}} c}\ [\text{E-Refl}] \qquad \frac{c \equiv_{\mathcal{P}} c'}{c' \equiv_{\mathcal{P}} c}\ [\text{E-Sym}] \qquad \frac{c \equiv_{\mathcal{P}} c'' \quad c'' \equiv_{\mathcal{P}} c'}{c \equiv_{\mathcal{P}} c'}\ [\text{E-Trans}] \qquad \frac{\mathcal{P} \models \mathcal{P}' \quad c \equiv_{\mathcal{P}'} c'}{c \equiv_{\mathcal{P}} c'}\ [\text{E-Restr}]$$

Congruence

$$\frac{c_1 \equiv_{\mathcal{P}} c_1' \quad c_2 \equiv_{\top} c_2'}{c_1; c_2 \equiv_{\mathcal{P}} c_1'; c_2'}\ [\text{E-Seq}] \qquad \frac{\mathcal{P} \models \mathcal{P}'[x \mapsto [\![E]\!](m)] \quad c \equiv_{\mathcal{P}'} c'}{x \leftarrow E; c \equiv_{\mathcal{P}} x \leftarrow E; c'}\ [\text{E-Seq-A}]$$

$$\frac{c_1 \equiv_{\mathcal{P} \wedge P} c_1' \quad c_2 \equiv_{\mathcal{P} \wedge \neg P} c_2'}{\texttt{if } P \texttt{ then } c_1 \texttt{ else } c_2 \texttt{ fi} \equiv_{\mathcal{P}} \texttt{if } P \texttt{ then } c_1' \texttt{ else } c_2' \texttt{ fi}}\ [\text{E-If}] \qquad \frac{\mathcal{P} \models \mathcal{P}'[x \mapsto d_x] \quad c \equiv_{\mathcal{P}'} c'}{\texttt{local } x \texttt{ in } c \equiv_{\mathcal{P}} \texttt{local } x \texttt{ in } c'}\ [\text{E-Loc}]$$

Collapse and counted loops

$$\frac{\mathcal{P} \models \neg P}{\texttt{if } P \texttt{ then } c_1 \texttt{ else } c_2 \texttt{ fi} \equiv_{\mathcal{P}} c_2}\ [\text{E-If-F}] \qquad \frac{\mathcal{P} \models x = d_x}{x \leftarrow d_x \equiv_{\mathcal{P}} \texttt{skip}}\ [\text{E-Idle}]$$

$$\frac{c \text{ writes no } x \quad n \in \mathbb{N} \text{ a constant}}{\texttt{while } x < n \texttt{ do } c; x \leftarrow x + 1 \texttt{ od} \equiv_{x+k=n} (c; x \leftarrow x + 1)^k}\ [\text{E-Count}]$$

Leaves, restriction transport and ghosts

$$\frac{\mathcal{P} \models [\![E']\!] = [\![E]\!]}{x \leftarrow E \equiv_{\mathcal{P}} x \leftarrow E'}\ [\text{E-Leaf-A}] \qquad \frac{\mathcal{P} \models [\![S']\!] = [\![S]\!]}{x \xleftarrow{\$} S \equiv_{\mathcal{P}} x \xleftarrow{\$} S'}\ [\text{E-Leaf-S}]$$

$$\frac{\mathcal{P} \text{ reads only } \bar{y} \quad c_1 \text{ writes no } y \in \bar{y} \quad c_2 \equiv_{\mathcal{P}} c_2'}{c_1; c_2 \equiv_{\mathcal{P}} c_1; c_2'}\ [\text{E-Seq-V}]$$

$$\frac{y \text{ occurs in neither } c \text{ nor } c' \quad \mathcal{P} \text{ does not read } y \quad c \equiv_{\mathcal{P} \wedge y = d_y} c'}{c \equiv_{\mathcal{P}} c'}\ [\text{E-Ghost}]$$

Fig. 9. Closure properties of the equivalence calculus.

where $\mathcal{I}_k \triangleq D^{\langle 1 \rangle} \simeq_j D^{\langle 2 \rangle} \wedge \theta^{\langle 1 \rangle} = \theta^{\langle 2 \rangle} \wedge \tau^{\langle 1 \rangle} = \tau^{\langle 2 \rangle} \wedge t^{\langle 1 \rangle} = t^{\langle 2 \rangle} = k$, and body is the body of the loop of Figure 1. In deriving (16), the crucial step is to replace the body of the `local` construct, via [Equiv], by the equivalent single sampling statement

$$\text{smp} \triangleq w \xleftarrow{\$} \int \left[(1-q)\, \mathcal{N}(g(B), v) + q\, \mathcal{N}(g(B) + u, v) \right] d\text{Btc}_q(B)\,, \tag{17}$$

where $g(B) \triangleq \sum_{i \in B} \text{clip}_\Gamma(\nabla \ell(\theta, D[i]))$ and $u \triangleq \text{clip}_\Gamma(\nabla \ell(\theta, D[j]))$, and where B ranges over the subsets of the indices other than the differing one j, under their law Btc_q (each index enters independently with probability q): the coin of j is independent of the others, so it factors out of the batch draw and stays inside the distribution being sampled. The sampling expression is thus the average, over all the batches B drawn from Btc_q, of two cases: if the j-th record is not included — with probability $1-q$ — the noise has as mean the gradient sum $g(B)$ of the other records of the batch; otherwise — with probability q — the j-th record is sampled too, so its contribution u is added and the mean of the noise is shifted by u. The two runs only differ in u: $\|u\| \leq \Gamma$ in the run that holds the record j, and $u = 0$ in the other, where $D[j] = \bot$, so there the sampling expression collapses to $\int \mathcal{N}(g(B), v)\, d\text{Btc}_q(B)$. We then apply Rule [Samp], obtaining the following judgment

$$\vdash \{\mathcal{I}_k \mid \Upsilon_\alpha(q, \Gamma, v)\}\ \text{smp} \sim_\alpha \text{smp}\ \{\mathcal{I}_k \wedge w^{\langle 1 \rangle} = w^{\langle 2 \rangle} \mid 1\}\,. \tag{18}$$

The budget bound follows by case analysis: when $D^{\langle 2\rangle}[j] = \bot$, we have:

$$\mathrm{H}_\alpha\Big(\int (1-q)\,\mathcal{N}(g(B),\,v) + q\,\mathcal{N}(g(B)+u,\,v)\;d\mathrm{Btc}_q(B)\,\Big\|\,\int \mathcal{N}(g(B),\,v)\;d\mathrm{Btc}_q(B)\Big) \;\le$$
$$\int \mathrm{H}_\alpha(\mu_B \,\|\, \nu_B)\;\; d\mathrm{Btc}_q(B) \;=\; \Upsilon_\alpha(q,\|u\|,v) \;\le\; \Upsilon_\alpha(q,\Gamma,v)\,,$$

where μ_B and ν_B are the two integrands at B; the inequality holds because the Hellinger integral is jointly convex; the equality holds because it is invariant under shifts and rotations, which reduce the pair at B to the one-dimensional pair defining $\Upsilon_\alpha(q,\|u\|,v)$; and the last step is monotonicity in the mean, as $\|u\| \le \Gamma$ (Proposition A.24, items 2 and 2 ⋀). When $D^{\langle 1\rangle}[j] = \bot$, the two arguments of the Hellinger integral are exchanged, and the same chain bounds it by $\Lambda_\alpha(q,\|u\|,v) \le \Lambda_\alpha(q,\Gamma,v) \le \Upsilon_\alpha(q,\Gamma,v)$, the last step by Lemma E.1 below, which is Mironov et al. [59, Corollary 7].

Lemma E.1 (Domination of the reversed order ⋀). *For every real $\alpha > 1$, $q \in [0,1]$, $\Gamma \in \mathbb{R}$ and $v > 0$, $\Lambda_\alpha(q,\Gamma,v) \le \Upsilon_\alpha(q,\Gamma,v)$.*

Evaluating the per-round cost. The derivation expresses the guarantee in terms of $\Upsilon_\alpha(q,\Gamma,v)$. The deployed DP-SGD instances evaluate it by leveraging the identities of Mironov et al. [59, Section 3.3], which we formalize below:

Proposition E.2 (Evaluating the per-round cost ⋀). *Let $v > 0$. Write $L \triangleq \mathrm{d}\mathcal{N}(\Gamma,v)/\mathrm{d}\mathcal{N}(0,v)$ for the likelihood ratio,* $\mathrm{erfc}(x) \triangleq \frac{2}{\sqrt{\pi}}\int_x^\infty \exp(-t^2)\,\mathrm{d}t$ *for the complementary error function, and*

$$z_1 \;\triangleq\; \frac{\Gamma}{2} + \frac{v}{\Gamma}\,\log\frac{1-q}{q}$$

for the point at which $(1-q)\mathcal{N}(0,v)$ and $q\mathcal{N}(\Gamma,v)$ have the same density. Then:

(1) *for every real $\alpha > 1$:*

$$\Upsilon_\alpha(q,\Gamma,v) \;= \frac{1}{2}\sum_{n\ge 0}\binom{\alpha}{n}(1-q)^{\alpha-n}q^n\;\exp\Big(\tfrac{n(n-1)\Gamma^2}{2v}\Big)\,\mathrm{erfc}\Big(\tfrac{n\Gamma-z_1}{\sqrt{2v}}\Big)$$
$$+\;\frac{1}{2}\sum_{n\ge 0}\binom{\alpha}{n}(1-q)^{n}q^{\alpha-n}\;\exp\Big(\tfrac{(\alpha-n)(\alpha-n-1)\Gamma^2}{2v}\Big)\,\mathrm{erfc}\Big(\tfrac{z_1-(\alpha-n)\Gamma}{\sqrt{2v}}\Big)\,,$$

where $\binom{\alpha}{n}$ is the generalized binomial coefficient;

(2) *for every integer $m > 1$:*

$$\Upsilon_m(q,\Gamma,v) \;=\; \sum_{k=0}^{m}\binom{m}{k}q^k(1-q)^{m-k}\exp\Big(\frac{k(k-1)\Gamma^2}{2v}\Big)\,.$$

Evaluating Item 1 numerically requires truncating its series; neither the truncation error nor floating-point evaluation is verified.

From the judgment to the reported guarantee. Once (15) is established, Proposition 4.4 entails that DP-SGD is $\big(\alpha,\; T\,\frac{\log\Upsilon_\alpha(q,\Gamma,v)}{\alpha-1}\big)$-individually RDP for the family $(\simeq_j)_j$, and Lemma 3.7 turns that family of guarantees — whose budget does not depend on j — into the same guarantee for the union $\simeq \triangleq \bigcup_j \simeq_j$ of the adjacencies, which is the standard adjacency relation. The conversion of Proposition 3.5 then gives the (ϵ,δ) form of the guarantee, and Proposition E.2 the number that Υ_α stands for.

Corollary E.3 ((ϵ, δ) guarantee 🅛). *Given the judgment* (15)*, for every* $\alpha > 1$ *and every* $\delta \in (0, 1)$*, the program dpsgd is* (ϵ, δ)*-differentially private with*

$$\epsilon \;=\; \frac{T \log \Upsilon_\alpha(q, \Gamma, v)}{\alpha - 1} \;+\; \log \frac{\alpha - 1}{\alpha} \;-\; \frac{\log \delta + \log \alpha}{\alpha - 1}.$$

E.2 DP-SGD, Last Iterate

With Proposition 4.4 at hand, we prove Rényi differential privacy of DP-SGD by deriving, for every $1 \le R \le T$, the judgment

$$\vdash \{D^{\langle 1\rangle} \simeq_j D^{\langle 2\rangle} \wedge \tau^{\langle 1\rangle} = \tau^{\langle 2\rangle} \mid \Omega_\alpha(R)\}\ \mathsf{lastIter} \sim_\alpha \mathsf{lastIter}\ \{\theta^{\langle 1\rangle} = \theta^{\langle 2\rangle} \wedge \tau^{\langle 1\rangle} = \tau^{\langle 2\rangle} \mid 1\}\,. \tag{19}$$

As in the transcript model, the two runs start on datasets that are adjacent at an arbitrary record j. Since the budget does not depend on j, Lemma 3.7 takes the family of judgments, one per record, to the guarantee at the union $\simeq \triangleq \bigcup_j \simeq_j$ of the adjacencies.

