Zero-knowledge proof of training: a comparative study

Oussama Jeddou
Department of Computer Science
Technical University of Munich
Email: oussamajaddou@gmail.com

***Abstract*—Ensuring the correctness of machine learning (ML) model training without revealing the underlying dataset or the final model parameters presents a critical challenge in privacy-preserving AI. This issue is especially significant when models are trained on sensitive or proprietary data, raising concerns about data security and intellectual property protection. Existing methods for verifying training integrity often compromise privacy by exposing either the dataset or the model itself, underscoring the need for a robust solution.**

Zero-Knowledge Proof of Training (zkPoT) emerges as a promising approach, enabling a prover to demonstrate the correctness of a training process without revealing any additional information. In this paper, we analyze and compare three different protocols for non-interactive zkPoT that require single interaction between the prover and verifier. We explore the improvements offered by each protocol, their respective use cases, and present new studies aimed at optimizing these protocols. Our study seeks to address the question: can a zero-knowledge proof of training protocol be designed to independently verify, in a single interaction ,the correctness of model training in a short time and small size, while maintaining the privacy of the model?

***Index Terms*—zkPoT, succinct zero-knowledge proof, scalable zero-knowledge proof, Bulletproofs**

I. Introduction

In the fields of machine learning (ML) and artificial intelligence (AI), ensuring that training processes are correct while keeping data and models private has become a major challenge. Since models often rely on sensitive data, it is vital to maintain their integrity without exposing any confidential information. This is important for protecting both data security and intellectual property. Unfortunately, traditional methods for verification usually need access to the dataset or the trained model, which raises big privacy concerns.

A promising solution comes from zero-knowledge proofs (ZKPs) [2], a cryptographic method introduced by Goldwasser et al. [1]. ZKPs allow one party, the prover, to convince another party, the verifier, that a statement is true without sharing any extra information. The catch is that regular ZKPs require back-and-forth communication between the prover and verifier [3], which can be cumbersome in real-life situations.

To address these limitations, Blum et al. introduced Non-Interactive Zero-Knowledge Proofs (NIZKPs) [4], which allow a prover to generate a proof that can be verified in a single interaction and subsequently revalidated by other verifiers [3]. This paper explores the application of ZKPs in ensuring the integrity of ML model training. Specifically, we examine three prominent families of non-interactive zero-knowledge proof of training protocols:

- zkSNARKs (section II-B): Known for their compact proof size and efficient verification, zk-SNARKs are widely deployed in applications requiring minimal computational overhead [5], [38], [43]–[46].
- zkSTARKs (section II-C): Offering a transparent setup and post-quantum security, zk-STARKs are ideal for scalable applications [3], [10], [12], [13].
- Bulletproofs (section II-D): Distinguished by their elimination of a trusted setup, Bulletproofs balance proof compactness with relatively higher verification costs [2], [19], [21], [22].

By providing a comparative analysis of these protocols, we aim to offer insights into their strengths, trade-offs, and optimal applications for privacy-preserving ML. This investigation contributes to the development of robust cryptographic frameworks that balance privacy, efficiency, and scalability in the verification of training processes.

A. *Motivation*

The integration of ZKP protocols in AI systems is driven by the need for privacy-preserving solutions in training models. Key applications include:

- **Federated Learning:** Ensuring that client-side computations are performed correctly without revealing local data.
- **Proof-of-Training:** Providing verifiable guarantees that a model was trained as specified, using designated datasets and algorithms.
- **Decentralized AI:** Enhancing trust in collaborative AI systems by enabling transparent validation of model updates.

B. *Road Map*

The rest of this paper is structured as follows: Section II provides background on zero-knowledge proof of training concepts and protocol families. Section III outlines the methodology used for benchmarking. Sections IV and V present detailed analyses of zkSNARKs, zkSTARKs, and Bulletproofs, followed by comparative results. Finally, Section VI explores future directions, and Section VII concludes with insights into the role of ZKPs in privacy-preserving AI.

II. Background

A. *Zero-Knowledge Proofs (ZKPs)*

Zero-Knowledge Proofs are cryptographic constructs that allow one party (the prover) to demonstrate the validity of a statement to another party (the verifier) without disclosing any additional information about the statement [1], [2]. A ZKP satisfies three fundamental properties:

- **Completeness**: ensures that if the statement is true, an honest prover can always convince an honest verifier to accept the proof. This is a fundamental property of interactive proofs, ensuring that valid statements are recognized as such by the verifier [6].
- **Soundness**: ensures that if the statement is false, no dishonest prover can convince the verifier to accept a proof. This property is essential for maintaining the integrity of the proof system. [3], [7]
- **Zero-Knowledge**: ensures that when a prover demonstrates a statement to a verifier, the verifier learns nothing beyond the validity of that statement. This means that the proof does not reveal any additional information about the inputs used to generate it [2].

The practical applications of ZKPs span multiple domains, including secure authentication, blockchain technologies, and privacy-preserving machine learning.

ZKP Protocol Families:
Three prominent families of Zero-Knowledge Proof protocols—zkSNARKs, zkSTARKs, and Bulletproofs—have emerged as solutions for various real-world applications. Each of these protocols possesses unique characteristics that make it well-suited for specific use cases.

Every ZKP protocol is composed of three key components: the setup phase, the prover's role, and the verifier's role.

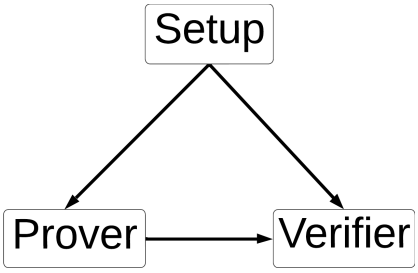

Fig. 1. Structure of zkPoT protocols

B. *zkSNARKs*

zkSNARKs, or Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge, are cryptographic protocols that enable a prover to convince a verifier that they possess a piece of information satisfying a certain computational problem without revealing the information itself. zkSNARKs are particularly valued for their succinctness, meaning the proof size is small regardless of the complexity of the computation, and for their efficient verification. zkSNARKs rely heavily on cryptographic primitives such as polynomial commitments [52], bilinear pairings [51], and elliptic curve cryptography (see section II-D). They are widely used in blockchain systems and other applications requiring efficient and private computations [7].

