Verifying Differential Privacy in Floating-Point Computation

Abstract

The programs implementing differentially private algorithms can be formally verified via existing logics and tools, such as apRHL^+[2]. These verifications rely on the assumption that these implementations whose underlining computation is real number based without computation error. However, in reality, these algorithms are mostly implemented under finite precision, which accounts for computation errors and causes critical failures in preserving differential privacy.

We develop a method for formally verifying differential privacy of algorithms implemented based on floating-point computation. Our method extends the relational program logic apRHL^+, and provides an operational semantics under floating-point computation, which work together to provide the formal verification.

We demonstrate our method via an algorithm: the *Snapping mechanism* proposed in [8], where its differential privacy is theoretically proved even with floating-point computation error. We implement our logic and example verifications in Coq.

1 Introduction

Differential privacy proposed by Dwork, McSherry, Nissim, and Smith [4] is considered the gold standard for privacy-preserving computations. Existing methods are able to formally verify the differential privacy of most of the popular algorithms [2]. However, these verifications are based on the ideal implementations of these algorithms without computation error. While in reality, the implementations of these algorithms are not ideal, mostly in floating-point computation with computation errors. These computation errors can cause critical failures in preserving differential privacy, which is identified in the literature [1, 5, 7, 8, 6]. In the meantime, some algorithms claim to preserve differential privacy even in floating-point computation, for instance:

- The *Snapping mechanism*, which is an improved version of the implementation of Laplace mechanism in floating-point computation, by adding extra rounding and clamping operations. (See [8].)
- The *Base-2 Exponential mechanism*, which is a revised version of the Exponential mechanism, by altering the base in Exponential mechanism from e into 2. (See [6].)
- The *Discrete Gaussian mechanism*, which is a discrete version of the Gaussian mechanism. (See [3].)

Unfortunately, existing pen-and-paper proofs of these algorithms use complex computation error approximations, which are hard to understand and error-prone.

This gives us the motivation that can we develop methods for formally and automatically verifying these algorithms by considering the errors that arose during the floating-point computation.

To verify the differential privacy property of the algorithms implemented in floating-point computation, we extend the relational program logic apRHL^+ from [2]. Then we compose it with an operational semantics designed for approximating the floating-point computation error. We prove the soundness via the methodology of probabilistic coupling and approximate lifting shown by Barth et al. [2].

2 System Overview

We propose a formal analysis of differentially private algorithms under finite computation, whose proof does not exclusively rely on the existing tools of differential privacy. The paper is composed of three main components.

The language and operational semantics We firstly design an operational semantics for approximating the floating-point computation errors on a standard imperative language with only the assignment, distribution sampling, and sequence commands.

In this language, the expressions (in real number precision) are evaluated into floating-point values $v^{\mathbb{F}}$ through floating-point computation with computation error arose during evaluation.

In order to track the computation error propagated in floating-point computation, we attach a pair of real values $(v_1^{\mathbb{R}}, v_2^{\mathbb{R}})$ to approximate the computation error in the environment for every variable. To be specific, a variable is mapped to $(v^{\mathbb{F}}_{(v_1^{\mathbb{R}}, v_2^{\mathbb{R}})})$, where $v^{\mathbb{F}}$ is a value evaluated from some expressions in floating-point precision, and $v_1^{\mathbb{R}}, v_2^{\mathbb{R}}$ are lower and upper bounds of the floating-point computation error propagated when evaluating into $v^{\mathbb{F}}$: $\mathtt{m}, e \Rightarrow v^{\mathbb{F}}_{(v_1^{\mathbb{R}}, v_2^{\mathbb{R}})}$. For example, $[], 1+1 \Rightarrow 2_{(1.99, 2.01)}$, where ± 0.01 are the relative computation errors relevant to the machine epsilon η.

The operational semantics for programs takes an environment $\mathtt{m}$ and a program p and returns a distribution over environment $\mathtt{m}'$: $\mathtt{m}, p \Rightarrow Distr(\mathtt{m}')$. For example, $[], x \xleftarrow{\$} \mathsf{unif}(0,1] \Rightarrow$ $\mathtt{let}$ $0.1_{(0.1,0.1)} = [\![\mathsf{unif}(0,1]]\!]$ $\mathtt{in}$ $\mathsf{unit}([x \mapsto 0.1_{(0.1,0.1)}])$, where we assume the sample from discrete floating-point numbers produce no error, accounting for the $0.1_{(0.1,0.1)}$ in the environment.

An extended apRHL^+ To verify the differential privacy, we adopt the proof principles of probabilistic coupling and approximate lifting and work with a relational program logic apRHL^+.

We conceive it as a floating-point computational variant of apRHL^+ based on the new operational semantics, i.e., in the environments, variables are mapping to floating-point values together with their lower and upper bounds on computation errors, instead of a single real number. The judgments are of the form: $p_1 \sim_\epsilon p_2 : \Phi \Rightarrow \Psi$, where p_1 and p_2 are programs and Φ and Ψ are assertions on pairs of the newly defined environments. Each assertion refers to two copies $x\langle 1\rangle$, $\langle 2\rangle$ of each program variable x, where tagged variables refer to the value of x in executions of p_1 and p_2 respectively. Different from apRHL^+, each assertion can refer to any values to which the variables are mapped in the environment through annotations on the top right of variables. For example, x^1, x^2 and x^3 refer to $v^{\mathbb{F}}, v_1^{\mathbb{R}}$ and $v_2^{\mathbb{R}}$ respectively if x is mapped to $v^{\mathbb{F}}_{(v_1^{\mathbb{R}}, v_2^{\mathbb{R}})}$ in the environment.

