Computational Attestations of Polynomial Integrity Towards Verifiable Machine Learning

Dustin Ray - dustinray@utexas.edu[1] and Caroline El Jazmi - eljazmi@utexas.edu[1]

[1]*University of Texas at Austin*

***Abstract*—Machine-learning systems continue to advance at a rapid pace, demonstrating remarkable utility in various fields and disciplines. As these systems continue to grow in size and complexity, a nascent industry is emerging which aims to bring machine-learning-as-a-service (MLaaS) to market. Outsourcing the operation and training of these systems to powerful hardware carries numerous advantages, but challenges arise when needing to ensure privacy and the correctness of work carried out by a potentially untrusted party. Recent advancements in the discipline of applied zero-knowledge cryptography, and probabilistic proof systems in general, have led to a means of generating proofs of integrity for any computation, which in turn can be efficiently verified by any party, in any place, at any time.**

In this work we present the application of a non-interactive, plausibly-post-quantum-secure, probabilistically-checkable argument system utilized as an efficiently verifiable guarantee that a privacy mechanism was irrefutably applied to a machine-learning model during the training process. That is, we prove the correct training of a differentially-private (DP) linear regression over a dataset of 60,000 samples on a single machine in 55 minutes, verifying the entire computation in 47 seconds. To our knowledge, this result represents the fastest known instance in the literature of provable-DP over a dataset of this size. Finally, we show how this task can be run in parallel, leading to further dramatic reductions in prover and verifier runtime complexity. We believe this result constitutes a key stepping-stone towards end-to-end private MLaaS.

***Index Terms*—Differential Privacy, Machine-Learning, Linear Regression, Zero-Knowledge Cryptography, Probabilistic Checkable Proofs, ZK-STARK.**

I. Introduction

The advent of cloud computing and software-as-a-service systems illustrates an unfolding trend in which many organizations have outsourced large aspects of their computing infrastructure to specialized external service providers, who in turn provide streamlined and robust access to networked hardware and software, often at a much lower cost than self-hosting such infrastructure. With the outsourcing of sensitive computing tasks, there arises a need for security infrastructure that allows for a consumer of such services to be confident that their information is protected and handled according to their specific needs.

Machine-learning, particularly generative systems, have continued to advance at a rapid pace, demonstrating utility in a wide variety of manners. A Delloitte study [1] found that at least 50% of surveyed organizations planned to use some form of machine-learning system in 2023. The growing utility of these systems has largely coincided with their ever-increasing complexity and dependence on vast troves of data used in the learning process. Training a machine-learning system is a computationally and financially expensive process, which is often conducted using specialized hardware such as graphics processing units (GPUs). Access to this hardware is offered through a variety of commercial services, and it is the interaction with these services that presents security challenges for anyone who wishes to leverage a machine-learning system using outsourced hardware on sensitive training data.

Our penultimate result in this work shows how an MLaaS operator can employ novel cryptographic techniques with minimal hard assumptions to provide *statements of computational integrity* to a consumer, such that 1.) [2] The consumer is convinced with high probability that the work was carried out correctly and 2.) The verification of such a computation requires a proportionally small amount of work. We argue that there is limited utility of a proof system which requires the consumer of this service to perform the same amount of work as the MLaaS operator, particularly during an ML training process. We then apply this mechanism to a relatively open problem in privacy-preserving machine-learning (PPML) literature, which is the proving of correct application of differential-privacy during a training process. We discuss our design in depth in further sections.

A. Privacy-Preserving Machine-Learning

The literature arising from the intersectional disciplines of theoretical cryptography and applied machine-learning is rich with technical achievements in secure multi-party computation (MPC) protocol design towards the goal of realizing machine-learning (ML) systems that can be operated securely and privately. There are abundant[3][4][5] examples of secure privacy mechanisms and constructions that can be readily leveraged in a variety of applications. This work advances the field by exploring the robust and efficient application of cryptographically provable privacy mechanisms towards machine-learning training processes.

Historically, the privacy mechanism we select for this work is applied solely by a MLaaS operator, which then has no concrete means of proving that the mechanism actually was involved in subsequent computations as agreed to the consumer. The best the consumer can do in this situation is place trust in a possibly malicious MLaaS operator that their data was trained over privately. (We define the notion of ”private training" in further sections). This work develops a means for the consumer to act as a verifier in an Interactive-Oracle-Proof (IOP) model and perform a small amount of work to verify that the result of training could only have been obtained by correct execution of an agreed-upon set of steps.

B. Zero-Knowledge Cryptography and Proofs of Computational Integrity

This work draws from an active field of cryptography surrounding the study and application of argument systems, specifically those of the probabilistically-checkable and non-interactive variety. The celebrated results of [6] realize a proof system consisting of a series of interactions between a prover $\mathbb{P}$ and a verifier $\mathbb{V}$, in which $\mathbb{P}$ tries to convince $\mathbb{V}$ of the truth of some statement, or, in a more applied context, that the output of a computation was obtained by computing over some input. [6] shows that while $\mathbb{V}$ could re-run the entire computation and compare outputs with $\mathbb{P}$, a logarithmic-size, non-deterministic sampling of an argument provided by $\mathbb{P}$, will, with extremely high probability, be sufficient to satisfy $\mathbb{V}$ that the output was obtained correctly, and conversely otherwise. We define this protocol between $\mathbb{P}$ and $\mathbb{V}$ below.