Mathematical foundations:

1- Common Reference String (CRS): is a key element in the zk-SNARK process, consisting of parameters used for creating proofs and verifying them, and it is generated during a trusted setup phase using secret random variables known as "toxic waste." These variables must be securely destroyed after use to prevent misuse in creating undetectable false proofs. Traditionally, a single trusted party handled CRS generation, but this method risks malicious intent or improper disposal of the toxic waste. To mitigate these risks, multi-party protocols were developed, distributing the setup among multiple participants to ensure no single party has complete knowledge of the witness, thereby enhancing system trust. However, the security of such setups depends on the assumption that at least one participant is honest, leaving the system vulnerable to collusion. [3], [8]

2- The Structured Reference String (SRS): is a vital element in zero-knowledge proof systems, especially in zkSNARKs, where it underpins the processes of proof generation and verification while ensuring soundness and security. Designed to be universal and updatable, the SRS can support statements about circuits of a certain bounded size and be updated by new participants without compromising its integrity, offering flexibility for practical applications. Its construction typically involves a trusted setup phase with secret random values, and its security hinges on the assumption that at least one participant in the setup is honest. In zkSNARKs, the SRS is crucial for producing succinct and verifiable proofs, enabling efficient communication and computation in cryptographic protocols. [9]

The zkSNARK process involves a series of well-defined phases, each with specific roles and computations. These phases ensure that the prover can demonstrate the correctness of a computation without revealing any private information. The process can be broken down as follows:

I. Setup Phase (Trusted Setup)

1) Trusted Setup:
   - Generate a *Structured Reference String (SRS)*:
     - Common Reference String (CRS): Public parameters for encoding the computation.
     - Secret Reference String (SRS): Private randomness for proof generation.
2) Commit to Inputs:
   - Compute cryptographic commitments:

$$C_D = Commit(D), \quad C_\theta = \text{Commit}(\theta_{\text{final}})$$

where:

- D: Private dataset.
- θ_{final}: Final trained model parameters.

3) Define Public Training Objective:
   - Agree on a public objective L_{public}, such as a maximum loss or accuracy threshold.
4) Circuit Construction:
   - Encode the training process A into zkSNARK-friendly constraints:
     - Input Consistency: Verify $C_D = \text{Commit}(D)$ and $C_\theta = \text{Commit}(\theta_{\text{final}})$.
     - Training Process Validation: Simulate forward pass, loss calculation, and gradient updates.
     - Objective Validation: Verify $L(\theta_{\text{final}}, D) \leq L_{\text{public}}$.

II**. Proof Generation Phase**

1) Encode Training Steps:
   - Simulate the training process A:

$$\begin{aligned}&\text{Forward pass: } y = f_\theta(x), \quad \forall x \in D\\&\text{Gradient updates: } \theta_{\text{new}} = \theta - \eta \cdot \nabla L\\&\text{Final loss: } L(\theta_{\text{final}}, D)\end{aligned}$$

2) Generate zkSNARK Proof:
   - Use the zkSNARK circuit to compute a proof P that:
     - C_D and C_θ match the private dataset and model parameters.
     - The training process A was executed correctly.
     - $L(\theta_{\text{final}}, D) \leq L_{\text{public}}$.

III**. Proof Verification Phase**

1) Validate Commitments:
   - Verify:

$$C_D = \text{Commit}(D), \quad C_\theta = \text{Commit}(\theta_{\text{final}})$$

2) Verify Training Process:
   - Use zkSNARK verification to ensure:
     - Forward passes and gradient updates were executed correctly.
     - The public objective L_{public} is satisfied.
3) Check Public Objective:
   - Ensure:

$$L(\theta_{\text{final}}, D) \leq L_{\text{public}}$$

4) Output Verification Result:
   - If all checks pass, output **Valid**.
   - Otherwise, output **Invalid**.

C. zkSTARKs

zkSTARKs, or Zero-Knowledge Scalable Transparent Arguments of Knowledge, are cryptographic protocols designed to prove the correctness of computations while ensuring zero-knowledge. They stand out for their transparency and scalability, making them particularly well-suited for applications requiring high security and efficiency [3], [10].

Mathematical Foundations:

1- Interactive Oracle Proofs (IOPs): zkSTARKs leverage IOPs to enable a prover to interactively convince a verifier about the validity of a statement. The prover encodes the computation into a low-degree polynomial and provides responses to the verifier's queries to validate the claim. IOPs allow for efficient querying and verification without requiring the prover to transmit the entire proof data, minimizing computational overhead for both parties [13], [14].

2- Arithmetization and Polynomial Commitments: The computation to be proven is converted into a sequence of states, which are then encoded as a trace table T. Each state transition is described by polynomial constraints $P(x)$, validating the relationships between consecutive states. Using Lagrange interpolation, the trace table and constraints are converted into low-degree polynomials. Polynomial commitments ensure that the prover securely commits to the trace, allowing the verifier to validate these commitments efficiently during verification [13].

3- Merkle Root: is the topmost hash in a Merkle tree, a cryptographic data structure that provides a compact and efficient way to verify the integrity of a dataset. It is computed by recursively hashing pairs of data chunks until a single hash remains at the root. The Merkle root serves as a cryptographic fingerprint of the dataset, allowing verification with minimal data using Merkle proofs. This ensures data integrity, compact representation, and efficient verification, making it widely used in blockchains, data integrity systems, and cryptographic protocols [2], [13].