Judgment (19) is derived by combining [Release] with an application of [While-Prod] to the following family of judgments:

$$\begin{aligned} &\vdash \{\mathcal{I}_k \mid F_k\}\ \mathsf{body} \sim_\alpha \mathsf{body}\ \{\mathcal{I}_{k+1} \mid 1\} \qquad \text{for every } k < T, \\ \text{where}\quad &\mathcal{I}_k \;\triangleq\; D^{\langle 1\rangle} \simeq_j D^{\langle 2\rangle} \;\wedge\; \|\theta^{\langle 1\rangle} - \theta^{\langle 2\rangle}\| \le z_k \;\wedge\; \tau^{\langle 1\rangle} = \tau^{\langle 2\rangle} \;\wedge\; t^{\langle 1\rangle} = t^{\langle 2\rangle} = k\,, \end{aligned} \tag{20}$$

where the radius and the per-round budget are

$$z_k \;=\; \begin{cases} \Delta & k < T - R, \\ \dfrac{(T-k)\,\Delta}{R} & T - R \le k \le T, \end{cases} \qquad F_k \;=\; \begin{cases} 1 & k < T - R, \\ \Omega_\alpha(R)^{1/R} & T - R \le k < T, \end{cases}$$

so that $z_T = 0$ and $\prod_{k<T} F_k = \Omega_\alpha(R)$. When $R = T$, both runs start from the same model, which is within the radius $z_0 = \Delta$.

In deriving (19), we exploit that Rule [While-Prod] does not require a single judgment derivation in its premise, but rather an indexed family of judgments. This allows us to provide different derivations for the rounds with $k < T - R$ and for those with $k \ge T - R$.

On each of the last R rounds [Equiv] exploits (4), the split (6) and the identity

$$[\![x \xleftarrow{\$} S; c]\!](m, \tau) \;=\; \mathbf{dlet}\ v \leftarrow [\![S]\!](m)\ \mathbf{in}\ [\![c]\!](m[x \mapsto v],\ \tau)\,,$$

to turn the block into

$$B \xleftarrow{\$} \mathrm{Btc}_q; z \xleftarrow{\$} (1-q)\,\mathcal{N}(0,\ \sigma_2^2) + q\,\mathcal{N}(u,\ \sigma_2^2); w \xleftarrow{\$} \mathcal{N}(g(B) + z,\ \sigma_1^2)\,, \tag{21}$$

with Btc_q, $g(B)$ and u as in the transcript model. The first sampling statement produces the same distribution on both runs and is discharged by Rule [Samp], with no additional cost. The second sampling, which merges the sampling of the differing element with the computation of the gradient and the noising, is discharged exactly as in the transcript model (§5.1.1), at budget $\Upsilon_\alpha(q, \Gamma, \sigma_2^2)$. The last statement samples the Gaussian noise. On that statement we apply [Samp-Gap] with shift $G = g^{\langle 1\rangle} - g^{\langle 2\rangle} - e$, where $e \triangleq -\min\{\|\delta\|, \Delta/R\}\,\delta/\gamma\|\delta\|$ (and $e \triangleq 0$ if $\delta = 0$) for $\delta \triangleq (\theta^{\langle 1\rangle} - \gamma g^{\langle 1\rangle}) - (\theta^{\langle 2\rangle} - \gamma g^{\langle 2\rangle})$, the difference of the two updated models before noising: e has norm at most $\Delta/R\gamma$ and brings that difference closer by $\min\{\|\delta\|, \Delta/R\}$, at a budget of $\exp\big(\alpha(\alpha-1)\Delta^2/2R^2\gamma^2\sigma_1^2\big)$. The two budgets multiply to $\Omega_\alpha(R)^{1/R}$.

On the first $T - R$ rounds, the inner loop Pois_q of line 5 is derived synchronously by [While] — at each of its iterations the two runs draw the coin b from the same $\mathrm{Bern}(q)$, which [Samp] equates at budget 1, by picking the same coin on the two sides; [If] enters the same branch and the assignments keep the batch and the counter equal. The two runs therefore leave the sub-loop

having drawn the *same* batch B, and line 6 is discharged by [Assn] on each side, which leaves the two gradient sums $g^{\langle 1\rangle}$ and $g^{\langle 2\rangle}$ differing only through the datasets.

The noise of line 7 is then discharged by [Samp-Gap] at the gap $G \triangleq g^{\langle 1\rangle} - g^{\langle 2\rangle}$. Shifting $\mathcal{N}(g^{\langle 2\rangle}, v)$ by G gives $\mathcal{N}(g^{\langle 1\rangle}, v)$ exactly, so the round does not increase the privacy budget. At the end of each round, the invariant asks for $\|\theta^{\langle 1\rangle} - \theta^{\langle 2\rangle}\| \le \Delta$, which [Assn] re-establishes at no cost, because the range of Π has diameter Δ.

Deriving the bound of Altschuler and Talwar [6]. From (19) we derive, as a corollary, a bound of the form of Altschuler and Talwar [6, Theorem 3.1]. The main difficulty is that they operate on a different adjacency, which replaces a record: $D \approx_j D'$ if D and D' agree everywhere except at j, where both may hold a record. Our adjacency $\simeq_j$ instead requires one of the two records to be $\perp$. Observe, however, that

$$D \approx_j D' \implies D \simeq_j D'' \simeq_j D', \qquad \text{where } D'' \triangleq D[j \mapsto \perp].$$

By leveraging this observation, we derive a bound for $\approx_j$ from two bounds for $\simeq_j$, via the weak triangle inequality for the Hellinger integral, a consequence of the Cauchy–Schwarz inequality: for $\alpha > 1$ and probability distributions μ, ξ, ν,

$$\mathrm{H}_\alpha(\mu \,\|\, \nu) \;\le\; \big(\mathrm{H}_{2\alpha}(\mu \,\|\, \xi) \cdot \mathrm{H}_{2\alpha-1}(\xi \,\|\, \nu)\big)^{1/2}. \qquad (22)$$

Taking for ξ the output distribution on D'', this yields the privacy bound: if lastIter is $(2\alpha, \rho_1)$-RDP and $(2\alpha-1, \rho_2)$-RDP with respect to $\simeq_j$, then it is $\big(\alpha, {}^{((2\alpha-1)\rho_1+(2\alpha-2)\rho_2)}/_{2(\alpha-1)}\big)$-RDP with respect to $\approx_j$, and, by Lemma 3.7, with respect to $\bigcup_j \approx_j$. Instantiating this result with the bound of Mironov et al. [59, Theorem 11] on the subsampled Gaussian gives the following bound.

Proposition E.4 (Last iterate at replace-one adjacency). *Let $\sigma_1 = \sigma_2 = \sigma$ and $\bar{T} \triangleq \max\big(1, \lceil {}^{\Delta}/_{\gamma q \Gamma} \rceil\big)$. Let the losses be convex, β-smooth and Γ-Lipschitz, and let $0 < \eta \le {}^{2q}/_{\beta}$. Let $\alpha \ge {}^{3}/_{2}$, $q \le {}^{1}/_{5}$ and $\sigma \ge 4\Gamma$, and let each of the orders $\alpha' \in \{2\alpha, 2\alpha-1\}$ satisfy, with $\bar{\sigma} \triangleq \sigma/\Gamma$ and $\mathsf{L}_{\alpha'} \triangleq \log(1 + {}^{1}/_{q(\alpha'-1)})$, the conditions of Mironov et al. [59, Theorem 11]: $\alpha' \le {}^{\bar{\sigma}^2 \mathsf{L}_{\alpha'}}/_{2} - 2\log\bar{\sigma}$ and $\alpha' \le ({}^{\bar{\sigma}^2 \mathsf{L}^2_{\alpha'}}/_{2} - \log 5 - 2\log\bar{\sigma})/(\mathsf{L}_{\alpha'} + \log(q\alpha') + {}^{1}/_{2\bar{\sigma}^2})$. Then lastIter is $\big(\alpha, 20\,\alpha\, {}^{q^2\Gamma^2}/_{\sigma^2} \cdot \min\{T, \bar{T}\}\big)$-RDP with respect to $\bigcup_j \approx_j$.*

Proposition E.4 has the form of Theorem 3.1 of Altschuler and Talwar [6], which is stated asymptotically. In our notation, with Γ their Lipschitz constant and $q = {}^{b}/_{|D|}$ the rate at which they draw a batch of b records, their level is at most a constant multiple of $\alpha\, {}^{q^2\Gamma^2}/_{\sigma^2} \cdot \min\{T, \bar{T}\}$.

The two results rest on different hypotheses. Our step condition $\eta \le {}^{2q}/_{\beta}$ is stricter than their $\eta \le {}^{2}/_{\beta}$ by a factor q (item 4 below). We additionally require $\alpha \ge {}^{3}/_{2}$. These differences are due to the slightly different setting, which we adopt for consistency with §5.1.1. We believe that a similar result can be obtained in their setting as well. The differences between the two settings are discussed below.

Comparison with the algorithm of Altschuler and Talwar [6]. For consistency with §5.1.1, the algorithm lastIter analysed here deviates from the one in Altschuler and Talwar [6] in four respects.

(1) They draw a minibatch of fixed size b out of n records without replacement, whereas we draw a Poisson batch of rate q. As a consequence, in their case the probability that the differing record is in the batch is ${}^{b}/_{n}$, whereas for us it is q.
(2) Although we clip each gradient to Γ, the clipping is the identity under their assumption that the losses are Γ-Lipschitz.
(3) Their adjacency replaces a record, whereas ours replaces it by $\perp$. Our proof is indirect and passes through the $\simeq$ adjacency, which is the source of the assumption $\alpha \ge {}^{3}/_{2}$ in

Proposition E.4. We believe that deriving the judgment directly would align the assumptions of the two theorems.

(4) Both analyses derive the non-expansiveness (7) from convexity and smoothness. Our update is normalised by the expected batch size $q|D|$, so a batch holding every record takes a step $1/q$ times the average, whence $\eta \le 2q/\beta$ instead of $\eta \le 2/\beta$.

We believe that drawing batches of fixed size, as they do, would recover their step condition.

E.3 Noisy Gradient Descent with Per-Record Rényi Filters

We now show that the algorithm of Figure 5 is $\big(\alpha,\ \frac{\alpha B}{2v}\big)$-individually RDP for every element of the dataset. We do so by deriving the following judgment

$$\vdash \{D^{\langle 1\rangle} \simeq_j D^{\langle 2\rangle} \wedge \tau^{\langle 1\rangle} = \tau^{\langle 2\rangle} \mid \exp\big((\alpha-1)\tfrac{\alpha B}{2v}\big)\}\ \mathsf{filter} \sim_\alpha \mathsf{filter}\ \{\theta^{\langle 1\rangle} = \theta^{\langle 2\rangle} \wedge \tau^{\langle 1\rangle} = \tau^{\langle 2\rangle} \mid 1\}\,, \tag{23}$$

where $j < |D|$ is an index at the meta-logical level of an arbitrary element of D and $D^{\langle 1\rangle} \simeq_j D^{\langle 2\rangle}$ states that the two datasets are adjacent at j — they agree off j and the record j is $\perp$ in one of them.