Informally, a judgment of the above form is valid if the two distributions produced by the executions of p_1 and p_2 on any two initial memories satisfying the precondition Φ are related by the ϵ-lifting (formally defined in 1) of the postcondition Ψ in floating-point computation. For example, $\mathsf{unif}(0,1]$ and $\mathsf{unif}(0,1]$ are related by the ϵ-lifting of the relation $\Phi \triangleq \{(x,y) \in (0,1]^{\mathbb{F}} \times (0,1]^{\mathbb{F}} | x = e^{\epsilon} y\}$, where $\mathsf{unif}(0,1]$ is the uniform distribution on floating-point numbers range over $(0,1]$.

Further, we introduce two proof rules representing the approximate probabilistic liftings for the process of sampling from uniform distribution —the key component of implementing the Laplace mechanism in floating-point computation— the **unif+** and **unif-**:

$\vdash x \xleftarrow{\$} \mathsf{unif}(0,1] \sim_\epsilon y \xleftarrow{\$} \mathsf{unif}(0,1] : \top \Rightarrow \forall l, r \in [0,1]^{\mathbb{R}}.\ l \leq x^1\langle 1\rangle \leq r \to (e^\epsilon l \leq y^1\langle 2\rangle \leq e^\epsilon r \wedge y^1\langle 2\rangle \leq 1);$
$\vdash x \xleftarrow{\$} \mathsf{unif}(0,1] \sim_\epsilon y \xleftarrow{\$} \mathsf{unif}(0,1] : \top \Rightarrow \forall l, r \in [0,1]^{\mathbb{R}}.\ l \leq x^1\langle 1\rangle \leq r \to (e^{-\epsilon} l \leq y^1\langle 2\rangle \leq e^{-\epsilon} r \leq 1),$
The two rules represent the ϵ-lifting of two uniform distribution on floating-point numbers range over $(0,1]$ w.r.t. to postconditions respectively. Although this process itself is not differentially private, its properties can be formally captured by approximate probabilistic liftings and be combined to show privacy for a larger program.

New privacy proofs We provide the first formal verification of the *Snapping mechanism*, which is a variant of the Laplace mechanism in floating-point computation with extra rounding and clamping operations. Its privacy proof is based on the floating-point computation error approximation, which cannot be formally verified by any of the existing tools. In contrast, we prove its privacy in floating-point computation within our logic.

We plan to formalize the language and the proof of the algorithm in Coq.

3 Syntax

The language is defined as follows.
In the language, $v^{\mathbb{R}}$ is in the domain of real number $\mathbb{R}$, $v^{\mathbb{R}} \in \mathbb{R}$. $v^{\mathbb{F}}$ is in the domain of floating point number $\mathbb{F}$, $v^{\mathbb{F}} \in \mathbb{F}$. Furthermore, $\mathbb{F}$ is a subset of $\mathbb{R}$ allowing the $v^{\mathbb{F}}$ be represented in $\mathbb{R}$ without lose of precision. On the other hand, $v^{\mathbb{R}}$ sacrifices its precision when casted into $\mathbb{F}$. The type $\mathbb{F}_{\mathbb{R}\times\mathbb{R}}$ represents the base type of values, composed of a floating-point value $v^{\mathbb{F}}$ and a pair of real-number values $v^{\mathbb{R}}$. In environment., the variables are mapped to $(v^{\mathbb{F}}_{(v^{\mathbb{R}},v^{\mathbb{R}})})$ of the type $\mathbb{F}_{\mathbb{R}\times\mathbb{R}}$.

Programs	p	::=	$x = e \mid x \xleftarrow{\$} \mu \mid p; p$
Expr	e	::=	$v^{\mathbb{R}} \mid x \mid e * e \mid \circ(e)$
Binary Operation	$*$	::=	$+ \mid - \mid \times \mid \div$
Unary Operation	$\circ$	::=	$\ln \mid - \mid \lfloor\cdot\rceil \mid \mathsf{clamp}_B(\cdot)$
Value	v	::=	$v^{\mathbb{F}}_{(v^{\mathbb{R}},v^{\mathbb{R}})}$
Distr	μ	::=	$\mathsf{unif}(0,1] \mid \mathsf{unif}\{-1,1\}$
Error	err	::=	$(v^{\mathbb{R}}, v^{\mathbb{R}})$
Env	$\mathtt{m}$	::=	$\cdot \mid \mathtt{m}[x \mapsto (v^{\mathbb{F}}_{(v^{\mathbb{R}},v^{\mathbb{R}})})]$
Type	τ	::=	$\mathbb{F} \mid \mathbb{R} \mid \mathbb{F}_{\mathbb{R}\times\mathbb{R}}$