Below is a detailed explanation of their workflow:

I **. Setup Phase**

1) Define the Training Problem:
   - The Prover identifies the training process A, which includes:
     - Forward passes on the dataset D.
     - Gradient computation and updates to model parameters θ.
     - A public training objective L_{public}, such as a loss threshold or accuracy goal.
2) Commit to Dataset and Model Parameters:

- Compute commitments:

$$C_D = \text{Merkleroot}(D), \quad C_\theta = \text{hash}(\theta_{\text{final}})$$

where:

  - C_D: Commitment to the private dataset D using a Merkle tree.
  - C_θ: Hash of the final model parameters θ_{final}.

3) Arithmetization:
   - Transform the training process into polynomial constraints:
     - Represent forward passes, gradient updates, and loss evaluations as polynomials.
     - Ensure consistency of intermediate states (e.g., weight updates).
4) Generate Public Randomness:
   - Use the Fiat-Shamir heuristic to derive randomness for proof generation without requiring a trusted setup.

II. **Proof Generation Phase**

1) Simulate the Training Process:
   - Execute the training algorithm A:

$$\begin{aligned}&\text{Forward pass: } y = f_\theta(x), \quad \forall x \in D\\ &\text{Gradient updates: } \theta_{\text{new}} = \theta - \eta \cdot \nabla L\\ &\text{Final loss: } L(\theta_{\text{final}}, D)\end{aligned}$$

2) Construct Merkle Tree Proofs:
   - Commit to the dataset D by creating a Merkle tree:

$$C_D = \text{Merkleroot}(D)$$

   - Store the Merkle tree leaf nodes and ensure individual samples can be verified with Merkle proofs.
3) Polynomial Proofs:
   - Encode training steps as polynomial constraints:
     - State transitions: Ensure consistency between model updates.
     - Objective constraint: Verify $L(\theta_{\text{final}}, D) \leq L_{\text{public}}$.
4) Low-Degree Test:
   - Verify that all polynomials used in the computation are of low degree.
5) Generate zkSTARK Proof:
   - Combine polynomial commitments, Merkle proofs, and randomness to produce a zkSTARK proof P, demonstrating:
     - Correct execution of the training process A.
     - Consistency of commitments C_D and C_θ.
     - Satisfaction of the public objective L_{public}.

III. **Proof Verification Phase**

1) Verify Dataset Commitment:
   - Check the commitment $C_D = \text{Merkleroot}(D)$.
   - Use Merkle proofs to validate individual data samples against the root C_D.
2) Verify Model Commitment:
   - Confirm the hash of the final model parameters:

$$C_\theta = \text{hash}(\theta_{\text{final}})$$

3) Check Polynomial Constraints:
   - Verify that:
     - Polynomial constraints (state transitions, loss computation) are satisfied.
     - Intermediate states in the training process are consistent.
4) Verify Low-Degree Property:
   - Confirm that all polynomials in the proof have low degree.
5) Output Verification Result:
   - If all checks pass, output **Valid**.
   - Otherwise, output **Invalid**. [6], [7]

D. Bulletproofs

Bulletproofs are a class of succinct, non-interactive zero-knowledge proofs that do not require a trusted setup. Bulletproofs can be used in a Proof of Training context to create compact, privacy-preserving proofs that a model has been trained under specific conditions. [3], [19], [20] By leveraging cryptographic commitments, specifically Pedersen commitments [23], Bulletproofs maintain value confidentiality while enabling verification . They rely on inner-product arguments, which allow efficient range proofs without revealing the actual values . Their logarithmic proof sizes (see section IV) relative to computational complexity provide scalability, making them a significant improvement over traditional proof systems [2].

Mathematical Foundations:

Bulletproofs rely on three primary mathematical constructs: inner-product arguments, Pedersen commitments, and elliptic curve cryptography.

1. Inner-Product Arguments: Bulletproofs utilize inner-product arguments to achieve logarithmic proof sizes. In each round, the prover and verifier interact to reduce the dimensionality of the problem, effectively compressing the representation of the computation. This iterative process allows Bulletproofs to handle complex computations while maintaining compact proofs [21].

2. Pedersen Commitments: A core component of Bulletproofs is the use of Pedersen commitments. These commitments are homomorphic, meaning arithmetic operations can be performed on committed values without revealing the values themselves. This property is crucial for enabling private computations in Bulletproofs [23].

3. Elliptic Curve Cryptography: Bulletproofs depend on the hardness of discrete logarithm problems in elliptic curve

groups for their security. Elliptic curves allow for efficient implementations of the cryptographic operations used in Bulletproofs, making them scalable and secure. However, this reliance on elliptic curves also means Bulletproofs are not post-quantum secure, as quantum computers could break elliptic curve cryptography [24].

The Bulletproofs protocol can be divided into the following key steps:

I. Setup Phase

1) Define the Training Problem:
   - The Prover defines the training process A, which includes:
     - Forward passes on the dataset D.
     - Gradient computation and updates for model parameters θ.
     - A public training objective L_{public}, such as a final loss or accuracy threshold.
2) Commit to Dataset and Model Parameters:
   - Use a Pedersen commitment [23] scheme to commit to the dataset D and the model parameters θ:

$$C_D = g^D h^r, \quad C_\theta = g^\theta h^s$$

   where:
   - g and h: Public generators.
   - r, s: Random values (blinding factors).
   - D: Dataset split into smaller elements (e.g., samples).
   - θ: Model parameters (weights).
3) Express Training Constraints:
   - Represent the training process as a set of constraints:
     - Forward pass and loss calculation: $y = f_\theta(x)$, $L(\theta, D)$.
     - Gradient updates: $\theta_{\text{new}} = \theta - \eta \cdot \nabla L$.
     - Training objective: Verify $L(\theta_{\text{final}}, D) \leq L_{\text{public}}$.
   - These constraints will later be encoded as range proofs.