Following the same recipe as the previous case studies, the crucial step for deriving (10) is using [While] to derive the following judgment:

$$\vdash \{\mathcal{R}_j \wedge t^{\langle 1\rangle} = t^{\langle 2\rangle} = k \mid \omega\}\ \mathsf{round} \sim_\alpha \mathsf{round}\ \{\mathcal{R}_j \wedge t^{\langle 1\rangle} = t^{\langle 2\rangle} = k+1 \mid \omega\} \tag{24}$$

for every $k < T$, where round is the body of the main loop of Figure 5. The budget here depends on the state, so the counted [While-Prod] does not apply and [While] is used directly, with invariant $\mathcal{R}_j \wedge t^{\langle 1\rangle} = t^{\langle 2\rangle} = k$ at round k. In the judgment, the logical invariant

$$\begin{aligned}\mathcal{R}_j \triangleq\ & D^{\langle 1\rangle} \simeq_j D^{\langle 2\rangle} \wedge \theta^{\langle 1\rangle} = \theta^{\langle 2\rangle} \wedge \tau^{\langle 1\rangle} = \tau^{\langle 2\rangle} \\ & \wedge (\forall l \neq j.\ s^{\langle 1\rangle}[l] = s^{\langle 2\rangle}[l]) \wedge 0 \le s^{\langle 1\rangle}[j] + s^{\langle 2\rangle}[j] \le B \wedge \textstyle\bigwedge_{i\in\{1,2\}} (D^{\langle i\rangle}[j] = \perp \Rightarrow s^{\langle i\rangle}[j] = 0)\end{aligned} \tag{25}$$

asserts adjacency of the datasets, identity of the models and transcripts, identity of the arrays $s^{\langle 1\rangle}$ and $s^{\langle 2\rangle}$ at positions $l \neq j$, boundedness of $s[j]$. It finally imposes that the budget $s[j]$ of the run without the record j is 0, so that $s^{\langle 1\rangle}[j] + s^{\langle 2\rangle}[j]$ is the budget spent by the run that holds the record. The quantitative invariant ω is $\exp\big(\frac{(\alpha-1)\alpha\,(B - s^{\langle 1\rangle}[j] - s^{\langle 2\rangle}[j])}{2v}\big)$. The derivation of (11) proceeds as follows: by backward (weakest pre-condition) reasoning through [Assn] and [Release], we push the post-condition $\mathcal{R}_j \wedge t^{\langle 1\rangle} = t^{\langle 2\rangle} = k+1$ through the trailing statements of line 11, yielding the intermediate judgment

$$\vdash \{\mathcal{R}_j \wedge t^{\langle 1\rangle} = t^{\langle 2\rangle} = k \wedge (acc + z)^{\langle 1\rangle} = (acc + z)^{\langle 2\rangle} \mid \omega\}\ \mathsf{tail} \sim_\alpha \mathsf{tail}\ \{\mathcal{R}_j \wedge t^{\langle 1\rangle} = t^{\langle 2\rangle} = k+1 \mid \omega\}\,, \tag{26}$$

where tail is the block of statements at line 11. The application of Rule [Samp-Gap] described in §5.2, with gap $G \triangleq acc^{\langle 2\rangle} - acc^{\langle 1\rangle}$, gives exactly

$$\begin{aligned}&\vdash \{\mathcal{R}_j^k \mid \omega \cdot \phi_G\}\ z \xleftarrow{\$} \mathcal{N}(0,v) \sim_\alpha z \xleftarrow{\$} \mathcal{N}(0,v)\ \{\mathcal{R}_j^k \wedge (acc+z)^{\langle 1\rangle} = (acc+z)^{\langle 2\rangle} \mid \omega\}\,, \\ &\qquad \text{where} \quad \mathcal{R}_j^k \triangleq \mathcal{R}_j \wedge t^{\langle 1\rangle} = t^{\langle 2\rangle} = k \quad \text{and} \quad \phi_G \triangleq \exp\big(\alpha(\alpha-1)\|G\|^2/2v\big)\,.\end{aligned} \tag{27}$$

On the inner loop, we derive the judgment

$$\vdash \{\mathcal{J}_0 \wedge t^{\langle 1\rangle} = t^{\langle 2\rangle} = k \mid \omega \cdot \phi_G\}\ \mathsf{recloop} \sim_\alpha \mathsf{recloop}\ \{\mathcal{J}_{|D|} \wedge t^{\langle 1\rangle} = t^{\langle 2\rangle} = k \mid \omega \cdot \phi_G\} \tag{28}$$

by application of [While], instantiated at $\mathcal{J}_i$. Here recloop is the inner loop and $\mathcal{J}_i$ is the invariant

$$\mathcal{J}_i \triangleq \mathcal{R}_j \wedge i^{\langle 1\rangle} = i^{\langle 2\rangle} = i \wedge \big(i \le j \Rightarrow acc^{\langle 1\rangle} = acc^{\langle 2\rangle}\big) \wedge \big\|acc^{\langle 1\rangle} - acc^{\langle 2\rangle}\big\|^2 \le s^{\langle 1\rangle}[j] + s^{\langle 2\rangle}[j]\,. \tag{29}$$

Its premise is discharged by combining [Assn] and [Seq] to propagate the two assignments at the logical level, and [Conseq] to derive the invariant $\mathcal{J}_{i+1}$.

We finally use [Seq] to combine (26), (27) and (28) and [Assn] to perform the backwards substitutions of the first two assignments. This yields the judgment

$$\vdash \{\mathcal{J}_0[i\mapsto 0, acc\mapsto 0] \wedge t^{\langle 1\rangle} = t^{\langle 2\rangle} = k \mid (\omega \cdot \phi_G)[acc\mapsto 0]\} \text{ round } \sim_\alpha \text{ round } \{\mathcal{R}_j \wedge t^{\langle 1\rangle} = t^{\langle 2\rangle} = k+1 \mid \omega\} \tag{30}$$

from which we derive (11) via [Conseq]: at the entry of a round the gap is 0, so $(\omega\cdot\phi_G)[acc\mapsto 0] = \omega$ and the substituted invariant is entailed by $\mathcal{R}_j$.

E.4 DP-FTRL without Subsampling

This subsection consists of two parts: §E.4.1 provides an overview of the DP-FTRL algorithm. §E.4.2 completes the derivation of the privacy guarantee of the DP-FTRL algorithm without subsampling as reported in §5.3, and compares that guarantee with the bounds of the literature.

E.4.1 Background on DP-FTRL. The DP-SGD algorithm requires knowing ahead of time the size of the dataset it is trained on, and is thereby unsuitable for online training scenarios where training takes place on the data that happen to be available at that moment, and the server has little control over which part of the data it sees and when [49]. DP-FTRL is a differentially private federated learning algorithm designed for exactly this situation. It uses no privacy amplification, so the data may be used at any round depending on their availability, and it builds on the Follow-The-Regularized-Leader (FTRL) algorithm: where SGD updates the current iterate with the newest gradient, FTRL recomputes the model at every round from the accumulated history. For a given initial model $\theta_0 \in \mathbb{R}^d$ and loss gradients $g_s = \nabla\ell(\theta_s)$, the sequence of models of FTRL is given by

$$\theta_{t+1} \triangleq \arg\min_{\theta\in\Theta} \sum_{s\le t} \langle g_s,\, \theta\rangle + \mathcal{R}(\theta) = \arg\min_{\theta\in\Theta} \left\langle \textstyle\sum_{s\le t} g_s,\, \theta \right\rangle + \mathcal{R}(\theta),$$

where $\langle\cdot,\cdot\rangle$ is the scalar product of two vectors. Intuitively, each step's model is the one that minimizes the previous steps' losses, plus a regularizer $\mathcal{R} : \mathbb{R}^d \to \mathbb{R}$. With $\mathcal{R}(\theta) = \frac{\lambda}{2}\|\theta\|^2$ this has the closed form $\theta_{t+1} = -\frac{1}{\lambda}\sum_{s\le t} g_s$. As in DP-SGD, privacy is obtained by adding Gaussian noise to the models. In the literature, different ways of noising have been considered [27, 32, 49, 56]. The naïve solution is perturbing each round's gradient, using $\tilde{g}_s \triangleq g_s + z_s$ for an independent Gaussian z_s and reading the models off the noisy prefixes $\sum_{s\le t}\tilde{g}_s = \sum_{s\le t} g_s + \sum_{s\le t} z_s$. Note that the t-th gradient sum carries t independent noises, and the error of the models grows like $\sqrt{T}$ over the run [49]. In practice, state-of-the-art mechanisms generalize the problem of computing the prefix sums to the one of computing $A\,g$, where $g \triangleq (g_0, \ldots, g_{T-1}) \in \mathbb{R}^{T\times d}$ stacks the round gradients and $A \in \mathbb{R}^{T\times T}$ is the lower-triangular matrix of ones, so that $(A\,g)_t = \sum_{s\le t} g_s$ is the t-th prefix and $\theta_{t+1} = -\frac{1}{\lambda}(A\,g)_t$. This representation allows for factorizing $A = B\,C$ for two arbitrary matrices, an *encoder* $C \in \mathbb{R}^{N\times T}$ and a *decoder* $B \in \mathbb{R}^{T\times N}$, with the number N of released aggregates free and allowed to exceed T. Under this representation, the r-th row of C gives one aggregate $(C\,g)_r = \sum_s C_{r,s}\, g_s$ as a linear combination of the gradients, and the t-th row of B describes how to combine those aggregates, $(A\,g)_t = \sum_r B_{t,r}\,(C\,g)_r$, to reconstruct the corresponding prefix sum. The algorithm perturbs the aggregate values computed through C. The factorization provides leverage on the noise usage, and achieves in practice better privacy/error trade-offs [27, 32, 56]. DP-FTRL with state-of-the-art factorizations — banded matrices [27], Buffered Linear Toeplitz matrices [56] — is used to train next-word-prediction models in Gboard, as was the tree-aggregation variant before them [76]. This family of mechanisms is implemented in JAX Privacy [47], while TensorFlow Privacy [70] implements the tree-aggregation variant.

E.4.2 Derivation. We now derive the judgment (13), stated in §5.3 for the program dpftrl of Figure 6:

$$\vdash \{D^{\langle 1\rangle} \simeq_j D^{\langle 2\rangle} \wedge \tau^{\langle 1\rangle} = \tau^{\langle 2\rangle} \mid \exp\big(\alpha(\alpha-1)\,\mathrm{sens}_j/2\sigma^2\big)\} \text{ dpftrl } \sim_\alpha \text{ dpftrl } \{\tau^{\langle 1\rangle} = \tau^{\langle 2\rangle} \mid 1\}\,. \tag{13}$$

The derivation follows the shape of the program: the main loop is handled by [While] with a budget Φ_k, the potential of the aggregates not yet released at round k; the inner loop by [While] with a budget $\Psi_{k,r}$, which also accounts for the aggregates of round k released before its step r; and each release at line 8 pays the Gaussian factor $\exp(\alpha(\alpha-1)\|\Delta_r\|^2/2\sigma^2)$ of the displacement Δ_r of the released aggregate, and the potential decreases by the same factor (item 4 below).

We use the notation of §5.3: $\Gamma_{j,t} = |\mathrm{part}(j,t)|\,\Gamma$; the set $\mathcal{U}_j$ of the $U \in \mathbb{R}^{T\times d}$ with $\|U_t\| \le \Gamma_{j,t}$ for all t; $\mathrm{sens}_j = \sup_{U\in\mathcal{U}_j} \sum_r \|(C\cdot U)_r\|^2$; the set $\mathcal{U}_j^k$ of the $U \in \mathcal{U}_j$ with $U_t = H^{\langle 2\rangle}[t] - H^{\langle 1\rangle}[t]$ for $t < k$; and the quantitative invariant

$$\Phi_k \;=\; \sup_{U\in\mathcal{U}_j^k} \exp\!\big(\alpha(\alpha-1)\textstyle\sum_{r:\mathrm{done}(r)\ge k} \|(C\cdot U)_r\|^2/2\sigma^2\big),$$

where $\mathrm{done}(r)$, the largest s with $C_{r,s} \neq 0$ (or 0 for a zero row), is the round at which the aggregate r is released. Inside the round k, after line 4 has written $H[k]$, the inner loop releases the aggregates r with $\mathrm{done}(r) = k$ one at a time. Before its step r, the slots $t \le k$ are written, so U ranges over $\mathcal{U}_j^{k+1}$. The aggregates still to be released form two groups, one for each sum below: those of a later round, $\mathrm{done}(r') > k$, and those of the round k that the inner loop has not reached yet, $\mathrm{done}(r') = k$ and $r' \ge r$. The quantitative invariant of the inner loop, which we denote by Ψ to distinguish it from the invariant Φ of the main loop, is therefore

$$\Psi_{k,r} \;\triangleq\; \sup_{U\in\mathcal{U}_j^{k+1}} \exp\!\Big(\alpha(\alpha-1)\Big(\textstyle\sum_{r':\mathrm{done}(r')>k} \|(C\cdot U)_{r'}\|^2 + \sum_{r'\ge r:\mathrm{done}(r')=k} \|(C\cdot U)_{r'}\|^2\Big)/2\sigma^2\Big).$$

The two invariants are related as follows. At $r = 0$, $\Psi_{k,0}$ sums over the same aggregates as Φ_k, those with $\mathrm{done}(r') \ge k$, but one more slot, $H[k]$, is fixed: $\Psi_{k,0}$ is Φ_k after the write of line 4. Each step of the inner loop that releases an aggregate removes it from the second sum. At $r = N$, only the aggregates with $\mathrm{done}(r') > k$ are left and the slots $t < k+1$ are fixed, so $\Psi_{k,N} = \Phi_{k+1}$. These quantities have the following properties.