Semantics. The denotational semantics are defined as follows.
$\boxed{Env \times Expr \to Value}$

$$[\![e]\!]_{\mathtt{m}} \in \left\{v^{\mathbb{F}}_{(v^{\mathbb{R}}_l,v^{\mathbb{R}}_u)} \mid \mathtt{m}, e \Rightarrow (v^{\mathbb{F}}_{(v^{\mathbb{R}}_l,v^{\mathbb{R}}_u)})\right\}$$

$\boxed{Env \times \mathsf{Distr} \to \mathsf{Distr}(Value)}$

$$\begin{aligned}[\![\mathsf{unif}(0,1]]\!]_{\mathtt{m}} &\in \left\{(v^{\mathbb{F}}_{(v^{\mathbb{R}}_l,v^{\mathbb{R}}_u)}) \mid v^{\mathbb{F}} \leftarrow \mathcal{U}(0,1] \wedge v^{\mathbb{R}}_l = v^{\mathbb{R}}_u = v^{\mathbb{F}}\right\} \\ [\![\mathsf{unif}\{-1,1\}]\!]_{\mathtt{m}} &\in \left\{(-1_{(-1,-1)}), (1_{(1,1)}) \mid each\ w.p.\ 0.5\right\}\end{aligned}$$

$\boxed{Env \times prog \to \mathsf{Distr}(Env)}$

$$\begin{array}{rcl}
[\![x \xleftarrow{\$} \mu]\!]_{\mathtt{m}} & = & \mathtt{let}\ v^{\mathbb{F}}_{(v^{\mathbb{R}}_l, v^{\mathbb{R}}_u)} = [\![\mu]\!]_{\mathtt{m}}\ \mathtt{in}\ \mathsf{unit}(\mathtt{m}[x \mapsto v^{\mathbb{F}}_{(v^{\mathbb{R}}_l, v^{\mathbb{R}}_u)}]) \\
[\![x = e]\!]_{\mathtt{m}} & = & \mathsf{unit}(\mathtt{m}[x \mapsto [\![e]\!]_{\mathtt{m}}]) \\
[\![p_1; p_2]\!]_{\mathtt{m}} & = & \mathtt{let}\ \mathtt{m}_1 = [\![p_1]\!]_{\mathtt{m}}\ \mathtt{in}\ [\![p_2]\!]_{\mathtt{m}_1}
\end{array}$$

In the semantics, $\mathtt{m}, e \Rightarrow (v^{\mathbb{F}}_{err})$ reads given an environment $\mathtt{m}$, the expression e is transited to $v^{\mathbb{F}}$ with error bound $err = (v^{\mathbb{R}}_l, v^{\mathbb{R}}_u)$ in floating point transition semantics, s.t. $v^{\mathbb{R}}_l \leq v^{\mathbb{F}} \leq v^{\mathbb{R}}_u$. The semantics is presented in Figure. 1. $\mathtt{m}, p \Rightarrow \mathtt{m}'$ represents, given an environment $\mathtt{m}$, the program p is transited to a new environment $\mathtt{m}'$. The $\mathcal{U}(0,1] \in \mathsf{Distr}(\mathbb{F})$ is the mathematic uniform distribution over floating point values ranging over $(0,1]$.

4 Judgement and Validity

Definition 1 (ϵ−lifting [2])
Two sub-distributions $\mu_1 \in \mathsf{Distr}(\mathcal{D}_1)$, $\mu_2 \in \mathsf{Distr}(\mathcal{D}_2)$ are related by the ϵ−lifting of $\Psi \subseteq \mathcal{D}_1 \times \mathcal{D}_2$, written $\mu_1 \Psi^{\#(\epsilon)} \mu_2$, if there exist two witness sub-distributions $\mu_L \in \mathsf{Distr}(\mathcal{D}_1 \times \mathcal{D}_2)$ and $\mu_R \in \mathsf{Distr}(\mathcal{D}_1 \times \mathcal{D}_2)$ s.t.:

1. $\pi_1(\mu_L) = \mu_1$ and $\pi_2(\mu_R) = \mu_2$;
2. $\mathsf{supp}(\mu_L) \subseteq \Psi$ and $\mathsf{supp}(\mu_R) \subseteq \Psi$; and
3. $\Delta_\epsilon(\mu_L, \mu_R) \leq 0.0$.

Definition 2 (Λ equivalent)
Given two floating point values v_1 and v_2, if for some floating point value $v^{\mathbb{F}}$ which is a multiple of Λ:

$$v^{\mathbb{F}} - \frac{\Lambda}{2} \leq v_1 < v^{\mathbb{F}} + \frac{\Lambda}{2} \ \wedge \ v^{\mathbb{F}} - \frac{\Lambda}{2} \leq v_2 < v^{\mathbb{F}} + \frac{\Lambda}{2},$$

then v_1 and v_2 are Λ equivalent, i.e., $v_1 \equiv_\Lambda v_2 \equiv_\Lambda v^{\mathbb{F}}$.