1) Protocol Attribute Constraints: In probabilistic polynomial time, we strive for the following properties for a robust and efficient proof system under random oracle assumptions:[6]

Completeness: True statements can always be proven by a prover and will always be accepted by a verifier, except with negligible probability. Formally: For every instance-witness pair $(\mathbb{x}, \mathbb{w})$ in a relation $\mathscr{R}$, Pr $[\mathbb{V}^p(\mathbb{x}, \mathbb{P}^p(\mathbb{x}, \mathbb{w})) = 1] = 1$ for probability p taken over $\mathbb{P}$ and any randomness from $\mathbb{P}$ or $\mathbb{V}$.

Soundness: A prover should not be able to deceive a verifier into accepting a false statement as true, except with negligible probability. Formally: For every instance $\mathbb{x}$ not in the language of $\mathscr{R}$ and every malicious prover $\widetilde{\mathbb{P}}$ submitting at most a polynomial number of queries to a random oracle, $Pr\left[\mathbb{V}^p\left(\mathbb{x}, \widetilde{\mathbb{P}}^p\right) = 1\right]$ is negligible in the security parameter. [4] shows how a quantifiable lower-bound of security can be derived from the soundness parameter, which confers a configurable level of $n-$bit security directly from the security of the choice of underlying hash function.

Succinctness: We define the relationship between the work done by $\mathbb{P}$ and $\mathbb{V}$ in our construction. Strictly speaking, if $\Omega(n)$ is the amortized asymptotic upper bound of complexity for proof generation and verification respectively, then for our construction, we strictly require that $\Omega(n)_{\mathbb{P}} \leq quasi(n)$ and for $\Omega(n)_{\mathbb{V}} \leq polylog(n)$. In other words, we restrict the upper bound on the work done by the prover as quasi-linear, and the work performed by the verifier as poly-logarithmic. This definition of succinctness leads to a construction uniquely suited to the SaaS setting. An ML operator can perform a computationally expensive algorithm, the result of which can be verified for correctness in logarithmic time (and size) with respect to the computation itself. This is a significant result with important ramifications: a hypothetical computation requiring 10,000,000 steps can be verified with *only 23 queries*.

Minimal Hard Assumptions: Wherever possible, we desire the protocol to rely on minimal cryptographic hard assumptions. The result from [**?**] satisfies this requirement through the use of only a secure hash function as the underlying primitive, then paired with error-correcting codes, which ultimately lead to plausibly post-quantum-secure computational integrity statements. We do not simulate a quantum adversary in this work, but the security of our scheme follows naturally from the underlying construction [7] in use.

The previously defined parameters for our protocol are realized through the application of a form of cryptographic non-interactive proof scheme resembling a *zero-knowledge scalable transparent argument of knowledge* [8] to a machine learning algorithm known as a *differentially-private linear regression* [9]. We make a subtle divergence from the nomenclature of [8] in our usage of the term "computational integrity statements", by observing that such statements do not require perfect zero-knowledge in our scheme.

Perfect zero-knowledge in our result follows trivially from the application of [8] with surprisingly little overhead, but it is not necessarily a requirement for our protocol because the MLaaS operator and consumer are assumed in this setting to be the only parties involved in the computation, both having access to the same sensitive data and resulting machine-learning model. However, since the proof itself *is* zero-knowledge following the application of [8], it indeed reveals nothing of the model or dataset used in the computation, and could be passed to any public, untrusted party for safe and efficient verification.

C. Differential Privacy

We begin by introducing *differential privacy*, which we show can be provably applied during a machine learning training process. Differential privacy is presented formally:

Definition I.1. *(ϵ, δ) - Differential Privacy [10], [11]: A randomized algorithm $\mathscr{M}$ with domain $\mathbb{N}^{|\mathcal{X}|}$ is (ϵ, δ) - differentially private if for all $S \subseteq$ Range($\mathscr{M}$) and for all $x, y \in \mathbb{N}^{|\mathcal{X}|}$ such that $||x - y||_1 \leq 1$:*

$$Pr[\mathscr{M}(x) \in S] \leq exp(\epsilon)Pr[y \in S] + \delta \tag{1}$$

Where:

- $\mathscr{M}$: A randomized algorithm (query(db) + noise, or query(db+noise))
- S: All possible outputs of $\mathscr{M}$ that could be guessed
- x: Entries in database ($\mathbb{N}$)
- y: Entries in parallel database ($\mathbb{N} \pm 1$)
- ϵ: Maximum distance between a query on $\mathbb{N}(x)$ and the same query on $\mathbb{N}(y)$
- δ: The probability of some given information being leaked

We emphasize that this definition holds for a single query and *not* for multiple queries, and that it does not imply that an algorithm is differentially private, rather it is a measure of how much privacy is leaked to an observer given a single query on a database. The notion of a parallel database $\mathbb{N} \pm 1$ is meant to signify a database that differs by a single entry from $\mathbb{N}$. Definition 2.4 then shows that ϵ and δ are measures of by how much the probability distributions of the entries of $\mathbb{N}$ differ from $\mathbb{N} \mp 1$.