(1) *Start.* $\Phi_0 = \exp(\alpha(\alpha-1)\,\mathrm{sens}_j/2\sigma^2)$, as $\mathcal{U}_j^0 = \mathcal{U}_j$ and every aggregate is still to be released.
(2) *Write.* Writing $H[k]$ can only decrease the budget: the value of $\Psi_{k,0}$ after the write is at most the value of Φ_k before it, as one more slot is fixed and $\mathcal{U}_j^{k+1} \subseteq \mathcal{U}_j^k$.
(3) *Skip and end of round.* $\Psi_{k,r+1} = \Psi_{k,r}$ if $\mathrm{done}(r) \neq k$, and $\Psi_{k,N} = \Phi_{k+1}$.
(4) *Release.* Let $\Delta_r \triangleq (C\cdot(H^{\langle 2\rangle} - H^{\langle 1\rangle}))_r$ be the actual displacement of the aggregate r. If $\mathrm{done}(r) = k$, then $C_{r,t} = 0$ for $t > k$, so $(C\cdot U)_r = \Delta_r$ for every $U \in \mathcal{U}_j^{k+1}$, and
$$\exp\!\big(\alpha(\alpha-1)\,\|\Delta_r\|^2/2\sigma^2\big)\cdot\Psi_{k,r+1} \;\le\; \Psi_{k,r}.$$
(5) *Final.* $\Phi_T \ge 1$ when $H^{\langle 2\rangle} - H^{\langle 1\rangle} \in \mathcal{U}_j$, as no aggregate is left and $\mathcal{U}_j^T$ is not empty.

The derivation uses the logical invariant

$$\mathcal{R}_j \;\triangleq\; D^{\langle 1\rangle} \simeq_j D^{\langle 2\rangle} \wedge \theta^{\langle 1\rangle} = \theta^{\langle 2\rangle} \wedge Y^{\langle 1\rangle} = Y^{\langle 2\rangle} \wedge \tau^{\langle 1\rangle} = \tau^{\langle 2\rangle} \wedge H^{\langle 2\rangle} - H^{\langle 1\rangle} \in \mathcal{U}_j \tag{31}$$

which asserts adjacency of the datasets, equality of the models, of the aggregate vectors and of the transcripts, and that the difference of the histories is one of those that j can cause. For presentation's sake, we also introduce the two families of conditions $\mathcal{I}_k \triangleq \mathcal{R}_j \wedge t^{\langle 1\rangle} = t^{\langle 2\rangle} = k$ and $\mathcal{L}_{k,r} \triangleq \mathcal{I}_k \wedge r^{\langle 1\rangle} = r^{\langle 2\rangle} = r$ which add the counters of the two loops. Both decide the guards: $\mathcal{I}_k$ for $k < T$ and $\mathcal{L}_{k,r}$ for $r < N$ force them true, while $\mathcal{I}_T$ and $\mathcal{L}_{k,N}$ force them false.

Following the same recipe as the previous case studies, the crucial step for deriving (13) is using [While], with the budgets Φ_k, to derive, for every $k < T$, the judgment below. As in Appendix E.3, the budgets depend on the state, so the counted [While-Prod] does not apply and [While] is used directly, here for both loops:

$$\vdash \{\mathcal{I}_k \mid \Phi_k\}\ \mathrm{round} \sim_\alpha \mathrm{round}\ \{\mathcal{I}_{k+1} \mid \Phi_{k+1}\} \tag{32}$$

where round is the body of the main loop of Figure 6.

The derivation of (32) proceeds as follows: by backward (weakest pre-condition) reasoning through [Assn] and [Release], we push the post-condition $\mathcal{I}_{k+1}$ through the block tail of the trailing statements at lines 11–12, yielding the intermediate judgment

$$\vdash \{\mathcal{L}_{k,N} \mid \Psi_{k,N}\}\ \mathsf{tail} \sim_\alpha \mathsf{tail}\ \{\mathcal{I}_{k+1} \mid \Phi_{k+1}\}\,. \tag{33}$$

These statements consist of assignments and a release only, and do not write H, so each step leaves the potential unchanged, and $\Psi_{k,N} = \Phi_{k+1}$ by property 3. Specifically, the prefix sum $\hat{s} \leftarrow (B \cdot Y)_t$ is a function of the aggregate vector and of the counter, which $\mathcal{L}_{k,N}$ equates, so the two $\hat{s}$ are equal; the leader step $\theta \leftarrow -(1/\lambda)\,\hat{s}$ then equates the models; `release` θ appends equal values to equal transcripts, so [Release] preserves $\tau^{\langle 1\rangle} = \tau^{\langle 2\rangle}$; and the increment of t turns $\mathcal{I}_k$ into $\mathcal{I}_{k+1}$.

We now turn our attention to the inner loop relloop of Figure 6, where we establish the following judgment

$$\vdash \{\mathcal{L}_{k,0} \mid \Psi_{k,0}\}\ \mathsf{relloop} \sim_\alpha \mathsf{relloop}\ \{\mathcal{L}_{k,N} \mid \Psi_{k,N}\} \tag{34}$$

again via Rule [While], this time applied to the family of judgments on the body relstep of the inner loop, the guarded release of line 8 together with the increment of r:

$$\vdash \{\mathcal{L}_{k,r} \mid \Psi_{k,r}\}\ \mathsf{relstep} \sim_\alpha \mathsf{relstep}\ \{\mathcal{L}_{k,r+1} \mid \Psi_{k,r+1}\} \qquad \text{for every } r < N. \tag{35}$$

The guard $\mathrm{done}(r) = t$ is a function of the two counters, which $\mathcal{L}_{k,r}$ equates; it therefore agrees on the two runs, and we apply [If] to derive the judgment

$$\begin{array}{l}\vdash \{\mathcal{L}_{k,r} \mid \Psi_{k,r}\}\\ \texttt{if}\ \mathrm{done}(r) = t\ \texttt{then}\ z \xleftarrow{\$} \mathcal{N}(0,\sigma^2);\, Y[r] \leftarrow (C\cdot H)_r + z\\ \sim_\alpha\ \texttt{if}\ \mathrm{done}(r) = t\ \texttt{then}\ z \xleftarrow{\$} \mathcal{N}(0,\sigma^2);\, Y[r] \leftarrow (C\cdot H)_r + z\\ \{\mathcal{L}_{k,r} \mid \Psi_{k,r+1}\}\end{array} \tag{36}$$

When the guard is false, the corresponding judgment can be trivially derived, as $\Psi_{k,r+1} = \Psi_{k,r}$ by property 3 and [Skip] relates the two `skip` of the omitted `else` branch. On the true branch, our goal is to derive

$$\begin{array}{l}\vdash \{\mathcal{L}_{k,r} \wedge \mathrm{done}(r) = k \mid \Psi_{k,r}\}\\ z \xleftarrow{\$} \mathcal{N}(0,\sigma^2);\, Y[r] \leftarrow (C\cdot H)_r + z \sim_\alpha z \xleftarrow{\$} \mathcal{N}(0,\sigma^2);\, Y[r] \leftarrow (C\cdot H)_r + z\\ \{\mathcal{L}_{k,r} \mid \Psi_{k,r+1}\}\end{array} \tag{37}$$

where the guard has fixed $\mathrm{done}(r) = k$, and we proceed as follows.

The sampling is discharged by the derived [Samp-Gap], applied at the displacement

$$\Delta_r \;=\; \big(C\cdot(H^{\langle 2\rangle} - H^{\langle 1\rangle})\big)_r \;=\; \sum_{t<T} C_{r,t}\,\big(H^{\langle 2\rangle}[t] - H^{\langle 1\rangle}[t]\big)\,, \tag{38}$$

which pairs the noise z on the left with $z - \Delta_r$ on the right. We take the post-budget $\Psi_{k,r+1}$, which does not depend on z, so that the weighted Hellinger integral of the premise is $\Psi_{k,r+1}$ times the unweighted one, and obtain

$$\begin{array}{l}\vdash \{\mathcal{L}_{k,r} \mid \exp\!\big(\alpha(\alpha-1)\|\Delta_r\|^2/2\sigma^2\big)\cdot \Psi_{k,r+1}\}\\ z \xleftarrow{\$} \mathcal{N}(0,\sigma^2) \sim_\alpha z \xleftarrow{\$} \mathcal{N}(0,\sigma^2)\\ \{\mathcal{L}_{k,r} \wedge \big((C\cdot H)_r + z\big)^{\langle 1\rangle} = \big((C\cdot H)_r + z\big)^{\langle 2\rangle} \mid \Psi_{k,r+1}\}\end{array} \tag{39}$$

On the assignment, we apply [Assn], which leaves $\Psi_{k,r+1}$ unchanged as it does not depend on Y, yielding:

$$\begin{array}{l}\vdash \{\mathcal{L}_{k,r} \wedge \big((C\cdot H)_r + z\big)^{\langle 1\rangle} = \big((C\cdot H)_r + z\big)^{\langle 2\rangle} \mid \Psi_{k,r+1}\}\\ Y[r] \leftarrow (C\cdot H)_r + z \sim_\alpha Y[r] \leftarrow (C\cdot H)_r + z\\ \{\mathcal{L}_{k,r} \mid \Psi_{k,r+1}\}\end{array} \tag{40}$$

[Seq] composes the two judgments, yielding

$$\begin{array}{l}\vdash \{\mathcal{L}_{k,r} \mid \exp\big(\alpha(\alpha-1)\|\Delta_r\|^2/2\sigma^2\big)\cdot \Psi_{k,r+1}\}\\ z \xleftarrow{\$} \mathcal{N}(0,\sigma^2);\, Y[r] \leftarrow (C\cdot H)_r + z \sim_\alpha z \xleftarrow{\$} \mathcal{N}(0,\sigma^2);\, Y[r] \leftarrow (C\cdot H)_r + z\\ \{\mathcal{L}_{k,r} \mid \Psi_{k,r+1}\}\end{array} \tag{41}$$

We derive (37) from (41) by [Conseq] and the release property 4, which applies as $\mathrm{done}(r) = k$. This completes the proof of (36), which we combine with the [Assn] of the increment,

$$\vdash \{\mathcal{L}_{k,r} \mid \Psi_{k,r+1}\}\ r \leftarrow r+1 \sim_\alpha r \leftarrow r+1\ \{\mathcal{L}_{k,r+1} \mid \Psi_{k,r+1}\}, \tag{42}$$

via [Seq], obtaining (35) and thereby, by the application of [While] anticipated above, (34).