II. Proof Generation Phase

1) Simulate the Training Process:
   - Execute the training algorithm A over D and θ_{init}:

$$\begin{aligned} &\text{Forward pass: } y = f_\theta(x), \quad \forall x \in D \\ &\text{Gradient updates: } \theta_{\text{new}} = \theta - \eta \cdot \nabla L \\ &\text{Final loss: } L(\theta_{\text{final}}, D) \end{aligned}$$

2) Construct Range Proofs:
   - Use Bulletproofs to generate range proofs for the following:
     - Model Parameters: Prove that θ values lie within acceptable ranges (e.g., to ensure valid weight updates).
     - Loss Values: Prove that $L(\theta_{\text{final}}, D) \leq L_{\text{public}}$ without revealing the exact value of L.
3) Link Commitments to Range Proofs:
   - Prove consistency between commitments and the training process:

$$C_D = g^D h^r, \quad C_\theta = g^\theta h^s$$

   - Ensure that D and θ satisfy all training constraints.
4) Generate Bulletproof Proof:
   - The Prover generates a single Bulletproof P that includes:
     - Proof of valid range constraints for θ and L.
     - Consistency of commitments C_D and C_θ with the dataset D and model parameters θ.
     - Correct execution of the training process A.

III. Proof Verification Phase

1) Verify Commitments:
   - Check that the commitments $C_D = g^D h^r$ and $C_\theta = g^\theta h^s$ are valid.
2) Validate Range Proofs:
   - Verify that the Prover's range proofs satisfy:
     - θ values lie within valid ranges.
     - $L(\theta_{\text{final}}, D) \leq L_{\text{public}}$.
3) Check Training Constraints:
   - Confirm that the Prover's commitments are consistent with the training process constraints.
   - Ensure all constraints (forward pass, gradients, loss) are satisfied.
4) Output Verification Result:
   - If all checks pass, output **Valid**.
   - Otherwise, output **Invalid**.

E. Applications in Proof-of-Training

Zero-Knowledge Proofs (ZKPs) have transformative applications in the context of proof of training within artificial intelligence (AI), addressing critical issues of privacy, trust, and ownership in model development and deployment. In **federated learning**, ZKPs allow devices to cryptographically prove the correctness of their local computations during training without revealing sensitive data, ensuring privacy preservation while maintaining the integrity of the distributed learning process . For **decentralized AI** systems, ZKPs provide robust mechanisms for validating the correctness of collaboratively trained models without disclosing private data or model parameters. This transparency fosters trust among participants and ensures the reliability of shared computations [5], [6]. Additionally, ZKPs are instrumental in establishing **model ownership**, enabling developers to prove that a model was trained on a specific dataset while safeguarding intellectual property. This capability prevents unauthorized claims over models and ensures rightful attribution to their creators . These applications underline the critical role of ZKPs in secure and verifiable AI training processes [5], [6], [25].

F. Expected Security and Performance Considerations

Each ZKP (Zero-Knowledge Proof) protocol is designed to balance security, scalability, and efficiency. zkSNARKs are highly efficient, making them a great choice for applications that require quick proof generation. However, they rely on a trusted setup, which can introduce vulnerabilities and potentially compromise security in adversarial environments. zkSTARKs, on the other hand, are designed to be post-quantum secure, offering robust protection against future quantum threats. The trade-off is their larger proof sizes, which can make them less practical for networks with limited bandwidth. Bulletproofs stand out for their compact proofs and security, as they do not require a trusted setup, making them versatile for a wide range of use cases. However, their high verification costs can hinder performance in scenarios where rapid proof validation is critical [3].

We will explore these trade-offs further through a benchmarking and comparative analysis in the next sections.

TABLE I
COMPARISON OF ZKP PROTOCOLS

Protocol	Setup	Proof Size	Verification Cost
zkSNARKs	Trusted	Compact	Low
zkSTARKs	Transparent	Large	Moderate
Bulletproofs	None	Moderate	High

To evaluate these protocols in the context of privacy-preserving training models, we established benchmarks focusing on proof size, computational cost, verification efficiency, scalability, and security. The methodology is detailed in the next section.

III. BENCHMARKING OF THE THREE ZKPOT PROTOCOLS

This section outlines the methodology adopted to evaluate zkSNARKs, zkSTARKs, and Bulletproofs within the context of privacy-preserving AI training models. The evaluation framework is designed to assess their performance across a range of metrics critical to real-world applications, such as proof size, computational cost, verification efficiency, and scalability.

A. Evaluation Framework

The evaluation framework is structured around five key metrics:

1) **Proof Size**: The size of the proof generated by the protocol, which directly impacts bandwidth usage and storage requirements.
2) **Computational Cost**: The computational resources required for proof generation and verification, measured in terms of time complexity and hardware requirements.
3) **Verification Efficiency**: The time and computational effort needed for the verifier to validate the proof, crucial for resource-constrained systems.
4) **Scalability**: The protocol's ability to handle large datasets or complex computations without significant performance degradation.
5) **Security**: The robustness of the protocol against attacks, including post-quantum security considerations.

These metrics were chosen to provide a comprehensive understanding of each protocol's strengths and limitations, enabling informed decisions for their application in AI systems.

B. Benchmarking Criteria

To ensure consistency and comparability, the following benchmarking criteria were defined:

- **Use Case Scenarios**: The protocols were tested in scenarios representative of real-world applications, such as federated learning and decentralized AI.
- **Complexity Levels**: A range of computational complexities, from simple arithmetic operations to deep neural network training, was considered.
- **Hardware Variations**: Benchmarks were conducted on diverse hardware configurations, including high-performance servers and resource-constrained devices.
- **Data Sensitivity**: Scenarios involving sensitive data were prioritized to evaluate the protocols' privacy-preserving capabilities.

C. Experimental Setup

The experiments were conducted using a combination of open-source libraries and custom implementations to ensure reproducibility and reliability.

1) Implementation Tools: The following tools and frameworks were utilized:

- **Libsnark**: An open-source library specifically designed for zkSNARK implementations, facilitating efficient proof generation and verification [28].
- **Libstark**: A library optimized for zkSTARKs, focusing on scalability and post-quantum security, which is crucial for future-proofing against quantum attacks [26].
- **Bulletproofs Library**: A C++ implementation of Bulletproofs, focused on efficient cryptographic commitments [27].
- **Machine Learning Frameworks**: TensorFlow and PyTorch were used to simulate AI training scenarios and integrate ZKP protocols.