1) Epsilon (ϵ): [11] shows how ϵ is a metric of privacy loss at a differential change in a database, such as when an entry is added or removed. ϵ is defined as the maximum distance between a query on database $\mathbb{N}$ versus the same query on database $\mathbb{N} \pm 1$. It is necessary to examine the effect that differing values of ϵ has on the privacy of a given database query.

δ is a value typically defined to be an exceedingly small bias that represents the possibility that some information is leaked from a given query over a database. It is common in literature to choose δ to be $||\mathbb{N}||^{-1}$, or the inverse of the size of the given database to be processed.

(ϵ, $\delta = 0$) differential privacy indicates that an adversary cannot distinguish whether the output of an algorithm $\mathscr{M}$ was produced by processing $\mathbb{N}$ versus $\mathbb{N} \pm 1$. As the value of ϵ approaches zero, the privacy guarantees offered by such values becomes increasingly similar. Conversely, larger values of ϵ indicate that there exists an adversary that can distinguish with higher probability whether the output of $\mathscr{M}$ was obtained from either $\mathbb{N}$ or $\mathbb{N} \pm 1$.

ϵ-DP thus facilitates control over how much privacy can be "added" to a given database query. Smaller values of ϵ require that queries over $\mathbb{N}$ and $\mathbb{N} \pm 1$ produce similar outputs. In the following sections, we show precisely how this can be achieved in terms of various probability distribution mechanisms.

2) Laplace Mechanism: A numeric database query can be defined as:

$$f : \mathbb{N}^{|\mathscr{X}|} \rightarrow \mathbb{R}^k$$

Such a query maps a database to k real numbers. We proceed to define $\ell 1$ sensitivity, which is a measure of how a mapping will respond to adjacent datasets different by only a single entry:

Definition I.2 ($\ell 1$ sensitivity:)**.** *[10] The $\ell 1$-sensitivity of a function $f : \mathbb{N}^{|\mathscr{X}|} \rightarrow \mathbb{R}^k$ is:*

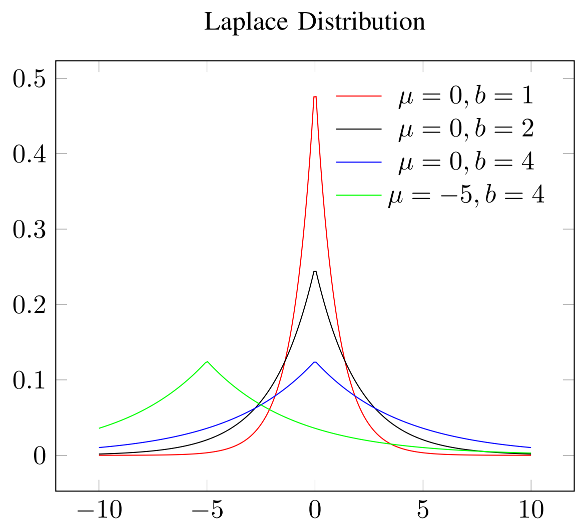
$$\Delta f = \underset{\substack{x, y \in \mathbb{N}^{|\mathscr{X}|} \\ ||x - y||_1 = 1}}{max} ||f(x) - f(y)||_1. \tag{2}$$

Definition 3.1 provides a means by measuring the maximum effect of a single entry x in a database on the output of the function f, which in turn helps to quantify the amount of noise that should be added to the output of f in order to hide the presence of x from the output of f. We proceed to define the Laplace distribution, which can be leveraged as a source of noise to incorporate into f.

Definition I.3. *The Laplace Distribution: The Laplace distribution centered at zero with scale b is the distribution with the probability density function:*

$$Lap(x|\mu, b)\frac{1}{2b}exp\left(-\frac{|x-\mu|}{b}\right) \tag{3}$$

The Laplace distribution is represented as two symmetric exponential distributions with an additional location parameter[12]. The Laplace mechanism then will run algorithm f and perturb each input with noise drawn from the Laplace distribution. The particular amount of noise to be added to an input is obtained by calculating the $\ell 1$ sensitivity of $\frac{f(\text{query})}{\epsilon}$, with the added condition that $\delta = 0$, and thus the Laplace mechanism achieves (ϵ, 0) differential privacy.



Definition I.4. *[10] The Laplace Mechanism: Given any function $f : \mathbb{N}^{|\mathscr{X}|} \rightarrow \mathbb{R}^k$, the Laplace mechanism is defined as:*

$$\mathscr{M}_L(x, f(\cdot), \epsilon) = f(x) + (Y_1 ..., Y_k) \tag{4}$$

Where all Y_i are independently identically distributed random variables drawn from Lap $\left(\frac{\Delta f}{\epsilon}\right)$. Formally stated:

The Laplace Mechanism: Given any function $f : \mathbb{N}^{|\mathscr{X}|} \rightarrow \mathbb{R}^k$, the following definition of $F(x)$ satisfies ϵ-differential privacy:

$$F(x) = f(x) + \text{Lap}(\frac{S}{\epsilon}) \tag{5}$$

Where S is the sensitivity of f, and $Lap(\frac{S}{\epsilon})$ denotes sampling from the Laplace distribution with center 0 and scale S. We use sensitivity in this setting to quantify the change in output from f when the input database changes by exactly a single entry. We have thus presented a definition of differential privacy in terms of the Laplace mechanism. If we consider our problem space to be that of measuring the accuracy of computing f, the Laplace mechanism provides a general-purpose approach in which we can achieve differential privacy.