We finally use [Seq] to combine (34) and (33) and [Assn] to perform the backwards substitutions of the gradient sum $\mathrm{gsum} \triangleq \sum_{i<|D|} \mathrm{part}(i,k)\,\mathrm{clip}_\Gamma(\nabla\ell(\theta, D[i]))$ of line 4 and of the reset $r \leftarrow 0$. This yields the judgment

$$\vdash \{\mathcal{L}_{k,0}[r\mapsto 0, H[k]\mapsto \mathrm{gsum}] \mid \Psi_{k,0}[H[k]\mapsto \mathrm{gsum}]\}\ \mathrm{round} \sim_\alpha \mathrm{round}\ \{\mathcal{I}_{k+1} \mid \Phi_{k+1}\}, \tag{43}$$

From (43) we easily derive (32) via [Conseq] by observing that the substituted invariant is entailed by $\mathcal{I}_k$, and that the substituted budget is at most Φ_k by the write property 2. The substitution touches two conjuncts of $\mathcal{L}_{k,0}$ and no other. The counter conjunct becomes trivially true, and the conjunct $H^{\langle 2\rangle} - H^{\langle 1\rangle} \in \mathcal{U}_j$ becomes, at the slot k, a bound on the two gradient sums, $\left\|\mathrm{gsum}^{\langle 1\rangle} - \mathrm{gsum}^{\langle 2\rangle}\right\| \le \Gamma_{j,k}$. Since the slots $t \neq k$ are left as they are, the entailment reduces to the single obligation

$$\mathcal{I}_k \ \models\ \left\|\mathrm{gsum}^{\langle 1\rangle} - \mathrm{gsum}^{\langle 2\rangle}\right\| \le \Gamma_{j,k},$$

which the $|D|$ summands of gsum discharge. Those at $i \neq j$ contribute the same vector to both sides, the models being equal and the records agreeing off j, so they cancel; the one at j contributes $\mathrm{part}(j,k)$ times a clipped gradient in the run that holds the record and, since $D[j] = \bot$ in the other and $\nabla\ell(\theta,\bot) = 0$, the zero vector there. Writing D for the dataset of the run that holds j, the norm of the difference is the same in either order. What is left is one summand:

$$\left\|\mathrm{gsum}^{\langle 1\rangle} - \mathrm{gsum}^{\langle 2\rangle}\right\| \ =\ \left\|\mathrm{part}(j,k)\,\mathrm{clip}_\Gamma\big(\nabla\ell(\theta, D[j])\big)\right\| \ \le\ |\mathrm{part}(j,k)|\,\Gamma \ =\ \Gamma_{j,k},$$

the inequality being the clipping bound. So no invariant over the records is needed, the gradient sum entering the privacy argument through this bound alone. This completes the derivation of (32); [While] takes that family to the whole run, concluding at $\mathcal{I}_T$ with post-budget 1, as $\Phi_T \ge 1$ by property 5. Four [Assn] for the initialization of line 2 establish $\mathcal{I}_0$, and leave Φ_0 unchanged, as it fixes no slot. One [Conseq] then rewrites the pre-budget by the start property 1 and weakens the post-condition $\mathcal{I}_T$ to its transcript conjunct $\tau^{\langle 1\rangle} = \tau^{\langle 2\rangle}$, which gives (13) and, by Proposition 4.4, the individual bound.

Comparison with previous bounds. Depending on the choice of the factorization and of the participation schedule, Figure 6 generalizes, up to minor cosmetic differences, several variants of DP-FTRL from the literature: tree aggregation [49], where the encoder's rows are the indicators of the dyadic intervals and each record takes part in exactly one round; the square lower-triangular mechanism of Denisov et al. [32], under the same single participation (they draw the whole $N \times d$ noise matrix up front, where we draw each row at the round of its release); and multi-epoch factorizations [28], which correspond to arbitrary participation schedules part. Let $\mathrm{gap}_{j,r} \triangleq \sum_{t<T} |C_{r,t}|\,\Gamma_{j,t}$. By the triangle inequality, $\|(C\cdot U)_r\| \le \mathrm{gap}_{j,r}$ for every $U \in \mathcal{U}_j$, so $\mathrm{sens}_j \le \sum_r \mathrm{gap}_{j,r}^2$ 🅛. The following proposition evaluates $\sum_r \mathrm{gap}_{j,r}^2$ at these instances.

Proposition E.5 (Evaluating the sensitivity of the encoder 🅛). *Let $\Gamma \ge 0$ and let the record j participate with the $0/1$ schedule of a set P_j of rounds,* $\mathrm{part}(j,\cdot) = \mathbf{1}_{P_j}$. *Then:*

(1) *if P_j is a singleton $\{s\}$, then, for every C,*

$$\sum_r \mathrm{gap}_{j,r}^2 \;=\; \|C_{\cdot,s}\|^2\,\Gamma^2\,, \qquad \|C_{\cdot,s}\|^2 \;\triangleq\; \sum_r C_{r,s}^2\,.$$

(2) *if C is b-banded — every row's nonzero entries lie within b consecutive columns — and P_j is b-min-separated (any two of its rounds are at least b apart), then*

$$\sum_r \mathrm{gap}_{j,r}^2 \;=\; \sum_{s\in P_j} \|C_{\cdot,s}\|^2\,\Gamma^2\,,$$

since each row meets P_j in at most one nonzero column, so the sign of C plays no role.

Kairouz et al. [49, Theorem 4.1] study DP-FTRL with tree aggregation. Every record takes part in exactly one round, with weight 1, and the encoder C has one row per node of a complete binary tree whose leaves are the $T = 2^h$ rounds: the row of a node is 1 at the rounds below that node and 0 elsewhere. In our notation, they prove that the algorithm is (α, ρ)-RDP, where

$$\rho \;=\; \frac{\alpha\,(h+1)\,\Gamma^2}{2\sigma^2}\,.$$

The factor $h + 1$ is the number of nodes on the path from a leaf to the root, that is, the number of aggregates to which one round contributes.

Our bound implies theirs. If the record j takes part in the round s, claim (1) of Proposition E.5 gives $\mathrm{sens}_j \le \Gamma^2\|C_{\cdot,s}\|^2$, and the column s of C has one entry 1 per node on the path from the leaf s to the root, so $\|C_{\cdot,s}\|^2 = h + 1$ ∎. By (13), every record is thus (α, ρ)-individually RDP, and Lemma 3.7 gives that the algorithm is (α, ρ)-RDP.

The conditions compare as follows. Their theorem allows any number n of rounds, with $\lceil \log_2(n+1) \rceil$ in place of $h + 1$; we consider complete trees, where the two coincide. Their noise is specified relative to Γ, as a standard deviation σ_0 with $\sigma = \Gamma\sigma_0$. Single participation is part of their setting, whereas our derivation covers every schedule.

Denisov et al. [32, Theorem 4.1] study DP-FTRL with an arbitrary factorization of square lower-triangular matrices, where every record takes part in exactly one round, with weight 1. They prove that the algorithm has the guarantee of a Gaussian mechanism of variance σ^2 applied to a function whose ℓ_2 sensitivity is $\Gamma \max_s \|C_{\cdot,s}\|$, the largest norm of a column of C, also when the gradients are chosen adaptively. This Gaussian mechanism is (α, ρ)-RDP, where

$$\rho \;=\; \frac{\alpha\,\Gamma^2}{2\sigma^2}\,\max_s\,\|C_{\cdot,s}\|^2\,.$$

The column s of C lists the coefficients with which the round s enters the aggregates, so $\Gamma\|C_{\cdot,s}\|$ is the largest distance between the noiseless aggregates of two adjacent runs whose differing record takes part in the round s.

Our bound implies theirs. If the record j takes part in the round s, claim (1) of Proposition E.5 gives $\mathrm{sens}_j \le \Gamma^2\|C_{\cdot,s}\|^2 \le \Gamma^2 \max_{s'} \|C_{\cdot,s'}\|^2$. By (13), every record is thus (α, ρ)-individually RDP, and Lemma 3.7 gives that the algorithm is (α, ρ)-RDP.

The conditions compare as follows. Their encoder is square and lower-triangular, and they draw the whole noise matrix before the first round; ours is any $N \times T$ matrix, and we draw the noise of each aggregate at the round of its release. They state their result for (ϵ, δ)-differential privacy, through the Gaussian mechanism; we state it for RDP. Each record is charged for the column of its own round rather than for the largest column.

Choquette-Choo et al. [28] study the setting where every record takes part, with weight 1, in a fixed set of rounds, its *participation pattern*, chosen from a known family $\mathfrak{P}$ of sets of rounds. In our notation, $\mathrm{part}(j, t)$ is 1 if $t \in P_j$ and 0 otherwise, where $P_j \in \mathfrak{P}$ is the pattern of the record j.

In their Theorem 2.2 and Equation (3), they prove that, if all the entries of $C^\top C$ are nonnegative, the algorithm is (α, ρ)-RDP with respect to $\bigcup_j \simeq_j$, also when the gradients are chosen adaptively, where

$$\rho \;=\; \frac{\alpha\,\Gamma^2}{2\sigma^2}\;\max_{P\in\mathfrak{P}}\left\|\sum_{t\in P} C_{\cdot,t}\right\|^2$$

and $C_{\cdot,t}$ is the column t of C. Here $\Gamma \,\|\, \sum_{t\in P} C_{\cdot,t}\|$ is the distance between the noiseless aggregates of two adjacent runs when the differing record contributes the same vector of norm Γ at every round of the pattern P; under the sign condition, no other contributions give a larger distance. They state the result as the guarantee of a Gaussian mechanism of variance σ^2, whose RDP is (α, ρ).

Our bound implies theirs. For every $U \in \boldsymbol{\mathcal{U}}_j$, expanding the norm row by row gives $\|C \cdot U\|^2 = \sum_{t,t'} (C^\top C)_{t,t'} \langle U_t, U_{t'}\rangle \le \Gamma^2 \sum_{t,t'\in P_j} |(C^\top C)_{t,t'}|$, by Cauchy–Schwarz and $\|U_t\| \le \Gamma_{j,t}$. When all the entries of $C^\top C$ are nonnegative, this bound is attained: taking U_t equal to one vector of norm Γ for $t \in P_j$, and $U_t = 0$ otherwise, makes every cross term equal to $\Gamma^2 (C^\top C)_{t,t'}$, so

$$\mathrm{sens}_j \;=\; \Gamma^2 \sum_{t,t'\in P_j} (C^\top C)_{t,t'} \;=\; \Gamma^2 \left\|\sum_{t\in P_j} C_{\cdot,t}\right\|^2 \;\le\; \Gamma^2 \max_{P\in\mathfrak{P}} \left\|\sum_{t\in P} C_{\cdot,t}\right\|^2 .$$

By (13), every record is thus (α, ρ)-individually RDP, and Lemma 3.7 gives that the algorithm is (α, ρ)-RDP.

The conditions compare as follows. The sign condition on $C^\top C$ only serves to compute the worst case in closed form. Without it, our bound still holds, and sens_j is still the quantity that their analysis charges for the pattern P_j (their Definition 1): the largest distance between the noiseless aggregates over all the contributions of the record. Our weights $\mathrm{part}(j, t)$ are arbitrary reals rather than $0/1$, and each record is charged for its own pattern rather than for the worst one of $\mathfrak{P}$. They obtain adaptivity by a separate argument (their Theorem 2.2); for us it is part of the program. Besides removing a record, as $\simeq_j$ does, their adjacency also allows replacing its contributions, which we do not consider.

E.5 DP-FTRL with Subsampling

This subsection covers the Amplified Banded Matrix Factorization algorithm of Choquette-Choo et al. [27], deferred from §5.3, whose notation for the matrix mechanism we keep throughout. Here the encoder is square, $N = T$, and the aggregate $Y[t]$ is formed and released at round t.

To confine the influence of a single record to a few aggregates, the algorithm constrains both the encoder and the participations. The encoder is *b-banded*: $C_{t,p} = 0$ unless $p \le t < p + b$, so that each of its rows reads only the b most recent rows of the history. Each record is eligible only at rounds that are at least b apart. The two conditions together imply that a record taking part in the round p influences only the b aggregates $Y[p], \ldots, Y[p + b - 1]$, and that the aggregates influenced by two participations of the same record are disjoint. Finally, the algorithm boosts its privacy guarantees by subsampling: an eligible record takes part in a round only with probability q.