2) Dataset and Model Selection: To emulate realistic training conditions, we selected datasets and models commonly used in AI research:

- **Datasets**: CIFAR-10[1] for image classification and a synthetic dataset for federated learning.

[1]CIFAR-10 is a widely used dataset in the field of machine learning and computer vision, consisting of 60,000 32x32 color images across 10 different classes, making it a benchmark for evaluating image classification algorithms.

- **Models**: A convolutional neural network (CNN) for CIFAR-10 and a logistic regression model for the synthetic dataset.

The diversity of datasets and models ensures a balanced evaluation of each protocol's performance across different AI workloads.

D. Benchmarks and Metrics

The benchmarking process was divided into three phases:
- Proof Generation: Measuring the time, computational resources, and memory usage required to generate proofs for AI training computations.
- Verification: Evaluating the time and computational effort needed for verifying the generated proofs, with emphasis on resource-constrained environments.
- Scalability Tests: Analyzing performance as the size of the dataset and complexity of the model increase.

1) Proof Size Analysis: The size of the proofs generated by each protocol was measured in kilobytes (KB). This metric is critical for applications in bandwidth-limited environments, such as edge computing and IoT devices. Previous studies have shown that zkSNARKs typically produce the smallest proofs, while zkSTARKs generate larger ones due to their transparent setup .

2) Computational Cost: The computational cost was assessed by tracking the CPU cycles and execution time required for proof generation. High-performance servers were used to measure baseline performance, while additional tests on low-power devices evaluated the protocols' feasibility in constrained environments.

3) Verification Efficiency: Verification efficiency was analyzed by measuring the time required to validate proofs under different network and hardware conditions. The results highlight the trade-offs between zkSNARKs' rapid verification and Bulletproofs' higher verification costs .

4) Security Analysis: Security considerations were assessed by reviewing the theoretical guarantees of each protocol. zkSNARKs rely on trusted setups, which pose risks if compromised. zkSTARKs offer robust post-quantum security due to their use of hash-based commitments, while Bulletproofs eliminate trusted setups entirely but are susceptible to higher computational burdens.

E. Challenges and Assumptions

Several challenges were encountered during the evaluation process:

- **Implementation Complexity**: zkSTARKs require significant computational resources, which may not be available in all experimental setups.
- **Hardware Constraints**: The use of resource-limited devices highlighted the importance of efficient protocol implementations.
- **Fair Comparisons**: Ensuring consistent benchmarks across different protocols required standardizing input parameters and experimental conditions.

Assumptions made during the study include the availability of collision-resistant hash functions for zkSTARKs and the integrity of trusted setups for zk-SNARKs. These assumptions align with real-world deployment scenarios and industry practices.

IV. Comparative Analysis

This section provides a detailed evaluation of zkSNARKs, zkSTARKs, and Bulletproofs across key performance metrics. By analyzing their proof size, computational cost, verification efficiency, scalability, and security, we aim to highlight their strengths, limitations, and suitability for privacy-preserving AI training and other applications.

The comparison presented in this study is based on the results reported in [2], [3].

A. Performance Metrics

a) Proof Size: the table 2 (see Appendix) demostrate that the proof sizes vary across different zero-knowledge proof systems, each with its own strengths and trade-offs. zkSNARKs are known for generating the smallest proofs, with sizes that remain constant regardless of the computation's complexity. This is made possible through elliptic curve cryptography, which efficiently encodes computations into compact proofs. Such small proof sizes are highly advantageous in applications like blockchain systems, where minimizing storage and bandwidth usage is crucial. For example, ZCash uses Groth16 zkSNARKs[2] to enable privacy-preserving transactions while keeping storage requirements low.

In contrast, zkSTARKs produce larger proofs that grow logarithmically with the size of the computation. This is because they rely on hash-based cryptographic primitives to maintain proof integrity, which involves more data. Although their proofs are bigger, they eliminate the need for a trusted setup and offer greater security. This makes them particularly useful in environments where trusted setups may be vulnerable. Additionally, zkSTARKs' reliance on hash functions makes them resistant to quantum attacks, which helps justify their larger proof sizes.

Bulletproofs strike a balance, offering moderately compact proofs. These proofs use inner-product arguments to achieve logarithmic compression as computations increase. While Bulletproofs don't match the constant proof sizes of zkSNARKs, they also don't require a trusted setup, making them a good choice for financial applications. For instance, they are widely used in range proofs for cryptocurrencies, where both compactness and transparency are essential. However, the compression techniques used in Bulletproofs introduce some computational overhead, resulting in proofs that are not as small as those of zkSNARKs.

[2]Groth16 refers to a zero-knowledge succinct non-interactive argument of knowledge (zkSNARK). It is named after Jens Groth, who introduced the protocol in 2016 [50].

b) Computational Cost: ZkSNARKs have among the lowest computational costs, particularly for verifying proofs. This efficiency is due to pre-computed public parameters generated during the trusted setup phase, which significantly reduces the workload for proof generation and verification. However, the trusted setup is resource-intensive and must be repeated for each new circuit. Despite this, zk-SNARKs are highly efficient for applications involving repeated computations, such as smart contracts and blockchain systems.

Unlike zkSNARKs, zkSTARKs do not require a trusted setup, which results in more computational work during proof generation. They utilize hash-based commitments that are more demanding but provide transparency and robust security, including resistance to quantum attacks. This makes them suitable for large-scale computations, such as federated learning, where the additional computational cost is justified by the enhanced security. Furthermore, zkSTARKs scale efficiently with larger datasets, making them ideal for data-heavy applications.