Definition 3 (tagged variable)
Let $\mathcal{X}\langle 1\rangle$ and $\mathcal{X}\langle 2\rangle$ be the sets of tagged variables, finite sets of variable names tagged with $\langle 1\rangle$ or $\langle 2\rangle$ respectively:

$$\mathcal{X}\langle 1\rangle = \{x\langle 1\rangle \mid x \in \mathcal{X}\} \text{ and } \mathcal{X}\langle 2\rangle = \{x\langle 2\rangle \mid x \in \mathcal{X}\},$$

where $\mathcal{X}$ is a finite set of variable names.

Assertion. We consider a set $\mathcal{A}$ of assertions (predicates) from first order logic by the following grammar:

$$\begin{array}{lll}
\text{Logic Expr.}\ \mathcal{E} & ::= & \mathcal{L} \mid \mathcal{V} \mid \mathcal{E}+\mathcal{E} \mid \mathcal{E}-\mathcal{E} \mid \mathcal{E}\cdot\mathcal{E} \mid \ln(\mathcal{E}) \mid -\mathcal{E} \mid e^{\mathcal{E}} \\
\text{Assert.}\ \mathcal{A} & ::= & e^i\langle 1/2\rangle = e^i\langle 1/2\rangle \mid e^i\langle 1/2\rangle < e^i\langle 1/2\rangle) \mid e^i\langle 1/2\rangle \leq e^i\langle 1/2\rangle \mid e^i\langle 1/2\rangle \equiv_\Lambda e^i\langle 1/2\rangle \\
 & & \mid \mathcal{E} = \mathcal{E} \mid \cdots \mid \mathcal{E} = e^i\langle 1/2\rangle \mid \cdots \mid e^i\langle 1/2\rangle = \mathcal{E} \mid \cdots \\
 & & \mid \top \mid \bot \mid \mathcal{A} \wedge \mathcal{A} \mid \mathcal{A} \vee \mathcal{A} \mid \neg\mathcal{A} \mid \mathcal{A} \to \mathcal{A} \mid \forall L \in \mathcal{D}.\,\mathcal{A} \mid \exists L \in \mathcal{D}.\,\mathcal{A}
\end{array}$$

We typically use capital Greek letters $(\Phi, \Psi, \cdots)$ for predicates. $e\langle 1/2\rangle$ denotes an expression where program variables are tagged with $\langle 1\rangle$ or $\langle 2\rangle$. $e^i\langle 1/2\rangle$ represents an expression where program variables are projected to the i^{th} value from its triples, where $i \in \{1,2,3\}$. $\mathcal{D}$ is a specfic domain, it could

$$\boxed{\Theta, e \Rightarrow v^{\mathbb{F}}_{(v^{\mathbb{R}}, v^{\mathbb{R}})} : Env \times Expr \Rightarrow Value}$$

$$\frac{\Theta(x) = (v^{\mathbb{F}}_{(\underline{v}^{\mathbb{R}}, \bar{v}^{\mathbb{R}})})}{\Theta, x \Rightarrow (v^{\mathbb{F}}_{(\underline{v}^{\mathbb{R}}, \bar{v}^{\mathbb{R}})})}\ \text{VAR} \qquad \frac{v^{\mathbb{F}} = \mathtt{fl}(v^{\mathbb{R}}) \quad v^{\mathbb{F}} \neq v^{\mathbb{R}} \qquad v^{\mathbb{R}} \geq 0}{\Theta, v^{\mathbb{R}} \Rightarrow \left(v^{\mathbb{F}}_{(\frac{v^{\mathbb{R}}}{(1+\eta)}, v^{\mathbb{R}}(1+\eta))}\right)}\ \text{VAL}$$

$$\frac{v^{\mathbb{F}} = \mathtt{fl}(v^{\mathbb{R}}) \quad v^{\mathbb{F}} \neq v^{\mathbb{R}} \qquad v^{\mathbb{R}} < 0}{\Theta, v^{\mathbb{R}} \Rightarrow \left(v^{\mathbb{F}}_{(v^{\mathbb{R}}(1+\eta), \frac{v^{\mathbb{R}}}{(1+\eta)})}\right)}\ \text{VAL-NEG} \qquad \frac{v^{\mathbb{F}} = \mathtt{fl}(v^{\mathbb{R}}) \quad v^{\mathbb{F}} = v^{\mathbb{R}}}{\Theta, v^{\mathbb{R}} \Rightarrow (v^{\mathbb{F}}_{(v^{\mathbb{R}}, v^{\mathbb{R}})})}\ \text{VAL-EQ}$$

$$\frac{\Theta, e_1 \Rightarrow (v^{\mathbb{F}}_{1(\underline{v}_1^{\mathbb{R}}, \bar{v}_1^{\mathbb{R}})}) \quad \Theta, e_2 \Rightarrow (v^{\mathbb{F}}_{2(\underline{v}_2^{\mathbb{R}}, \bar{v}_2^{\mathbb{R}})}) \quad v_1^{\mathbb{F}} \geq 0 \quad v_2^{\mathbb{F}} \geq 0 \quad v^{\mathbb{F}} = \mathtt{fl}(v_1^{\mathbb{F}} * v_2^{\mathbb{F}}) \quad * \in \{\times, \div\}}{\Theta, e_1 * e_2 \Rightarrow \left(v^{\mathbb{F}}_{(\frac{\underline{v}_1^{\mathbb{R}} * \underline{v}_2^{\mathbb{R}}}{(1+\eta)}, (\bar{v}_1^{\mathbb{R}} * \bar{v}_2^{\mathbb{R}})(1+\eta))}\right)}\ \text{BOP-PP}$$