D. (ϵ, 0) Differentially-Private Regressions Over Linear Subspaces

A linear regression is a simple form of machine-learning algorithm which attempts to find a line passing through a set of points such that the distance, referred to further as the *error* of each predicted value with respect to the line is minimized.

For a domain of values $\mathbf{x} = (x_1...x_n)^T$ mapped to a range of features $\mathbf{y} = (y_1...x_n)^T$:

$$\tilde{x} = \frac{1}{n}\sum_{i=1}^{n} x_i, \tilde{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$$

$$\text{with ncov}(x, y) = \langle \mathbf{x} - x_1, \mathbf{y} - y_1 \rangle$$

$$\text{and nvar}(x, y) = \langle \mathbf{x} - x_1, \mathbf{x} - x_1 \rangle = n \cdot \text{var}(\mathbf{x})$$

The linear regression, in matrix notation, is defined as $Y = \alpha \cdot \mathbf{x} + \beta + \mathbf{e}$:

$$\begin{bmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ \cdot \\ y_n \end{bmatrix} = \alpha \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ 1 & x_n \end{bmatrix} + \begin{bmatrix} \beta_1 \\ \beta_2 \\ \cdot \\ \cdot \\ \cdot \\ \beta_n \end{bmatrix} + \mathbf{e}$$

For a noise distribution $F_e(0, \sigma_e^2)$ and cost function "mean squared error" defined as:

$$\mathscr{L}(\theta) = \frac{1}{n}\sum_{i=0}^{n}(y_i - (\alpha x_i + \beta_i))^2$$

We select the compact $(\epsilon, 0)$-DP NoisyStats algorithm as a means of introducing differential privacy which perturbs values in the β vector with noise drawn from the Laplace mechanism. As specified in [13], we fail if the denominators of noisy versions of $ncov$ and cov become less than or equal to zero.

Algorithm 1 NoisyStats: ($\epsilon = 2$, 0)-DP [4]

```
**Data:** $\{(x_i, y_i)\}_{i=1}^n \in ([0,1] \times [0,1])^n$
**Privacy Params:** $\epsilon$
**Hyperparams:** none
Define $\Delta_1 = \Delta_2 = (1 - 1/n)$
Sample $L_1 \sim$ Lap(0,3$\Delta_1/\epsilon$)
Sample $L_2 \sim$ Lap(0,3$\Delta_2/\epsilon$)
**if** $nvar(\mathbf{x}) + L_2 > 0$ **then**
    $\tilde{\alpha} = \frac{ncov(\mathbf{x},\mathbf{y})+L_1}{nvar(\mathbf{x})+L_2}$
    $\Delta_3 = 1/n \cdot (1 + |\tilde{\alpha}|)$
    Sample $L_3 \sim$ Lap(0,3$\Delta_3/\epsilon$)
    $\tilde{\beta} = (\tilde{y} - \tilde{\alpha}\tilde{x}) + L_3$
    **return** $\tilde{\alpha} + \tilde{\beta}$
**else**
    **return** $\perp$
**end if**
```

Algorithm 1 produces β, which is now $(\epsilon, 0)$ differentially-private. This technique for private regression modeling is well-studied and common in relevant literature.[13][14] The novelty of our approach lies in the pairing of this scheme with a cryptographic computational attestation of integrity, such that our protocol definition from section B is satisfied. Perhaps most importantly, however, is how we shift the workload of the computation from the verifier to the prover, which is presented further.

E. Differential Privacy Variations

Differential-privacy is amenable to instantiation over varying hyperparameter choices. This allows for a choice of privacy budget which is suitable to the task at hand.

In this context, our research focuses on the efficacy of a novel non-interactive argument system, specifically engineered to enhance both the integrity and verification mechanisms essential for the deployment of machine learning algorithms under privacy constraints. Among the differential privacy variants evaluated, Differentially-Private Ordinary Least Squares (DP-OLS) is identified as the most appropriate privacy-preserving error estimation function for a linear regression. Other differential privacy methodologies, such as Pure Differential Privacy (Pure DP), Approximate Differential Privacy (Approximate DP) and Rényi Differential Privacy (RDP) were also considered.

Pure DP (ε-DP), in which δ = 0, is known as the strongest version of differential privacy [15]. Originating from a seminal paper by Dwork et al. [16], which introduced the initial definition of what we now recognize as Pure DP, this method adjusts the added noise based on the ℓ_1 sensitivity of the query or queries being processed. This approach [16] shows the incorporation of noise scaled to a privacy parameter, ε, ensuring that the probability of generating any particular output remains relatively unchanged even if the data of any single individual in the dataset is modified. Despite its strong privacy assurances, Pure DP typically necessitates the introduction of a higher level of noise [17], which may disproportionately compromise the utility of simpler statistical models such as OLS. In the context of implementing DP-OLS, Pure DP could be applied by adding noise directly to the OLS coefficients. While the stringent privacy guarantees of Pure DP are well-suited for highly sensitive applications, the considerable noise added can often markedly diminish the utility of the regression model.