Our instance of DP-FTRL is shown in Figure 10. Its parameters are: the dataset D, a collection of elements from $\mathbb{R}^d_\perp$, as in §5.1; the number of rounds T; the regularization parameter $\lambda \in \mathbb{R}_{>0}$; the norm $\Gamma \in \mathbb{R}_{>0}$ to which the gradients are clipped; the standard deviation $\sigma \in \mathbb{R}_{>0}$ of the noise; the sampling rate $q \in (0, 1)$; the bandwidth b; and a factorization $A = BC$ of the prefix-sum matrix into a b-banded encoder C and a decoder B. The algorithm begins at line 2 by initializing the model θ, the history array H, the aggregates array Y and the round counter to 0. Then, it iterates T rounds of training: each of them opens with the block Pois^b_q of line 4, which scans the dataset and selects the records eligible at that round (those whose index i satisfies $i \bmod b = t \bmod b$). This test enforces that two participations of the same record are at least b rounds apart. It then tosses a coin of bias q

1: **input** dataset D; factorization $A = BC$; parameters $T, \lambda, \Gamma, \sigma, q, b$.
2: $\theta \leftarrow 0; H \leftarrow 0; Y \leftarrow 0; t \leftarrow 0$
3: `while` $t < T$ `do`
4: $\quad \mathsf{Pois}_q^b$
5: $\quad H[t] \leftarrow acc; z \xleftarrow{\$} \mathcal{N}(0, \sigma^2); Y[t] \leftarrow (C \cdot H)_t + z$
6: $\quad \hat{s} \leftarrow (B \cdot Y)_t; \theta \leftarrow -(1/\lambda)\,\hat{s}$; `release` θ
7: $\quad t \leftarrow t + 1$
8: `od`

where Pois_q^b is

$i \leftarrow 0; acc \leftarrow 0$
`while` $i < |D|$ `do`
$\quad$ `if` $i \bmod b = t \bmod b$ `then`
$\quad\quad$ `local` u `in` $\big(u \xleftarrow{\$} \mathsf{Bern}(q);$
$\quad\quad acc \leftarrow acc + u\,\mathrm{clip}_\Gamma(\nabla \ell(\theta, D[i]))\big)$
$\quad i \leftarrow i + 1$
`od`

Fig. 10. Amplified Banded Matrix Factorization.

for each of them and accumulates the clipped gradients of the selected ones into *acc*. Line 5 then writes the history slot $H[t] \leftarrow acc$ and forms the noisy aggregate $Y[t]$, and line 6 decodes the model from $B \cdot Y$ and releases it.

Overview of the derivation. As with DP-SGD (§5.1.1), the privacy guarantees of DP-FTRL stem from the same underlying mechanisms (Gaussian noise injection and subsampling); therefore the corresponding privacy bounds can be understood analogously. In DP-SGD, for a fixed order $\alpha > 1$, the privacy budget of a single round is a function $\Upsilon_\alpha(q, \Gamma, v)$ of the subsampling rate q, the clipping norm Γ that bounds the maximum displacement of the model due to a record, and the noise variance v; as each record takes part in every round, the RDP budget of the entire training is T times that of a single learning step, making DP-SGD $(\alpha,\ T \log \Upsilon_\alpha(q,\Gamma,v)/\alpha-1)$-RDP. For DP-FTRL, the bound is obtained in a similar way: the algorithm is $\big(\alpha,\ k \log \Upsilon_\alpha(q,\Gamma c,\sigma^2)/\alpha-1\big)$-individually RDP, where $k \triangleq \lceil T/b \rceil$ caps the number of times a record is eligible for training, and Γc is an upper bound on the displacement that a record can induce on the aggregates that it affects. Specifically, a record taking part at the round p contributes a clipped gradient of norm at most Γ to the history slot $H[p]$, which the encoder reads with the weights of its p-th column, so the displacement as a whole has norm at most $\Gamma\, c_p$, where $c_p \triangleq \sqrt{\sum_{r<\min(b,T-p)} C_{p+r,p}^2}$ is the norm of that column, restricted to the rows of the run. Taking $c \triangleq \max_p c_p$ gives an upper bound over all the columns of C. With Proposition 4.4 at hand, proving $\big(\alpha,\ k \log \Upsilon_\alpha(q,\Gamma c,\sigma^2)/\alpha-1\big)$-individual RDP of DP-FTRL reduces to deriving the following judgment

$$\vdash \{D^{\langle 1\rangle} \simeq_j D^{\langle 2\rangle} \wedge \tau^{\langle 1\rangle} = \tau^{\langle 2\rangle} \mid \textstyle\prod_{t\in P_j} \Upsilon_\alpha(q, \Gamma c_t, \sigma^2)\}\ \mathsf{dpftrl}_{\mathrm{amp}} \sim_\alpha \mathsf{dpftrl}_{\mathrm{amp}}\ \{\tau^{\langle 1\rangle} = \tau^{\langle 2\rangle} \mid 1\}\,. \qquad (44)$$

Here, for a record j, $P_j \triangleq \{\, t < T \mid t \equiv j \,(\mathrm{mod}\ b)\,\}$ is the set of rounds at which j can be drawn into the gradient computation, and it is easy to verify that $|P_j| \le k$. The adjacency $\simeq_j$ is defined as in §5.1.1. In the derivation, it is crucial to apply two program identities via [Equiv]. The first identity unrolls the main loop and groups its rounds into *blocks* of b consecutive rounds, one opened by each participation of j. The second one is a lazy vs. eager sampling replacement: inside a block, the coin sampled for j and the b noises drawn in that block are replaced by a single draw

$$w \xleftarrow{\$} \mathsf{Bern}(q) \otimes \mathcal{N}(0, \sigma^2 I) \qquad (45)$$

[Samp-Dec]
$$\frac{\forall s = ((m_1,\tau_1),(m_2,\tau_2)) \in \mathcal{R}.\ F_s \text{ is Markov and } \mathcal{S}[x{\mapsto}v, y{\mapsto}v'](s) \text{ for } \nu_s\text{-a.e. } v \text{ and } F_s(v)\text{-a.e. } v' \quad \forall s = ((m_1,\tau_1),(m_2,\tau_2)) \in \mathcal{R}.\ \mathbf{dlet}\ v \leftarrow \nu_s\ \mathbf{in}\ F_s(v) = [\![S_2]\!](m_2) \text{ and } \mathrm{H}_\alpha^{\bar\psi_s}([\![S_1]\!](m_1) \parallel \nu_s) \le \phi(s) \quad \text{where } \bar\psi_s(v) \triangleq \mathbb{E}_{v' \leftarrow F_s(v)}[\psi[x{\mapsto}v, y{\mapsto}v'](s)]}{\vdash \{\mathcal{R} \mid \phi\}\ x \xleftarrow{\$} S_1 \sim_\alpha y \xleftarrow{\$} S_2\ \{\mathcal{S} \mid \psi\}}$$

Fig. 11. The derived sampling rule [Samp-Dec]. ∧

at the head of the block. All the following sampling statements are modified to read their noise from w instead of recomputing it. Like the rewriting in §5.1.1, these rewritings unify in a single sampling statement the subsampling of j and the Gaussian noise that is added to its gradient. Without unifying the sampling statements, one would have to analyse them via distinct applications of sampling rules, obtaining looser bounds, as described in §5.1.2. Instead, our proof uses a single sampling rule on (45). Specifically, the proof uses Rule [Samp-Dec] to set the noises of the two runs at the distance that equates their models after they are summed at line 6, so that the values they release are identical. That distance has norm 0 when the coin of j is 0, as the two runs then compute the same gradient sums, and has strictly positive norm when the coin comes up 1. Rule [Samp-Dec] is general enough to set a displacement between the noises that depends on the outcome of the other marginal of the sampling.

Rule [Samp-Dec]. The rule, shown in Figure 11, is derived from Rule [Samp-Lift]. For every pair of states s, it asks for a decomposition of the sampling expression on the right in terms of a sub-probability measure $\nu_s \in \mathsf{D}(\mathsf{Val})$ and a Markov kernel $F_s \colon \mathsf{Val} \to \mathsf{D}(\mathsf{Val})$. Given the decomposition, the side-conditions of the rule impose that: (i) for ν_s-almost every value v, the post-condition $\mathcal{S}$ relates v to $F_s(v)$-almost every v'; and (ii) the pre-budget $\phi(s)$ bounds the Hellinger integral between the law of the left statement and the witness ν_s. Note that the cost only depends on the witness, so a decomposition that brings ν_s closer to the left sampling expression yields a smaller budget. Based on the choice of the decomposition, the rule subsumes different combinations of privacy budget and logical post-condition: [Samp-Gap] is the instance in which $F_s(v)$ is the Dirac measure at $v - G(s)$, and [Samp] the one in which it is the Dirac measure at v.

Derivation. In this section we prove individual RDP of the algorithm in Figure 10. Before starting with the proof, we make some preliminary observations that allow us to define the notion of *block*. Note that a record j may be used to compute the gradient exactly at the rounds P_j defined above, which we enumerate as $t_0 < \cdots < t_{K-1}$. Consecutive entries are exactly b apart; the rounds $p, \dots, \min(p+b, T) - 1$ with $p = t_l$ form the *block* l. Every block has b rounds except possibly the last one, which the end of the run may truncate to the $T - t_{K-1} \le b$ rounds $t_{K-1}, \dots, T-1$. The blocks therefore abut, and the only rounds outside every block are the t_0 initial ones.

Coming back to our proof, the goal is to derive

$$\vdash \{D^{\langle 1\rangle} \simeq_j D^{\langle 2\rangle} \wedge \tau^{\langle 1\rangle} = \tau^{\langle 2\rangle} \mid \textstyle\prod_{l<K} \Upsilon_\alpha(q, \Gamma c_{t_l}, \sigma^2)\}\ \mathsf{dpftrl}_{\mathrm{amp}} \sim_\alpha \mathsf{dpftrl}_{\mathrm{amp}}\ \{\tau^{\langle 1\rangle} = \tau^{\langle 2\rangle} \mid 1\}, \tag{46}$$

where $c_p \triangleq \sqrt{\sum_{r<\min(b,T-p)} C^2_{p+r,p}}$ is the norm of the column p of the encoder restricted to the block where it starts contributing to the gradient sum (cut off at the end of the run).

In this proof, the record j contributes not only to the release of the round at which it is subsampled, but to all the b releases of the block it opens; charging it round by round would cost one subsampled-Gaussian factor per training step rather than per block, so the derivation performs the two rearrangements anticipated above. Both rearrangements are derivable in the program

equivalence calculus of Appendix C. The first application of [Equiv] is

$$\texttt{while } t < T \texttt{ do round od} \;\equiv_{t=0}\; \mathsf{round}^{t_0}; \mathsf{block}_{t_0}; \cdots; \mathsf{block}_{t_{K-1}}, \tag{47}$$

an instance of [E-Count] on the round counter t at $k = T$, where round^n is the n-fold sequencing of the loop body and $\mathsf{block}_p \triangleq \mathsf{round}^{\min(b, T-p)}$ is the block starting at round p, so that block l is block_{t_l}. The second replacement occurs inside the blocks, and consists of moving their sampling style from lazy to eager:

$$\mathsf{block}_p \;\equiv_{w=d_w}\; \texttt{local } w \texttt{ in } \big(\mathsf{draw}; \mathsf{body}_p\big), \qquad \mathsf{draw} \triangleq\; w \xleftarrow{\$} \mathsf{Bern}(q) \otimes \mathcal{N}(0, \sigma^2 I) \quad \text{on } \{0,1\} \times (\mathbb{R}^d)^b. \tag{48}$$

Here body_p runs the $\min(b, T-p)$ rounds of block_p: in the first round, the subsampling loop is modified to take the coin corresponding to the j-th record from w, and in every round the noise is taken from w (a truncated last block reads the first $T-p$ of the b noises of w; the others are drawn and never read):

$$\begin{aligned}
\mathsf{body}_p &\triangleq \widetilde{\mathsf{round}}_0; \widetilde{\mathsf{round}}_1; \cdots; \widetilde{\mathsf{round}}_{\min(b, T-p)-1}, \\
\widetilde{\mathsf{round}}_0 &\triangleq \widetilde{\mathsf{Pois}}; \mathsf{tail}_0, \qquad \widetilde{\mathsf{Pois}} \triangleq\; i \leftarrow 0; acc \leftarrow 0; \mathsf{step}^j; \mathsf{step}_j; \mathsf{step}^{|D|-1-j}, \\
\widetilde{\mathsf{round}}_r &\triangleq \mathsf{Pois}^b_q; \mathsf{tail}_r \qquad (1 \le r < b), \\
\mathsf{tail}_r &\triangleq H[t] \leftarrow acc; z \leftarrow w.2[r]; Y[t] \leftarrow (C \cdot H)_t + z; \\
&\quad\; \hat{s} \leftarrow (B \cdot Y)_t; \theta \leftarrow -(1/\lambda)\, \hat{s}; \texttt{release } \theta; t \leftarrow t+1,
\end{aligned} \tag{49}$$

where step^n is the n-fold sequencing of step, one record step of the subsampling loop of Figure 10,

$$\begin{aligned}
\mathsf{step} \;\triangleq\; &\big(\texttt{if } i \bmod b = t \bmod b \texttt{ then} \\
&\quad \texttt{local } u \texttt{ in } \big(u \xleftarrow{\$} \mathsf{Bern}(q); acc \leftarrow acc + u\, \mathsf{clip}_\Gamma(\nabla \ell(\theta, D[i]))\big)\big); i \leftarrow i+1,
\end{aligned} \tag{50}$$

and step_j is step with the coin toss $u \xleftarrow{\$} \mathsf{Bern}(q)$ replaced by the read $u \leftarrow w.1$, where $w.1$ and $w.2$ are the coin and noise components of w.