$$\frac{\Theta, e_1 \Rightarrow (v^{\mathbb{F}}_{1(\underline{v}_1^{\mathbb{R}}, \bar{v}_1^{\mathbb{R}})}) \quad \Theta, e_2 \Rightarrow (v^{\mathbb{F}}_{2(\underline{v}_2^{\mathbb{R}}, \bar{v}_2^{\mathbb{R}})}) \quad v_1^{\mathbb{F}} < 0 \quad v_2^{\mathbb{F}} < 0 \quad v^{\mathbb{F}} = \mathtt{fl}(v_1^{\mathbb{F}} * v_2^{\mathbb{F}}) \quad * \in \{\times, \div\}}{\Theta, e_1 * e_2 \Rightarrow \left(v^{\mathbb{F}}_{(\frac{\bar{v}_1^{\mathbb{R}} * \bar{v}_2^{\mathbb{R}}}{(1+\eta)}, (\underline{v}_1^{\mathbb{R}} * \underline{v}_2^{\mathbb{R}})(1+\eta))}\right)}\ \text{BOP-NN}$$

$$\frac{\Theta, e_1 \Rightarrow (v^{\mathbb{F}}_{1(\underline{v}_1^{\mathbb{R}}, \bar{v}_1^{\mathbb{R}})}) \quad \Theta, e_2 \Rightarrow (v^{\mathbb{F}}_{2(\underline{v}_2^{\mathbb{R}}, \bar{v}_2^{\mathbb{R}})}) \quad v_1^{\mathbb{F}} \geq 0 \quad v_2^{\mathbb{F}} < 0 \quad v^{\mathbb{F}} = \mathtt{fl}(v_1^{\mathbb{F}} * v_2^{\mathbb{F}}) \quad * \in \{\times, \div\}}{\Theta, e_1 * e_2 \Rightarrow \left(v^{\mathbb{F}}_{((\bar{v}_1^{\mathbb{R}} * \underline{v}_2^{\mathbb{R}})(1+\eta), \frac{\underline{v}_1^{\mathbb{R}} * \bar{v}_2^{\mathbb{R}}}{(1+\eta)})}\right)}\ \text{BOP-PN}$$

$$\frac{\Theta, e_1 \Rightarrow (v^{\mathbb{F}}_{1(\underline{v}_1^{\mathbb{R}}, \bar{v}_1^{\mathbb{R}})}) \quad \Theta, e_2 \Rightarrow (v^{\mathbb{F}}_{2(\underline{v}_2^{\mathbb{R}}, \bar{v}_2^{\mathbb{R}})}) \quad v_1^{\mathbb{F}} < 0 \quad v_2^{\mathbb{F}} \geq 0 \quad v^{\mathbb{F}} = \mathtt{fl}(v_1^{\mathbb{F}} * v_2^{\mathbb{F}}) \quad * \in \{\times, \div\}}{\Theta, e_1 * e_2 \Rightarrow \left(v^{\mathbb{F}}_{((\underline{v}_1^{\mathbb{R}} * \bar{v}_2^{\mathbb{R}})(1+\eta), \frac{\bar{v}_1^{\mathbb{R}} * \underline{v}_2^{\mathbb{R}}}{(1+\eta)})}\right)}\ \text{BOP-NP}$$

$$\frac{\Theta, e_1 \Rightarrow (v^{\mathbb{F}}_{1(\underline{v}_1^{\mathbb{R}}, \bar{v}_1^{\mathbb{R}})}) \quad \Theta, e_2 \Rightarrow (v^{\mathbb{F}}_{2(\underline{v}_2^{\mathbb{R}}, \bar{v}_2^{\mathbb{R}})}) \quad v^{\mathbb{F}} = \mathtt{fl}(v_1^{\mathbb{F}} * v_2^{\mathbb{F}}) \quad v^{\mathbb{F}} \geq 0 \quad * \in \{+, -\}}{\Theta, e_1 * e_2 \Rightarrow \left(v^{\mathbb{F}}_{(\frac{\underline{v}_1^{\mathbb{R}} * \underline{v}_2^{\mathbb{R}}}{(1+\eta)}, (\bar{v}_1^{\mathbb{R}} * \bar{v}_2^{\mathbb{R}})(1+\eta))}\right)}\ \text{BOP-P}$$