Approximate DP ((ε, δ)-DP), while thoroughly investigated in scenarios where achieving Pure DP is complex, is less explored when the privacy loss parameter, $\delta > 0$[15]. Work by Bun, Ullman, and Vadhan [18] has identified strong lower bounds for this type of privacy, which are nearly optimal at $\delta \approx \frac{1}{n}$. This level represents the weakest privacy guarantee that still maintains practical relevance [15]. In contrast, DP-OLS adheres to more rigid differential privacy mechanisms, closely resembling those used in Pure DP. The (ϵ, δ) differential privacy model, characteristic of Approximate DP, introduces an additional parameter δ, which denotes a small probability where the privacy guarantee might not be fully upheld [19]. For linear regression models, maintaining strong privacy without significantly affecting model accuracy is crucial. While Approximate DP may be preferable in scenarios where a minor relaxation of privacy is acceptable to gain computational efficiency in more complex models, DP-OLS stands out for its strong privacy guarantees.

Rényi DP ((α, ε)-RDP), represents a natural relaxation of the standard DP model, preserving many of its core properties while introducing flexibility through parameterization based on Rényi divergence [20]. Compared with (ϵ, δ)-DP, RDP provides robust probabilistic privacy guarantees without the risk of complete privacy breaches that are permissible under the (ϵ, δ)-DP framework, which allows a δ probability of total information disclosure [20]. RDP's guarantees are contingent on outcome probabilities, thus prohibiting absolute privacy violations and preserving uncertainty even under weak parameters [20]. However, this dependency necessitates complex baseline risk assessments, especially challenging in large or dynamic datasets where probabilities are elusive. Despite its theoretical benefits, RDP's integration into cryptographic frameworks can be cumbersome due to the requisite precise assessments and their complex incorporation with cryptographic proofs. This complexity makes RDP less suited for demonstrating the feasibility of privacy-preserving computations in practical applications, especially those involving co-processing arrangements where trust and verifiability are paramount. In contrast, DP-OLS offers a simpler, direct method by embedding privacy controls within the regression algorithm [21], facilitating easier application, verification, and validation in cryptographic contexts, aligning effectively with robust, privacy-preserving research objectives.

II. ZK-Protocols for CI Statements

At this point in the research, there are two divergent paths leading to a result that satisfies the protocol constraints from section B. The first means of achieving a computational attestation of integrity for a differentially-private regression resembles that of constructing an application-specific integrated circuit (ASIC) which describes an algorithm or computation directly as a zero-knowledge circuit using standardized constraint systems (R1CS, AIR, etc). The "ASIC approach" is thus far common in ZK literature as a consequence of the field of study being relatively young, and as a result, few robust general-purpose ZK frameworks and languages exist at the time of this writing, and even fewer for a STARK-based approach.

Despite this apparent disparity, we present the RISC-Zero Zero-Knowledge Virtual Machine (ZKVM)[22]. It is an arithmetized representation of the RISC-V ISA, which can execute arbitrary Rust code targeting the RISC-V architecture. We leverage the RISC-Zero ZKVM towards measuring the performance of a relatively straightforward implementation of a linear regression paired with a differentially-private additive-noise mechanism. This choice has two distinct advantages over describing the same design as a "zk-ASIC":

- The RISC-Zero ZKVM accepts arbitrary Rust as input, leading to not only a substantial reduction in programming complexity as far as algorithm description is concerned, but to significant gains in productivity and application expressivity as well.
- As a virtual representation of a small RISC-V machine, the RISC-Zero ZKVM is compatible with essentially any Rust crate which can be targeted to the RISC-V ISA. Out-of-the-box, this platform is immediately usable by a large number of existing libraries, types and software in the Rust ecosystem.

Overhead can be incurred by executing certain applications on a general-purpose computing platform, and this is also true on non-ZK computing hardware. Despite the possible overhead incurred by leveraging a general-purpose ZK computing platform, it is worth noting that the RISC-Zero ZKVM realizes *de-facto* the state-of-the-art in ZK-STARK literature, employing techniques such as preprocessing, the DEEP soundness enhancement to the FRI protocol, Metal/CUDA integration, and composition of proving systems for small(and even constant-size) proof sizes. This results in a platform that exhibits remarkable performance and robust security guarantees, despite the substantial difference in complexity between an arithmetized ISA and a simple differentially-private regression circuit.

III. Observations

A. Methodologies and Results

This subsection defines the experimental setup of this research. We run our differentially-private regression within a single RISC-Zero ZKVM on a Macbook Pro M1 Max with 32gb of RAM. We use the "Kaggle Healthcare Dataset" which contains a record of patients admitted to hospital care, several factors regarding their health, and the resulting insurance amount billed for the visit. The focus of this work remains centered on the performance of the protocol in terms of runtimes, and not on the prediction accuracy of the resulting model. Thus we augment the size of the dataset from 10,000 to 60,000 samples using random copies of existing samples. We then isolate the columns in the data to a single independent variable and a single explanatory variable; Age vs Insurance Cost. We then produce a regression model which predicts patient insurance cost of a hospital visit versus their age.