Bulletproofs exhibit the highest computational costs among the three systems. Their proof generation involves iterative compression using inner-product arguments, which is resource-intensive. While this makes Bulletproofs more computationally expensive, they do not require a trusted setup, which is advantageous for decentralized systems that prioritize transparency. However, their high computational demands limit their practicality for handling very large datasets or complex models [3].

c) Verification Efficiency: zkSNARKs are known for their exceptional verification efficiency, offering constant-time validation regardless of the computation's size. This is made possible by using elliptic curve pairings, which allow verifiers to check proofs with very little computational effort. This efficiency is crucial for decentralized applications, where fast verification ensures the system remains responsive and scalable. For instance, blockchain systems use zkSNARKs to quickly validate transactions without compromising sensitive data.

zkSTARKs, on the other hand, have verification times that scale logarithmically with the size of the computation. While they might be slower than zkSNARKs for smaller tasks, their efficiency becomes more noticeable in larger-scale scenarios. By using hash-based commitments, zkSTARKs provide secure and scalable verification, even for large datasets. This makes them a great fit for decentralized machine learning systems, where scalability is often more important than speed.

Bulletproofs, however, have slower verification times compared to zkSNARKs and zkSTARKs. This is due to the need to validate elliptic curve operations and the compression steps involved in inner-product arguments. While Bulletproofs focus on compactness and transparency, their slower verification makes them less suitable for applications that require quick validation. Still, they are a solid choice for financial systems where having small, transparent proofs is more important than verification speed.

d) Scalability of zkPoT Protocols: zkSTARKs are the most scalable of the three protocols, making them ideal for handling large datasets and complex computations. Their scalability is attributed to quasi-linear prover complexity and logarithmic verifier complexity, allowing efficient processing of extensive data. This makes zkSTARKs particularly suitable for federated learning systems, where decentralized nodes collaborate to train models without sharing raw data. Additionally, their use of hash-based commitments ensures effective scaling while maintaining strong security. [13]

zkSNARKs are better suited for small to medium-scale computations. They face scalability challenges due to the requirement of a circuit-specific trusted setup for each new computation, which limits their flexibility for changing workloads. However, their constant proof size and verification time make them a good option for repetitive tasks in constrained environments, such as validating blockchain transactions. Recent innovations like PLONK [53] have introduced universal setups to mitigate some limitations, but zkSNARKs still lag behind zkSTARKs in scalability. [13]

Bulletproofs offer moderate scalability, with prover complexity growing logarithmically with computation size, making them more flexible than zkSNARKs for varied workloads. However, the high computational cost during proof generation can limit their effectiveness for very large computations. Consequently, Bulletproofs are most effective for applications with moderate data needs, such as privacy-focused financial transactions.

e) Security of zkPoT Protocols: zkSTARKs offer the highest level of security among the three protocols. They utilize hash-based commitments, making them resistant to attacks from both classical and quantum adversaries. This post-quantum security is crucial for scenarios requiring long-term cryptographic integrity. Additionally, they do not require a trusted setup, enhancing transparency and eliminating vulnerabilities associated with a compromised setup process. [3], [29]

zkSNARKs, while efficient and widely adopted, depend on elliptic curve cryptography for their security. This reliance makes them vulnerable to quantum attacks, as quantum computers could potentially solve the discrete logarithm problems they are based on. Furthermore, they require a trusted setup, which poses a security risk if the setup process is not entirely honest. Despite these limitations, they remain highly secure against classical attacks and are popular in applications where quantum resistance is not yet a pressing concern. [3], [29]

Bulletproofs, similar to zkSNARKs, rely on elliptic curve cryptography (see section II-D) and are therefore not resistant to quantum attacks. However, they eliminate the need for a trusted setup, reducing risks associated with centralized trusted parties and improving transparency. While Bulletproofs are not post-quantum secure, they provide strong protection against classical adversaries, making them a reliable option for current financial applications.

B. Application Suitability of zkPoT Protocols

ZkSNARKs are well-suited for Proof-of-Training (PoT) applications requiring compact proofs and fast verification. Their small proof size and rapid verification make them ideal for decentralized machine learning frameworks, where verifying model updates efficiently is crucial. However, their reliance on a trusted setup introduces limitations in scenarios requiring frequent retraining or adaptability.

On the other hand, zkSTARKs, with their scalability and transparency, excel in large-scale training environments, such as federated learning, privacy-preserving AI, and scientific simulations. Their robustness against quantum threats and ability to handle complex computations make them a strong choice for adversarial settings where secure, verifiable training is essential.

Meanwhile, Bulletproofs offer a balanced approach, particularly in privacy-focused PoT applications. They are highly effective for verifying model integrity and training fairness in financial and confidential AI systems. By eliminating the need for a trusted setup while maintaining efficiency, Bulletproofs enhance trust-minimization in decentralized machine learning markets and privacy-preserving AI training.

C. Challenges and Trade-offs of zkPoT Protocols

zkSNARKs prioritize efficiency and compact proof sizes, making them ideal for Proof-of-Training (PoT) in environments with limited bandwidth or computational resources. Their ability to generate small proofs and enable fast verification makes them particularly useful for verifying model updates in decentralized machine learning. However, their requirement for a trusted setup can introduce vulnerabilities in adversarial training scenarios, limiting flexibility in dynamic AI applications that demand frequent retraining or adjustments.

On the other hand, **zkSTARKs** offer exceptional scalability and strong security, including resistance to quantum attacks, making them well-suited for large-scale PoT applications. Their ability to handle complex computations makes them ideal for privacy-preserving AI, federated learning, and scientific simulations. Despite this, they come with higher computational costs and produce larger proofs, which can be a limitation for machine learning systems requiring minimal resource usage. Nonetheless, their security and adaptability make them a powerful choice for verifiable AI training in adversarial environments.

Bulletproofs strike a balance between proof size and transparency by eliminating the need for a trusted setup, enhancing trust in decentralized PoT systems. They are particularly effective for verifying model integrity and ensuring fair AI training in privacy-sensitive applications. However, due to higher prover costs and slower verification times, they may not be suitable for real-time or high-frequency model updates. Their strengths lie in ensuring fairness and confidentiality in financial AI models and privacy-preserving ML training.