$$\frac{\Theta, e_1 \Rightarrow (v^{\mathbb{F}}_{1(\underline{v}_1^{\mathbb{R}}, \bar{v}_1^{\mathbb{R}})}) \quad \Theta, e_2 \Rightarrow (v^{\mathbb{F}}_{2(\underline{v}_2^{\mathbb{R}}, \bar{v}_2^{\mathbb{R}})}) \quad v^{\mathbb{F}} = \mathtt{fl}(v_1^{\mathbb{F}} * v_2^{\mathbb{F}}) \quad v^{\mathbb{F}} < 0 \quad * \in \{+, -\}}{\Theta, e_1 * e_2 \Rightarrow \left(v^{\mathbb{F}}_{((\underline{v}_1^{\mathbb{R}} * \underline{v}_2^{\mathbb{R}})(1+\eta), \frac{\bar{v}_1^{\mathbb{R}} * \bar{v}_2^{\mathbb{R}}}{(1+\eta)})}\right)}\ \text{BOP-N}$$

$$\frac{\Theta, e \Rightarrow (v^{\mathbb{F}}_{1(\underline{v}^{\mathbb{R}}, \bar{v}^{\mathbb{R}})}) \quad v^{\mathbb{F}} = \mathtt{fl}(\circ(v_1^{\mathbb{F}})) \quad v^{\mathbb{F}} \geq 0}{\Theta, \circ(e) \Rightarrow \left(v^{\mathbb{F}}_{\left(\frac{\circ(\underline{v}^{\mathbb{R}})}{(1+\eta)}, (\circ(\bar{v}^{\mathbb{R}}))(1+\eta)\right)}\right)}\ \text{UOP-P} \qquad \frac{\Theta, e \Rightarrow (v^{\mathbb{F}}_{1(\underline{v}^{\mathbb{R}}, \bar{v}^{\mathbb{R}})}) \quad v^{\mathbb{F}} = \mathtt{fl}(\circ(v_1^{\mathbb{F}})) \quad v^{\mathbb{F}} < 0}{\Theta, \circ(e) \Rightarrow \left(v^{\mathbb{F}}_{\left(\circ(\underline{v}^{\mathbb{R}})(1+\eta), \frac{\circ(\bar{v}^{\mathbb{R}})}{(1+\eta)}\right)}\right)}\ \text{UOP-N}$$

Figure 1: Semantics of Transition for Expressions with Relative Floating Point Error

be integers, reals, floating-point numbers or range of them, etc.
The logic context are maps $\mathcal{L} \to \mathcal{V}$; usually written ρ. The logic expression $\mathcal{E}$ is interpreted under real number computation, such as: $[\![\mathcal{E}_1 + \mathcal{E}_2]\!]_\rho = [\![\mathcal{E}_1]\!]_\rho + [\![\mathcal{E}_2]\!]_\rho$, etc.

Assertion Interpretation. Assertions are interpreted as relations between environments, i.e., set of paired environments. Let Φ be an assertion, by the definition of $\mathcal{A}$, we have $[\![\mathcal{A}]\!]$ as:
$[\![e^i\langle 1\rangle = e^i\langle 2\rangle]\!]_\rho = \{(\mathtt{m}_1,\mathtt{m}_2) \mid ([\![e]\!]_{\mathtt{m}_1})^i = ([\![e]\!]_{\mathtt{m}_2})^i\};\ \cdots;$ $[\![\mathcal{E}_1 = \mathcal{E}_2]\!]_\rho = \{(\mathtt{m}_1,\mathtt{m}_2) \mid [\![\mathcal{E}_1]\!]_\rho < [\![\mathcal{E}_2]\!]_\rho\};\ \cdots;$
$[\![e^i\langle 1\rangle < \mathcal{E}]\!]_\rho = \{(\mathtt{m}_1,\mathtt{m}_2) \mid ([\![e]\!]_{\mathtt{m}_1})^i < [\![\mathcal{E}]\!]_\rho\};\ \cdots;$ $[\![\mathcal{E} \le e^i\langle 2\rangle]\!]_\rho = \{(\mathtt{m}_1,\mathtt{m}_2) \mid [\![\mathcal{E}]\!]_\rho \le ([\![e]\!]_{\mathtt{m}_2})^i\};\ \cdots;$
$[\![\mathcal{A}_1 \wedge \mathcal{A}_2]\!]_\rho = \{(\mathtt{m}_1,\mathtt{m}_2) \mid (\mathtt{m}_1,\mathtt{m}_2) \in [\![\mathcal{A}_1]\!]_\rho \wedge (\mathtt{m}_1,\mathtt{m}_2) \in [\![\mathcal{A}_2]\!]_\rho\};$ $[\![\top]\!]_\rho = \{(\mathtt{m}_1,\mathtt{m}_2)\};$ $[\![\bot]\!]_\rho = \{\};$
$[\![\mathcal{A}_1 \vee \mathcal{A}_2]\!]_\rho = \{(\mathtt{m}_1,\mathtt{m}_2) \mid (\mathtt{m}_1,\mathtt{m}_2) \in [\![\mathcal{A}_1]\!]_\rho \vee (\mathtt{m}_1,\mathtt{m}_2) \in [\![\mathcal{A}_2]\!]_\rho\};$ $[\![\neg\mathcal{A}]\!]_\rho = \{(\mathtt{m}_1,\mathtt{m}_2) \mid (\mathtt{m}_1,\mathtt{m}_2) \notin [\![\mathcal{A}]\!]_\rho\};$
$[\![\mathcal{A}_1 \to \mathcal{A}_2]\!]_\rho = \{(\mathtt{m}_1,\mathtt{m}_2) \mid (\mathtt{m}_1,\mathtt{m}_2) \in [\![\mathcal{A}_1]\!]_\rho \to (\mathtt{m}_1,\mathtt{m}_2) \in [\![\mathcal{A}_2]\!]_\rho\};$
$[\![\forall L \in \mathcal{D}.\ \mathcal{A}]\!]_\rho = \{(\mathtt{m}_1,\mathtt{m}_2) \mid \forall v \in \mathcal{D}.\ (\mathtt{m}_1,\mathtt{m}_2) \in [\![\mathcal{A}]\!]_{\rho[L\to v]}\};$
$[\![\exists L \in \mathcal{D}.\ \mathcal{A}]\!]_\rho = \{(\mathtt{m}_1,\mathtt{m}_2) \mid \exists v \in \mathcal{D}.\ (\mathtt{m}_1,\mathtt{m}_2) \in [\![\mathcal{A}]\!]_{\rho[L\to v]}\}.$