1) Training and Experiment Design: We learn two hypotheses: the first is based on a simple "ordinary least squares (OLS)" error estimator, and the second is a differentially private version of OLS (DP-OLS). We observe that the DP-OLS hypothesis initially diverges significantly from the OLS hypothesis when trained on few data samples. This is an expected result given how the data is perturbed in the DP-OLS training, and with few samples to learn from, the noisy hypothesis should display a strong divergence from the OLS estimator.

As larger volumes of samples are trained over, we observe both estimators converging to roughly the same hypothesis, which is again the expected result. This indicates that DP-OLS, despite the noise perturbation, is successfully learning approximately the same hypothesis as the OLS regression. The training procedure is conducted in a "batched" fashion, since we are limited by the numbers of samples we can process on our GPU at a given iteration. We discuss this further in the next subsection.

2) Performance: Performance metrics that we are interested in include "Proof time versus Dataset Size", "Verification Time vs Dataset Size", and "DP vs OLS Model Accuracy". We initially run the ZKVM over a smaller subset of 1200 total data samples using our CPU as the proving hardware. By recording the proof time at regular intervals, we observe that proof time versus dataset time appears to scale roughly linearly, while the verification time appears to scale roughly logarithmically, which is the expected result.

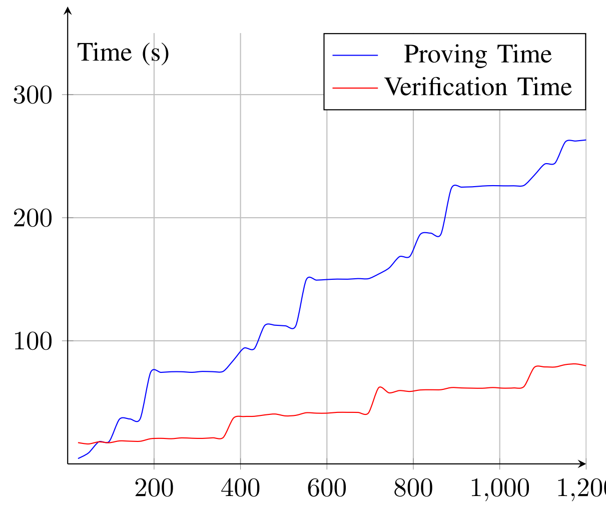


Fig. 1. Proving and Verification Time vs. Dataset Size Without CUDA

B. *Prediction Accuracy: Non-Private vs. Private*

The below chart measures the overall accuracy of the predictions made by both the DP and the non-DP estimators. The difference between mean absolute error for both regressions is extremely low after training over 60,000 samples, indicating that the DP regression makes predictions almost identical to that of the non-DP regression. Further, the standard error of both regressions is measured, with an extremely small delta of 0.00001. This indicates that the DP regression slope will make predictions well within the standard error of the non-DP regression. These results both strongly suggest that despite the noisy training data, the DP estimator converges to almost exactly the same model as the non-DP estimator.

MAE OLS	DP-MEA OLS	Δ
9495.09	9495.12	0.03
Slope SE	DP-Slope SE	Δ
0.83202	0.83201	0.00001

TABLE I. Accuracy Difference Between Non-private vs DP Model

C. *Scaling Up*

The RISC-Zero ZKVM is packaged with a CUDA-compatible GPU-compute backend which dramatically improves the computation time of our DP-regression training. This comes at the cost of utilizing nearly all of the memory of the graphics unit used in testing (a relatively modest NVIDIA GeForce RTX 4070 Laptop GPU with 8gb of available video RAM). We observe empirically with this hardware setup that we can run the entire DP-regression training over 175 samples before exceeding available RAM. We mitigate this limitation by dividing the dataset into 343 blocks instead, verifying each block with a single ZKVM in sequence. We obtain a DP-proving runtime for our regression training of 9 seconds for every 175 samples, or roughly 19 samples per second. We use this result to sequentially prove a differentially-private regression training over 60,000 (x, y) samples in approximately 55 minutes with a single GPU.

It is possible to deploy many of these ZKVMs in parallel, and as such, proving time becomes a function of each new node joining the system, up to an optimal number of nodes. When considering the overhead of instantiating the ZKVM and generating the proofs, we find that 175 samples is the optimal number of samples for each ZKVM to prove. If 60,000 samples are evenly divided across a network of 343 single GPUs, the entire dataset could be proven *in less than 10 seconds*. While 343 nodes in such a network may appear to be a high number, the Ethereum blockchain network is recorded [23] as having a compute power of 603,000 GHz on average at any given time in 2021. Our GPU has a clock speed of 1.61 GHz [24], suggesting that in 2021, there existed a distributed network of approximately 374,534 available ”GPU units” available for compute at any given time.