The table II summarizes the comparative analysis and provides a clear comparison:

zkSNARK	**zkSTARK**	**Bulletproofs**
Requires trusted setup	No trusted setup	No trusted setup
Limited scalability	Highly scalable	Moderate scalability
ECC-based	Hash-based	Discrete log-based
200-300 bytes	Kilobytes	Kilobytes
Fast verification	Fast verification	Slower verification
Expensive (ECC)	Efficient (FFT)	Medium complexity
Not post-quantum	Post-quantum secure	Not post-quantum
Good for small ML	Best for large ML	only range proofs
Non-interactive	Non-interactive	Non-interactive
Federated learning	Large models	Range proofs in ML

TABLE II
SIMPLIFIED COMPARISON OF ZKSNARKS, ZKSTARKS, AND BULLETPROOFS IN PROOF OF TRAINING

In summary, understanding these trade-offs is crucial when selecting the appropriate zkPoT protocol for a given application, whether it involves scaling AI training models, securing federated learning environments, or maintaining efficiency in decentralized machine learning frameworks.

In the next section, we will explore optimizations and promising future ideas aimed at mitigating one of these trade-offs in the protocol to enhance scalability, security, and succinctness.

V. OPTIMIZATION STUDIES AND FUTURE DIRECTIONS

The evolution of zero-knowledge proof protocols continues to play a pivotal role in advancing cryptographic research and real-world applications, particularly in Proof-of-Training systems. This section explores potential enhancements to zkSNARKs, zkSTARKs, and Bulletproofs, focusing on protocol optimizations, emerging applications in decentralized AI training, and innovative designs that address current limitations while expanding their usability in verifiable machine learning.

A. Protocol Optimizations

*a) **zkSNARKs + MPC in the head: Succinct proof size and small computation time**:* The study in [5] presents a new approach to constructing Zero-Knowledge Proofs of Training (zkPoT) for machine learning models, with a specific focus on logistic regression. It addresses the inefficiencies of classical methods such as zkSNARKs, which generate succinct proofs but are computationally intensive, and MPC-in-the-head techniques, which are faster but produce large proofs shown by [30]–[33]. By combining these two approaches, the authors achieve an optimal balance between proof size and computation time. The protocol is designed to be streaming-friendly, eliminating the need for massive RAM usage and allowing it to handle large datasets by loading computation incrementally from secondary storage. Divided into three phases—offline, data checks, and online—the

method enhances modularity and efficiency, enabling the reuse of intermediate proofs. The approach further innovates by employing packed secret sharing in the MPC protocol to reduce communication overhead and leveraging fixed-point arithmetic to optimize real-number operations in finite fields. Benchmarking on a dataset of over 262,000 records with 1024 features, the authors demonstrated that the prover could complete the online phase in under 10 minutes, while the verifier required less than 30 seconds, with a total proof size of less than 10% of the dataset size. This innovation not only makes zkPoTs feasible for large-scale ML tasks but also sets a foundation for extending this approach to more complex models, showcasing a significant leap in privacy-preserving machine learning.

b) ***Kaizen: scalable zkSNARKs for deep neural networks****:* The study in [6] about Zero-Knowledge Proofs of Training for Deep Neural Networks, introduces Kaizen, a state-of-the-art protocol designed to efficiently and succinctly handle Zero-Knowledge Proofs of Training (zkPoTs) for complex deep neural networks (DNNs). Unlike classical zkSNARKs, which struggle with the iterative nature and high computational demands of DNN training, Kaizen combines optimized proof systems for gradient descent with recursive proof composition to achieve scalability. Its Gradient Descent Proofs (PoGD), shown in 1 , leverage GKR-style (sumcheck-based) techniques [34]–[36] to efficiently handle matrix operations like multiplications and convolutions, while non-linear operations such as ReLU and Softmax are approximated for efficiency. The recursive composition ensures that proof size and verification runtime remain constant, regardless of the number of training iterations or dataset size, addressing a critical limitation in previous zkPoT designs. Benchmarked on the VGG-11 model with 10 million parameters, Kaizen achieves 24× faster proof generation and uses 27× less prover memory compared to traditional methods, with a fixed proof size of just 1.63 MB and verification time of 130 milliseconds. By introducing an aggregation scheme for multivariate polynomial commitments, Kaizen further reduces the cost of recursive composition, making it a practical solution for real-world DNN training. This innovation represents a major step forward in privacy-preserving machine learning, enabling zkPoTs for DNNs at scale while maintaining strong cryptographic guarantees.

c) ***Aurora: zkSNARk with transparent setup****:* Aurora a zero-knowledge succinct non-interactive argument of knowledge (zkSNARK) intreduced in [38] specifically designed for Rank-1 Constraint Satisfaction (R1CS). Aurora achieves very good efficiency through a transparent setup, plausible post-quantum security, and black-box use of lightweight symmetric cryptography. Unlike classical zkSNARKs, Aurora uses a new Interactive Oracle Proof (IOP) protocol [39], [40], which introduces a univariate analogue of the sumcheck problem [41], significantly reducing proof size and computational overhead. This innovative protocol enables proofs with polylogarithmic size in relation to circuit size, resulting in proofs that are 20–30× smaller than prior zkSNARKs like Ligero [42] or Stark [43], particularly for circuits with millions of gates. Additionally, Aurora's verifier runtime is logarithmic in the circuit size, with proofs requiring less than 130 kB even for large instances. The system is further complemented by an open-source library, libiop, which facilitates the creation of IOP-based arguments and enhances accessibility for practical implementations. Aurora's design represents a major leap forward in efficiency and scalability, making zero-knowledge proofs more practical for applications like blockchain systems while ensuring robust security in a post-quantum world.