Judgment. The judgments are defined in following form:

$$p_1 \sim_\epsilon p_2 : \Phi \Rightarrow \Psi.$$

Here, p_1 and p_2 are programs and Φ and Ψ are assertions on pairs of memories. Each assertion can refer to two copies $x\langle 1\rangle, x\langle 2\rangle$ of each program variable x, where these tagged variables refer to the value of x in the execution of p_1 and p_2 respectively.
A judgment is valid, written $\vdash p_1 \sim_\epsilon p_2 : \Phi_0 \Rightarrow \Phi$, if for any two environments $\mathtt{m}_1$ and $\mathtt{m}_2$ satisfying precondition Φ_0, i.e., $(\mathtt{m}_1,\mathtt{m}_2) \in [\![\Phi_0]\!]$, there exists a lifting of Φ relating the output distributions: $([\![p_1]\!]_{\mathtt{m}_1})\ [\![\Phi]\!]^{\#(\epsilon)}\ ([\![p_2]\!]_{\mathtt{m}_2})$.
Fig. 3 presents the main rules from apRHL+ [2] excluding the while and condition rules which are not defined in out syntax, as well as the sampling rule, which we generalized in extended apRHL. The extended rules in Fig. 2 represent the lifting proved in soundness theorem.

$\boxed{\vdash: prog \times prog \times \mathbb{R} \times Assert \times Assert}$

$$\frac{}{\vdash x \overset{\$}{\leftarrow} \mathsf{unif}(0,1] \sim_\epsilon y \overset{\$}{\leftarrow} \mathsf{unif}(0,1] : \top \Rightarrow \forall l, r \in [0,1]^{\mathbb{R}}.\ l \le x^1\langle 1\rangle \le r \to (e^\epsilon l \le y^1\langle 2\rangle \le e^\epsilon r \wedge y^1\langle 2\rangle \le 1)}\ \textbf{Unif+}$$

$$\frac{}{\vdash x \overset{\$}{\leftarrow} \mathsf{unif}(0,1] \sim_\epsilon y \overset{\$}{\leftarrow} \mathsf{unif}(0,1] : \top \Rightarrow \forall l, r \in [0,1]^{\mathbb{R}}.\ l \le x^1\langle 1\rangle \le r \to (e^{-\epsilon} l \le y^1\langle 2\rangle \le e^{-\epsilon} r)}\ \textbf{Unif-}$$

$$\frac{}{\vdash x_1 \overset{\$}{\leftarrow} \mu \sim_0 x_2 \overset{\$}{\leftarrow} \mu : \top \Rightarrow (x_2^1\langle 2\rangle) = (x_1^1\langle 1\rangle) \wedge (x_2^2\langle 2\rangle) = (x_1^2\langle 1\rangle) \wedge (x_2^3\langle 2\rangle) = (x_1^3\langle 1\rangle)}\ \textbf{Null}$$

$$\frac{\forall v.\ \text{a multiple of } \Lambda}{\vdash x_1 = \lfloor y_1 \rceil_\Lambda \sim_0 x_2 = \lfloor y_2 \rceil_\Lambda : y_1^1\langle 1\rangle \equiv_\Lambda v \to y_2^1\langle 2\rangle \equiv_\Lambda v \Rightarrow (x_1^1\langle 1\rangle = v) \to (x_2^1\langle 2\rangle = v)}\ \textbf{Round}$$

Figure 2: Rules Extended from apRHL+

Lemma 1 (Discrete Support of Distribution)
$\forall p, \mathtt{m}, \mu$, s.t. $\mu \in \mathsf{Distr}(Env)$ and $[\![p]\!]_{\mathtt{m}} = \mu$. Then $\mathsf{supp}(\mu)$ is discrete.