The pieces of (47) are composed by [Seq] through the family of invariants

$$\begin{aligned}
\mathcal{R}_t \;\triangleq\; & D^{\langle 1\rangle} \simeq_j D^{\langle 2\rangle} \wedge \theta^{\langle 1\rangle} = \theta^{\langle 2\rangle} \wedge Y^{\langle 1\rangle} = Y^{\langle 2\rangle} \wedge \tau^{\langle 1\rangle} = \tau^{\langle 2\rangle} \wedge t^{\langle 1\rangle} = t^{\langle 2\rangle} = t \\
& \wedge \forall p.\; t < p + b \implies H^{\langle 1\rangle}[p] = H^{\langle 2\rangle}[p] \wedge w^{\langle 1\rangle} = w^{\langle 2\rangle} = d_w.
\end{aligned} \tag{51}$$

The scratch variable w is fresh, so it is at d_w initially; the rounds outside the blocks do not touch it, and `local` resets it at the end of each block, so the last conjunct is preserved and provides the precondition of (48). Unlike the general matrix case, where we can only bound the distance between the histories (31), here we can impose a stronger invariant: that the b most recent history slots are identical. This is because the records are involved at rounds that are at least b apart, and since the adjacency relation states that only j can differ, we can assume identity of the previous b history entries. The two families of judgments to derive are then, for the rounds outside the blocks and for the blocks,

$$\vdash \{\mathcal{R}_t \mid 1\}\ \mathsf{round} \sim_\alpha \mathsf{round}\ \{\mathcal{R}_{t+1} \mid 1\} \qquad (t \notin \textstyle\bigcup_l [t_l, t_l + b)), \tag{52}$$

$$\vdash \{\mathcal{R}_p \mid \Upsilon_\alpha(q, \Gamma c_p, \sigma^2)\}\ \mathsf{block}_p \sim_\alpha \mathsf{block}_p\ \{\mathcal{R}_{\min(p+b,\, T)} \mid 1\} \qquad (p = t_l,\ l < K). \tag{53}$$

By combining these judgments via [Seq], and tacit applications of [Conseq] and [Scale], we precisely get (46).

We begin with (52), deriving first the judgment of the round's subsampling loop, then pushing it through the tail:

$$\vdash \{\mathcal{R}_t \mid 1\}\ \mathsf{Pois}^b_q \sim_\alpha \mathsf{Pois}^b_q\ \{\mathcal{R}_t \wedge acc^{\langle 1\rangle} = acc^{\langle 2\rangle} \mid 1\} \qquad (t \notin P_j), \tag{54}$$

by [While-Prod] at the family

$$\mathcal{J}_i \triangleq \mathcal{R}_t \wedge i^{\langle 1\rangle} = i^{\langle 2\rangle} = i \wedge acc^{\langle 1\rangle} = acc^{\langle 2\rangle}, \tag{55}$$

whose premise is established by meta-logical case analysis on the record i. For $i \neq j$ with $i \equiv t \pmod b$, the two coins are related at equal values by [Samp] at budget 1, and the records agree in the two datasets, the models being equal under $\mathcal{R}_t$, so the step adds the same vector to both accumulators; when $i \not\equiv t \pmod b$, the test is false on both sides, by [If] and [Skip]. For $i = j$ the test is also false, since $t \notin P_j$. This concludes the derivation of (54). We conclude the derivation of (52) by observing that from (54) the two accumulators are equal, so [Assn] writes the slot $H[t] \leftarrow acc$ equally on both sides, the noise is related at equal values by [Samp] at budget 1, and the released row agrees because the release reads live slots only,

$$(C \cdot H)_t^{\langle 1\rangle} - (C \cdot H)_t^{\langle 2\rangle} = \sum_{p \le t} C_{t,p} \left(H^{\langle 1\rangle}[p] - H^{\langle 2\rangle}[p]\right) = 0, \tag{56}$$

by combining bandedness and $\mathcal{R}_t$'s conjunct on H. [Assn] and [Release] then equate the model and extend equal transcripts, and the increment of t gives $\mathcal{R}_{t+1}$: this yields (52).

We now turn our attention to the proof of (53). [Local] reduces the goal to:

$$\vdash \{\mathcal{R}_p \mid \Upsilon_\alpha(q, \Gamma c_p, \sigma^2)\}\ \mathsf{draw}; \mathsf{body}_p \sim_\alpha \mathsf{draw}; \mathsf{body}_p\ \{\mathcal{R}_{\min(p+b,T)} \mid 1\}. \tag{57}$$

The crucial part of the proof is now to discharge the first sampling statement. There, we have two goals. The first is guided by the final budget that we want to achieve: to express the budget as

$$\Upsilon_\alpha(q, \Gamma c_p, \sigma^2) = \mathrm{H}_\alpha\big((1-q)\,\mathcal{N}(0,\sigma^2) + q\,\mathcal{N}(\Gamma c_p, \sigma^2) \,\big\|\, \mathcal{N}(0,\sigma^2)\big), \tag{58}$$

the subsampled-Gaussian integral (2). The other is to enrich the logical invariant $\mathcal{R}_p$ with the conjunct

$$\mathcal{W} \triangleq \left(D^{\langle 2\rangle}[j] = \bot \wedge w.2^{\langle 2\rangle} = w.2^{\langle 1\rangle} + w.1^{\langle 1\rangle}\,\Delta_p\right) \vee \left(D^{\langle 1\rangle}[j] = \bot \wedge w.2^{\langle 1\rangle} = w.2^{\langle 2\rangle} + w.1^{\langle 2\rangle}\,\Delta_p\right), \tag{59}$$

where g is the clipped gradient of the record j in the run that holds it, and Δ_p collects its contributions to the b rows released during the block:

$$g \triangleq \begin{cases} \mathsf{clip}_\Gamma(\nabla \ell(\theta, D^{\langle 1\rangle}[j])) & \text{if } D^{\langle 2\rangle}[j] = \bot, \\ \mathsf{clip}_\Gamma(\nabla \ell(\theta, D^{\langle 2\rangle}[j])) & \text{if } D^{\langle 1\rangle}[j] = \bot, \end{cases} \qquad \Delta_p \triangleq \left(C_{p+r,p}\, g\right)_{r<b}.$$

We set $C_{p+r,p} \triangleq 0$ when $p + r \ge T$; these entries belong to rounds of a truncated last block, which are never run. The models of the two runs are equal here, so θ carries no side. Like sh_p below, g and Δ_p are terms over the pair of states, not program variables.

The condition $\mathcal{W}$ comes from the following observation. In the run that holds the record, the subsampling loop adds the coin of j times g to the slot p of the history, and every row released during the block reads this slot through C. The rows of the two runs therefore differ by the coin times Δ_p. If the run without the record draws its noise shifted by the same amount, the two released rows coincide. The first disjunct of $\mathcal{W}$ covers the case where the record is missing on the right, the second the case where it is missing on the left.

We write sh_p for the difference that the record makes in the slot p, left minus right:

$$\mathrm{sh}_p \triangleq \begin{cases} w.1^{\langle 1\rangle}\, g & \text{if } D^{\langle 2\rangle}[j] = \bot, \\ -w.1^{\langle 2\rangle}\, g & \text{if } D^{\langle 1\rangle}[j] = \bot. \end{cases}$$

In both disjuncts of $\mathcal{W}$, the noises of row r then satisfy $w.2^{\langle 2\rangle}[r] - w.2^{\langle 1\rangle}[r] = C_{p+r,p}\,\mathrm{sh}_p$.

The sampling statement. The statement draw samples the coin of the record j together with the b noises of the block. We apply [Samp-Dec] (Figure 11) to it once, at the budget $\max(\Upsilon_\alpha, \Lambda_\alpha)(q, \Gamma c_p, \sigma^2)$. The rule allows the decomposition (ν_s, F_s) to depend on the pair of states s, and we choose it according to the run that holds the record. This case distinction is made in the meta-logic.

The left run holds the record ($D^{\langle 2\rangle}[j] = \bot$). Let

$$M_p \triangleq (1-q)\,\mathcal{N}(0,\sigma^2 I) + q\,\mathcal{N}(\Delta_p,\sigma^2 I)$$

be the law of the noise shifted by Δ_p times a coin of bias q. The posterior of the coin given a value W of this noise is

$$\pi(1 \mid W) \triangleq \frac{\mathrm{d}\big(q\,\mathcal{N}(\Delta_p,\sigma^2 I)\big)}{\mathrm{d}M_p}(W), \qquad \pi(0 \mid W) \triangleq 1 - \pi(1 \mid W), \tag{60}$$

a Radon–Nikodym derivative; with Lebesgue densities, it is the ratio of the two Gaussian densities at W. We write $K_W \triangleq \pi(\cdot \mid W) \otimes \delta_W$ for the kernel that keeps W and draws a coin from its posterior, and $h(u, W) \triangleq (u,\ W - u\,\Delta_p)$ for the map that removes the shift induced by the coin u. The decomposition is

$$\begin{aligned} \textbf{dlet } v \leftarrow \nu \textbf{ in } F(v) \;&=\; \mathsf{Bern}(q) \otimes \mathcal{N}(0,\sigma^2 I), \\ \nu \triangleq h^\sharp\big(\textbf{dlet } W \leftarrow \mathcal{N}(0,\sigma^2 I) \textbf{ in } K_W\big), \qquad & F(u,Z) \triangleq \mathsf{Bern}(q) \otimes \delta_{Z+u\,\Delta_p}. \end{aligned} \tag{61}$$

In words, ν draws the noise of the run without the record, infers a coin from it and subtracts the corresponding shift. The kernel F draws a fresh coin u, independent of the noise as the rule requires, and adds the shift $u\,\Delta_p$ back.