Lastly, the task of aggregating the individually computed models into a single regression remains. This step requires O(c) work with respect to the number of nodes in the system and is carried out by the verifier. It is assumed at this point that the verifier, as in our experiments, has already carried out verification of each received proof and can trust with high probability the correctness of each model it has received from a node. Each linear regression model is represented by the equation:

$$y = \beta_1 x + \beta_0$$

where β_1 is the slope and β_0 is the intercept. To combine models from multiple parties, the slopes (β_1) and intercepts (β_0) of these models are averaged.

1. Averaging Slopes:

$$\overline{\beta_1} = \frac{1}{n}\sum_{i=1}^{n} \beta_{1i}$$

where n is the number of parties (or models), and β_{1i} is the slope of the i-th model.

2. Averaging Intercepts:

$$\overline{\beta_0} = \frac{1}{n}\sum_{i=1}^{n} \beta_{0i}$$

where β_{0i} is the intercept of the i-th model.

The combined model is then given by:

$$y = \overline{\beta_1} x + \overline{\beta_0}$$

IV. Comparison To Other Works

We assess the literature surrounding the topic of provable-DP to be relatively sparse. The closest result to ours is found in **confidential-DP**[25]. To the best of our understanding, this work is derived from a zero-knowledge construction contained within the EMP toolkit [26], itself based on an argument system referred to as ”Wolverine” [27]. To the best of our knowledge, confidential-DP does not report a verifier runtime, and so we are left to assess the complexity of the underlying argument system in use. This implies that by [27], confidential-DP obtains an $O(n)$ prover and an $O(n)$ verifier. By contrast, our work obtains an $O(n)$ prover and an $O(polylog(n))$ verifier. confidential-DP is an interactive argument system, requiring the proving and verifying parties to be online and communicating during the execution of the protocol. Our result is non-interactive, the prover and verifier only exchange a single message to instantiate the protocol, and a single message from the prover is sent containing the result and proof of correctness. Both results are zero-knowledge. The verifying party need not be the same who initiated the protocol. A proof of correctness is zero-knowledge and reveals nothing except that the argument is correct.

Protocol	**DP**	**Model**	**Prover**	**Verifier**	**Online**
This work	NoisyStats, ($\epsilon = 2$, 0)	Linear	$O(n)$	$O(\log n)$	No
confidential-DP	DP-SGD, $\epsilon = 0.55$, $\delta = 10^{-5}$	Logistic	$O(n)$	$O(n)$	Yes

TABLE II. Summary of Protocols

Other observed differences between results are that confidential-DP classifies images, while our model predicts labels from data points represented as pairs of single floating-point numbers. Confidential-DP reports a total proving time of 100 hours over the CIFAR10/MNIST dataset, while we observe a 55 minute proving time over our Kaggle dataset. We remark however that runtime in this context is not a meaningful comparison of performance between the two results, since the tasks are performed over different data with different model architectures.

Lastly, confidential-DP proves only the DP portion of their training. They first extract features from the training data with a deep network, then classify those features with a DP-logistic regression. This is done because describing an entire deep network (particularly the computation graphs and gradients for back-propagation) as a ZK-circuit remains a highly non-trivial and contrived task. Our work proves the correctness of the entire training process, however our training regimen is a comparatively simple model consisting of a compact number of operations that are readily programmed into the ZKVM.

V. Conclusion

This work shows a robust and efficient means of cryptographically proving and verifying the correct execution of an agreed-upon machine-learning training process. We believe our construction results in an attestation of integrity which shows machine-learning algorithms, (albeit simple algorithms, for the time-being) are well-suited to this particular form of non-interactive argument system, demonstrating a practical application of introducing an ”asymmetry” into a heavy computation in order to leverage powerful hardware which may exist in a different physical or temporal location. Our result shows how a verifying party can apply this framework to efficiently obtain an irrefutably correct machine-learning model from an untrusted but powerful outside source.

Argument systems of this variety facilitate what seems at first to be a counter-intuitive, yet intriguing and powerful result; by working in concert with a prover, the verifier has learned the exact same machine-learning model, but appears to have done so with only $O(polylog(n))$ work with respect to the dataset size[6]. As the dependence on outsourced ML hardware continues to grow, we anticipate the need for secure ”co-processing” solutions of this nature to expand in kind. We believe this work shows that state-of-the-art ZKVM constructions are equipped to play a unique role in the growth of privacy-preserving machine-learning.

We believe that this research plays a vital role in the development of completely private end-to-end machine-learning, in which distinct and distrustful parties (model operators and consumers) may interact with each other in complete privacy. During the inference phase of this research, the model itself and the data to be classified are revealed out of necessity. This could be hypothetically be remedied by recent advancements in the field of fully-homomorphic encryption. [28] demonstrates the practical feasibility of conducting regressions over encrypted data, while [29] leverages the RISC-Zero ZKVM to prove fully-homomorphic computations over encrypted ciphertexts. Future work targets the understanding of the costs of fully homomorphic and provable differentially-private regressions and classifications using the ZKVM, along with other frameworks and tools. □

References

[1] Deloitte, “Generative Artificial Intelligence — www2.deloitte.com,” https://www2.deloitte.com/us/en/pages/consulting/articles/generative-artificial-intelligence.html, [Accessed 20-09-2023].

[2] D. Ray, “capy2vML: Provably-secure differentially-private machine learning training,” https://github.com/drcapybara/capy2vML, 2024.