$$\frac{}{\vdash x_1 = e_1 \sim_0 x_2 = e_2 : \Phi[e_1/x_1\langle 1\rangle][e_2/x_2\langle 2\rangle] \Rightarrow \Phi}\ \textbf{Assn} \qquad \frac{p_1 \sim_\epsilon p_2 : \Phi_1 \Rightarrow \Phi_1' \qquad p_1' \sim_{\epsilon'} p_2' : \Phi_1' \Rightarrow \Phi_2}{\vdash p_1; p_1' \sim_{\epsilon+\epsilon'} p_2; p_2' : \Phi_1 \Rightarrow \Phi_2}\ \textbf{Seq}$$

$$\frac{p_1 \sim_\epsilon p_2 : \Phi_1' \Rightarrow \Phi_2' \qquad \Phi_1 \Rightarrow \Phi_1' \qquad \Phi_2' \Rightarrow \Phi_2 \qquad \epsilon \le \epsilon'}{p_1 \sim_{\epsilon'} p_2 : \Phi_1 \Rightarrow \Phi_2}\ \textbf{Conseq}$$

Figure 3: Proving Rules from apRHL

Proof. By induction on p, we have the proof of this Lemma in detailed versions. □

Theorem 2 (Soundness)
$\forall p_1,\, p_2, \vdash p_1 \sim_\epsilon p_2 : \Phi_0 \Rightarrow \Phi$, $\forall \mathtt{m}_1, \mathtt{m}_2$ s.t Φ_0: $\mathtt{m}_1\ [\![\Phi_0]\!]\ \mathtt{m}_2$, then

$$([\![p_1]\!]_{\mathtt{m}_1})[\![\Phi]\!]^{\#(\epsilon)}([\![p_2]\!]_{\mathtt{m}_2})$$

.

Proof. By induction on the judgment $\vdash p_1 \sim_\epsilon p_2 : \Phi_0 \Rightarrow \Phi$, we have the proof of this Lemma in detailed versions. □

Theorem 3

$$\mathcal{U}(0,1]\ R^{\#(\epsilon)}\ \mathcal{U}(0,1],$$

where $R = \{(x,y) \in \mathbb{F}\times\mathbb{F} \mid \forall L^{\mathbb{R}}, R^{\mathbb{R}} \in [0,1]^{\mathbb{R}}.\ L^{\mathbb{R}} \le x \le R^{\mathbb{R}} \to (e^\epsilon L^{\mathbb{R}} \le y \le e^\epsilon R^{\mathbb{R}} \wedge y \le 1)\}$.

Proof of Theorem 3 seen in full version. □

Theorem 4

$$\mathcal{U}(0,1]\ R^{\#(\epsilon)}\ \mathcal{U}(0,1],$$

where $R = \{(v_1^{\mathbb{F}}, v_2^{\mathbb{F}}) \in \mathbb{F}\times\mathbb{F} \mid \forall L^{\mathbb{R}}, R^{\mathbb{R}} \in (0,1]^{\mathbb{R}}.(L^{\mathbb{R}} \le v_1^{\mathbb{F}} \le R^{\mathbb{R}} \to e^{-\epsilon} L^{\mathbb{R}} \le v_2^{\mathbb{F}} \le e^{-\epsilon} R^{\mathbb{R}})\}$.

Proof of Theorem 4 seen in full version. □

5 Examples

Definition 4 (Snapping Mechanism: $\mathsf{Snap}(a) : A \to \mathsf{Distr}(\mathbb{R})$**)**
Given privacy parameter ϵ, the Snapping mechanism $\mathsf{Snap}(a)$ is defined as:

$$u \overset{\$}{\leftarrow} \mathsf{unif}(0,1); s \overset{\$}{\leftarrow} \mathsf{unif}\{-1,1\}; x = f(a) + \frac{1}{\epsilon} \times s \times \ln(u); y = \lfloor x \rceil_{\Lambda}; z = \mathsf{clamp}_B(y)$$

where $f(a)$ represents a value that the query function f be evaluated over input database $a \in A$, ϵ is the privacy parameter, B is the clamping argument and Λ is the rounding argument satisfying $\lambda = 2^k$ where 2^k is the smallest power of 2 greater or equal to the $\frac{1}{\epsilon}$.

Theorem 5 (The Snap **mechanism is** $\epsilon-$**differentially private)**

seen in full version. □

References

[1] Victor Balcer and Salil Vadhan. Differential privacy on finite computers. *arXiv preprint arXiv:1709.05396*, 2017.

[2] Gilles Barthe, Marco Gaboardi, Benjamin Grégoire, Justin Hsu, and Pierre-Yves Strub. Proving differential privacy via probabilistic couplings. In *LICS 2016*.

[3] Clément Canonne, Gautam Kamath, and Thomas Steinke. The discrete gaussian for differential privacy. *arXiv preprint arXiv:2004.00010*, 2020.

[4] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating Noise to Sensitivity in Private Data Analysis. In *TCC*, 2016.

[5] Ivan Gazeau, Dale Miller, and Catuscia Palamidessi. Preserving differential privacy under finite-precision semantics. *Theoretical Computer Science*, 655:92–108, 2016.

[6] Christina Ilvento. Implementing the exponential mechanism with base-2 differential privacy. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 717–742, 2020.

[7] Ryan McKenna and Daniel R Sheldon. Permute-and-flip: A new mechanism for differentially private selection. *NIPS 2020*.

[8] Ilya Mironov. On significance of the least significant bits for differential privacy. In *CCS 2012*, 2012.