We check the two premises of the rule. For the budget, the law of the draw has the same form as ν, with M_p in place of $\mathcal{N}(0,\sigma^2 I)$:

$$\mathsf{Bern}(q) \otimes \mathcal{N}(0,\sigma^2 I) \;=\; h^\sharp\big(\textbf{dlet } W \leftarrow M_p \textbf{ in } K_W\big). \tag{62}$$

Hence

$$\begin{aligned} \mathrm{H}_\alpha\big(\mathsf{Bern}(q) \otimes \mathcal{N}(0,\sigma^2 I) \,\big\|\, \nu\big) \;&\le\; \mathrm{H}_\alpha\big(\textbf{dlet } W \leftarrow M_p \textbf{ in } K_W \,\big\|\, \textbf{dlet } W \leftarrow \mathcal{N}(0,\sigma^2 I) \textbf{ in } K_W\big) \\ &=\; \mathrm{H}_\alpha\big(M_p \,\big\|\, \mathcal{N}(0,\sigma^2 I)\big) \;=\; \Upsilon_\alpha(q, \|\Delta_p\|, \sigma^2) \;\le\; \Upsilon_\alpha(q, \Gamma c_p, \sigma^2). \end{aligned} \tag{63}$$

The first inequality is the data processing inequality for the Hellinger integral (item 1 of Proposition A.20) along h. The following equality holds because both sides apply the same kernel K_W, and the last inequality uses $\|\Delta_p\| \le \Gamma c_p$. For the postcondition, F adds $u\,\Delta_p$ to the noise Z, where u is the coin it draws; this establishes the first disjunct of $\mathcal{W}$.

The right run holds the record ($D^{\langle 1\rangle}[j] = \bot$). The roles of the two runs are exchanged:

$$\nu' \triangleq \mathsf{Bern}(q) \otimes M_p, \qquad F'(u,Z) \triangleq h^\sharp K_Z.$$

The left noise Z is drawn from M_p, and F' infers the coin of the right run from Z and removes the shift. By (62), $\textbf{dlet } v \leftarrow \nu' \textbf{ in } F'(v) = \mathsf{Bern}(q) \otimes \mathcal{N}(0,\sigma^2 I)$. The kernel F' establishes the second disjunct of $\mathcal{W}$, and the cost is

$$\mathrm{H}_\alpha\big(\mathsf{Bern}(q) \otimes \mathcal{N}(0,\sigma^2 I) \,\big\|\, \nu'\big) \;=\; \mathrm{H}_\alpha\big(\mathcal{N}(0,\sigma^2 I) \,\big\|\, M_p\big) \;=\; \Lambda_\alpha(q, \|\Delta_p\|, \sigma^2) \;\le\; \Lambda_\alpha(q, \Gamma c_p, \sigma^2).$$

In both cases the cost is bounded by $\max(\Upsilon_\alpha, \Lambda_\alpha)(q, \Gamma c_p, \sigma^2)$, and by Lemma E.1 this maximum equals $\Upsilon_\alpha(q, \Gamma c_p, \sigma^2)$, the budget of (57).

The rounds of the block. Under $\mathcal{W}$, every round of body_p is derived at budget 1. Between rounds we use the assertions (for $r \le \min(b, T-p)$)

$$\Theta_r \triangleq D^{\langle 1\rangle} \simeq_j D^{\langle 2\rangle} \wedge \theta^{\langle 1\rangle} = \theta^{\langle 2\rangle} \wedge Y^{\langle 1\rangle} = Y^{\langle 2\rangle} \wedge \tau^{\langle 1\rangle} = \tau^{\langle 2\rangle} \wedge t^{\langle 1\rangle} = t^{\langle 2\rangle} = p + r \wedge \mathcal{W}$$
$$\wedge\, \forall p' \neq p.\ p + r < p' + b \implies H^{\langle 1\rangle}[p'] = H^{\langle 2\rangle}[p'] \wedge H^{\langle 1\rangle}[p] - H^{\langle 2\rangle}[p] = \begin{cases} \mathrm{sh}_p & 1 \le r\,, \\ 0 & r = 0\,. \end{cases} \tag{64}$$

They differ from the invariant (51) at $t = p + r$ in two respects. They include $\mathcal{W}$, which relates the noises that remain to be read. And the slot p is not required to be equal in the two runs: after round 0 has written it, its two values differ by sh_p. The rounds are chained by [Seq]:

$$\vdash \{\Theta_r \mid 1\}\ \widetilde{\mathsf{round}}_r \sim_\alpha \widetilde{\mathsf{round}}_r\ \{\Theta_{r+1} \mid 1\} \qquad (r < \min(b, T-p))\,. \tag{65}$$

Each round runs the subsampling loop and then tail_r, which writes the slot, reads the noise, writes the row and releases the model.

Subsampling. For $r \ge 1$ the record j is not due, and Pois^b_q is treated as in (54): it adds the same vector to both accumulators. For $r = 0$ the loop is unrolled into $\widetilde{\mathsf{Pois}}$, and we derive

$$\vdash \{\Theta_0 \mid 1\}\ \widetilde{\mathsf{Pois}} \sim_\alpha \widetilde{\mathsf{Pois}}\ \{\Theta_0 \wedge acc^{\langle 1\rangle} - acc^{\langle 2\rangle} = \mathrm{sh}_p \mid 1\} \tag{66}$$

by [Seq] through the assertions (55), with Θ_0 in place of $\mathcal{R}_t$. The steps for the records before and after j (step^j and $\mathrm{step}^{|D|-1-j}$) add the same vectors on both sides, as in (54). In step_j the guard holds, since $p \in P_j$, and the coin is read from $w.1$ by [Assn]. The run that holds the record adds the coin times g to its accumulator, and the other run adds 0, since $\nabla\ell(\theta, \bot) = 0$. The accumulators then differ by sh_p in either disjunct of $\mathcal{W}$.

The tail. We derive it backwards with [Assn] and [Release]. The assignment $H[t] \leftarrow acc$ copies the difference of the accumulators into the slot $p + r$: the difference is sh_p for $r = 0$, and 0 for $r \ge 1$.

So when the row is written, the slot p differs by sh_p in every round. For $r \ge 1$ this is already part of Θ_r. For $r = 0$ it is not: Θ_0 describes the state at the start of the round, before the slot p is written, and says that the slot is equal in the two runs. We therefore write Θ^+_r for the assertion Θ_r in which the last conjunct is replaced by $H^{\langle 1\rangle}[p] - H^{\langle 2\rangle}[p] = \mathrm{sh}_p$, for every r. It only differs from Θ_r when $r = 0$. For every $r < \min(b, T-p)$ we derive

$$\vdash \{\Theta^+_r \wedge z^{\langle 2\rangle} - z^{\langle 1\rangle} = C_{p+r,p}\,\mathrm{sh}_p \mid 1\}\ Y[t] \leftarrow (C \cdot H)_t + z \sim_\alpha Y[t] \leftarrow (C \cdot H)_t + z\ \{\Theta^+_r \mid 1\}\,. \tag{67}$$

The relation between the noises in the precondition follows from $\mathcal{W}$ and the read $z \leftarrow w.2[r]$. The slot p is the only slot on which the histories differ, and its contribution to the row cancels the difference of the noises:

$$\big((C \cdot H)_{p+r} + z\big)^{\langle 1\rangle} - \big((C \cdot H)_{p+r} + z\big)^{\langle 2\rangle} \;=\; C_{p+r,p}\,\mathrm{sh}_p \;-\; C_{p+r,p}\,\mathrm{sh}_p \;=\; 0\,. \tag{68}$$

With equal rows, the two runs compute the same model and release the same value.

Conclusion of the proof. [Seq] combines the sampling statement with the rounds, and [Conseq] adjusts the two ends: $\mathcal{R}_p \wedge \mathcal{W} \models \Theta_0$ at the start, and $\Theta_{\min(b, T-p)} \models \mathcal{R}_{\min(p+b, T)}$ at the end, where the slot p is no longer read. The budgets multiply to $\Upsilon_\alpha(q, \Gamma c_p, \sigma^2) \cdot 1 \cdots 1$, the budget of (57), which proves (53).

Comparison with Choquette-Choo et al. [27]. The algorithm analysed here follows the *b-min-sep-participation* pattern of Choquette-Choo et al. [27]: the records are partitioned into b groups by their index, and the group t mod b is the one used for subsampling at round t. The judgment (44) is the counterpart of their amplification theorem [27, Theorem 4], which states that the mechanism satisfies every standard privacy guarantee of k subsampled-Gaussian queries at sensitivity κ, where

k bounds the number of participations of a record and κ every column norm of C. The algorithms and our formal results deviate in three respects.

(1) Within the due group, they draw a minibatch of fixed size without replacement; we toss a Poisson coin at rate q for each record of the group.
(2) Their bound is uniform over the records: every query is accounted for at the maximal column norm κ and the same rate. Our bound (44) is individual. Converting it from individual to global RDP (Lemma 3.7), and bounding every factor by the maximal column norm, recovers the same kind of guarantee, with $k = \max_j |P_j|$ and $\kappa = c$.
(3) Their theorem is a reduction: the mechanism inherits any DP guarantee of the k-fold composition of the subsampled Gaussian mechanism, for a wide class of guarantees including RDP. Our (44) gives the RDP bound itself.

F Deployed Implementations of the Sampled-Gaussian Accountant

The paper claims that the per-step formulas of Mironov et al. [59] reported in Proposition E.2 and the conversion of Proposition 3.5 are used in deployments to report (ϵ, δ)-differential privacy guarantees. This appendix gives references to those libraries. Table 2 lists several deployments that implement the per-step formulas of Mironov et al. [59], and the conversion each of them applies. The libraries attribute the conversion either to Balle et al. [12, Theorem 21] or to Canonne et al. [25, Proposition 12]; the two conversions are equivalent. Sources were read on 2026-10-15. Links to the sources follow.

Table 2. Where the per-step computation of Mironov et al. [59, Section 3.3] and the Rényi-to-(ϵ, δ) conversion are implemented. "Integer- and fractional-order computations" means the integer-order binomial sum and the fractional-order split at z_1, evaluated through log erfc.

Library	Per-step Rényi cost of the sampled Gaussian	Conversion
Opacus [77]	integer- and fractional-order computations	Balle et al. [12]
dp_accounting [41]	integer- and fractional-order computations	Canonne et al. [25]
TensorFlow Privacy [70]	through dp_accounting	Canonne et al. [25]
Flower [23]	through dp_accounting	Canonne et al. [25]
FedML [46]	integer- and fractional-order computations, derived from Opacus	Balle et al. [12]
fast-differential-privacy [9]	integer- and fractional-order computations	Canonne et al. [25]
private-transformers [52]	integer- and fractional-order computations	Canonne et al. [25]
SmartNoise [71]	through Opacus	Balle et al. [12]

Links. In the order of Table 2.

- Opacus: the two per-step computations are _compute_log_a_for_int_alpha and _compute_log_a_for_frac_alpha of https://github.com/pytorch/opacus/blob/main/opacus/accountants/analysis/rdp.py; the conversion is get_privacy_spent in the same file.
- dp_accounting: _compute_log_a_int, _compute_log_a_frac and the conversion are in https://github.com/google/differential-privacy/blob/main/python/dp_accounting/dp_accounting/rdp/rdp_privacy_accountant.py.
- TensorFlow Privacy: the library's own copy of the computation was removed in July 2022 and the dependency on dp_accounting kept (https://github.com/tensorflow/privacy/blob/master/tensorflow_privacy/privacy/analysis/compute_dp_sgd_privacy_lib.py). The last version carrying the computation itself is https://github.com/tensorflow/privacy/blob/v.0.8.0/tensorflow_privacy/privacy/analysis/rdp_accountant.py.

- Flower: https://github.com/adap/flower/blob/main/framework/py/flwr/supercore/privacy_accounting/rdp_accountant.py.
- FedML: https://github.com/FedML-AI/FedML/blob/master/python/fedml/core/dp/budget_accountant/rdp_analysis.py for the integer- and fractional-order computations and https://github.com/FedML-AI/FedML/blob/master/python/fedml/core/dp/budget_accountant/rdp_accountant.py for the Rényi-to-(ϵ, δ) one.
- fast-differential-privacy: https://github.com/awslabs/fast-differential-privacy/blob/main/fastDP/accounting/rdp_accounting.py.
- private-transformers: https://github.com/lxuechen/private-transformers/blob/main/private_transformers/accounting/rdp_accounting.py.
- SmartNoise: the dependency is declared in https://github.com/opendp/smartnoise-sdk/blob/main/synth/pyproject.toml and used in https://github.com/opendp/smartnoise-sdk/blob/main/synth/snsynth/pytorch/nn/dpgan.py.