[3] J. Lee and D. Kifer, “Scaling up differentially private deep learning with fast per-example gradient clipping,” 2020. [Online]. Available: https://arxiv.org/abs/2009.03106

[4] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, “Deep learning with differential privacy,” in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS ’16. New York, NY, USA: Association for Computing Machinery, 2016, p. 308–318. [Online]. Available: https://doi.org/10.1145/2976749.2978318

[5] “Applying differential privacy to large scale image classification,” Feb 2022. [Online]. Available: https://ai.googleblog.com/2022/02/applying-differential-privacy-to-large.html

[6] J. S. A. Y. Shafi Goldwasser, Guy Rothblum, “ECCC - TR20-058 — eccc.weizmann.ac.il,” [Accessed 20-09-2023].

[7] E. Ben-Sasson, A. Chiesa, and N. Spooner, “Interactive oracle proofs,” Cryptology ePrint Archive, Paper 2016/116, 2016, https://eprint.iacr.org/2016/116. [Online]. Available: https://eprint.iacr.org/2016/116

[8] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev, “Scalable, transparent, and post-quantum secure computational integrity,” Cryptology ePrint Archive, Paper 2018/046, 2018, https://eprint.iacr.org/2018/046. [Online]. Available: https://eprint.iacr.org/2018/046

[9] D. Alabi, A. McMillan, J. Sarathy, A. Smith, and S. Vadhan, “Differentially private simple linear regression,” 2020.

[10] C. Dwork and A. Roth, “The algorithmic foundations of differential privacy,” *Foundations and Trends® in Theoretical Computer Science*, vol. 9, no. 3-4, p. 211–407, 2013.

[11] S. Fathima, “Differential privacy-noise adding mechanisms,” Oct 2020. [Online]. Available: https://becominghuman.ai/differential-privacy-noise-adding-mechanisms-ede242dcbb2e

[12] “Laplace distribution,” May 2022. [Online]. Available: https://en.wikipedia.org/wiki/Laplace_distribution

[13] D. Alabi, A. McMillan, J. Sarathy, A. D. Smith, and S. P. Vadhan, “Differentially private simple linear regression,” *CoRR*, vol. abs/2007.05157, 2020. [Online]. Available: https://arxiv.org/abs/2007.05157

[14] “Implement differential privacy with tensorflow privacy nbsp;: nbsp; responsible ai toolkit.” [Online]. Available: https://www.tensorflow.org/responsible_ai/privacy/tutorials/classification_privacy

[15] T. Steinke and J. Ullman, “Between pure and approximate differential privacy,” 2015.

[16] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006*, ser. Lecture Notes in Computer Science, vol. 3876. Springer, 2006, pp. 265–284. [Online]. Available: https://iacr.org/archive/tcc2006/38760266/38760266.pdf

[17] S. Das and S. Mishra, *Advances in Differential Privacy and Differentially Private Machine Learning*. Springer Nature Singapore, 2024, p. 147–188. [Online]. Available: http://dx.doi.org/10.1007/978-981-97-0407-1_7

[18] M. Bun, J. Ullman, and S. Vadhan, "Fingerprinting codes and the price of approximate differential privacy," 2018.

[19] Y.-X. Wang, J. Lei, and S. E. Fienberg, "Learning with differential privacy: Stability, learnability and the sufficiency and necessity of erm principle," 2016.

[20] I. Mironov, "Rényi differential privacy," in *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*. IEEE, Aug. 2017. [Online]. Available: http://dx.doi.org/10.1109/CSF.2017.11

[21] O. Sheffet, "Differentially private ordinary least squares," 2017.

[22] RiscZero, "About — risczero.com," https://www.risczero.com/about, [Accessed 20-09-2023].

[23] M. M. Spreadsheet, "Mining model spreadsheet," https://docs.google.com/spreadsheets/d/138M4R1-_zS-OLBsl2VJeN_anfTSCRCFc6EguYUVG-yA/edit#gid=1339763553, [Accessed 20-09-2023].

[24] NotebookCheck, "Notebookcheck," https://www.notebookcheck.net/NVIDIA-GeForce-RTX-4070-Laptop-GPU-Benchmarks-and-Specs.675690.0.html, 2024, [Accessed 4-18-2024].

[25] S. et al, "Confidential-dpproof : Confidential proof of differentially private training," https://openreview.net/pdf?id=PQY2v6VtGe, 2024, [Accessed 4-18-2024].

[26] W. et al, "emptoolkit," https://github.com/emp-toolkit/emp-zk, 2023, [Accessed 4-18-2024].

[27] C. e. a. Weng, "Wolverine: Fast, scalable, and communication-efficient zero-knowledge proofs for boolean and arithmetic circuits," https://eprint.iacr.org/2020/925.pdf, 2020, [Accessed 4-18-2024].

[28] Zama, "Concrete ml," https://github.com/zama-ai/concrete-ml, 2024, [Accessed 20-09-2023].

[29] L. w. Weikeng Chen, Research Partner, "Tech deep dive: Verifying fhe in risc zero, part i," https://l2ivresearch.substack.com/p/tech-deep-dive-verifying-fhe-in-risc, 2024, [Accessed 4-18-2024].