d) ***zkVMs: engineering- and resource-friendly zkPoT implementation****:* There is an idea still in development that has shown a promising improvement in zkPoT. It is the Zero-Knowledge Virtual Machines (zkVMs) presented in study [37] by Tim Dokchitser and Alexandr Bulkin. It introduces innovative techniques to achieve configurability, scalability, and efficiency in ZKVM design while preserving privacy.This allows users to create custom ZKVMs by specifying high-level command sets, enabling auto-generation of Prover and Verifier components along with LLVM-compatible backends. The proposed framework significantly reduces the fragility of ZK schemes by enabling rapid updates to Prover/Verifier pairs in response to evolving security needs or innovations in zero-knowledge proof technology. The study outlines contributions such as a ZKfriendly memory model, which replaces registers with memory cells for improved efficiency, and a polynomial-based constraint system for validating execution traces. The use of techniques like FRI low-degree testing and the Fiat-Shamir heuristic ensures that proofs remain efficient, post-quantum secure, and non-interactive.Additionally, it introduces primitives like permutation and lookup to streamline ZKVM operations, supporting complex instructions like conditional jumps, arithmetic, and logical operations. The configurability of this framework allows ZKVMs to balance efficiency and flexibility. By making ZK technology more accessible and adaptable, this work represents a major step forward in general-purpose ZK proof systems.

B. ***Applications in Advanced Domains***

a) Privacy-Preserving AI Training Models: As privacy concerns grow in AI training, integrating ZKP protocols into federated learning and other collaborative training systems becomes increasingly relevant. zkSTARKs and Bulletproofs, with their scalability and transparency, are particularly well-suited for validating model updates without exposing sensitive training data. Additionally, zkSNARKs could be used in smaller-scale AI applications where compact proofs and rapid verification are essential. These advancements would facilitate secure and efficient training across decentralized nodes, promoting wider adoption of privacy-preserving AI. [5]

b) Quantum-Secure Cryptographic Systems: The rise of quantum computing necessitates cryptographic systems that

remain secure against quantum attacks. zkSTARKs, with their hash-based cryptographic foundations, provide a robust framework for quantum-secure applications. Future research could explore extending zkSTARK designs to address emerging quantum-resistant requirements, ensuring long-term viability in post-quantum environments. These developments are critical for safeguarding blockchain networks, financial systems, and secure communications.

c) Decentralized Identity Management: ZKP protocols have significant potential in decentralized identity systems, where privacy and data integrity are paramount. Bulletproofs, with their moderate proof sizes and trusted setup independence, could enable efficient and transparent identity validation. zkSNARKs and zkSTARKs, with their scalability and rapid verification, are also strong candidates for large-scale decentralized identity frameworks. Future developments could focus on optimizing these protocols for seamless integration into real-world identity management systems.

C. *Protocol Design Innovations*

a) Leveraging zkSTARKs for Real-Time AI Model Validation: zk-STARKs' scalability and logarithmic verification times make them ideal for real-time AI model validation in decentralized environments. Future designs could optimize zkSTARKs to support continuous validation of training progress without requiring centralized control. By enabling verifiable updates at every training step, zkSTARKs would ensure integrity and accuracy in collaborative AI systems. This approach would also enhance trust in federated learning setups, where the lack of transparency is often a challenge.

b) Adaptive zkSNARK Systems for Evolving Computations: While zkSNARKs are efficient, their reliance on circuit-specific setups limits their adaptability to evolving computations. Future work could explore adaptive zkSNARK designs that use universal setups capable of accommodating a wide range of computational tasks. Such systems would eliminate the need for repeated setups, reducing operational overhead and broadening zkSNARKs' applicability. Techniques like dynamic circuit generation and modular proof structures could pave the way for more versatile zkSNARK implementations.

c) Efficient Aggregation of Bulletproofs: Aggregating multiple Bulletproofs into a single proof would significantly reduce verification overhead in scenarios involving batch processing. For example, a financial application handling multiple range proofs simultaneously could benefit from this optimization. Research into efficient aggregation techniques, such as multi-prover frameworks or commitment-based compression, could enhance Bulletproofs' scalability and efficiency in high-throughput systems.

D. *Summary of Future Potential*

Optimizing zkSNARKs, zkSTARKs, and Bulletproofs for real-time, scalable, and secure operations will unlock new possibilities across various domains. By addressing current limitations, such as trusted setups, proof sizes, and computational costs, ZKP protocols can achieve broader adoption in applications ranging from AI training to decentralized identity systems and cross-chain blockchain networks. Continued research and innovation will play a crucial role in advancing ZKP technology to meet the evolving demands of privacy-preserving systems.

VI. Conclusion

The comparative analysis of zkSNARKs, zkSTARKs, and Bulletproofs underscores the transformative potential of zero-knowledge proof of training (zkPoT) protocols in cryptographic research and real-world applications. This study evaluated these protocols across key performance metrics, including proof size, computational cost, verification efficiency, scalability, and security, with a focus on their applicability to privacy-preserving AI training models and other decentralized systems. The results reveal distinct strengths, limitations, and trade-offs, highlighting the versatility and evolving nature of ZKP technologies.

A. *Key Findings*

zkSNARKs, particularly Groth16 , demonstrated exceptional efficiency in proof size and verification. Their compact proofs and constant-time verification make them highly suitable for bandwidth-constrained and resource-limited environments, such as blockchain systems. However, their reliance on elliptic curve cryptography (see section II-D) and circuit-specific trusted setups poses challenges in scalability and quantum resistance. Innovations like PLONK and Marlin have mitigated some of these limitations by introducing universal setups and optimizing proof generation for larger computations. These developments position zkSNARKs as a critical component in applications where compactness and efficiency are paramount.

zk-STARKs emerged as the most scalable and secure protocol, excelling in scenarios requiring verifiable computations across large datasets. Their transparent setups and reliance on hash-based cryptography eliminate the vulnerabilities associated with trusted setups, making them resistant to quantum attacks. While zkSTARKs' larger proof sizes and higher computational costs remain areas for improvement, ongoing research into proof size reduction and efficiency optimizations suggests a promising future for zkSTARK adoption in federated learning, scientific simulations, and other large-scale applications.

Bulletproofs provided a balanced approach with moderate proof sizes and trusted setup independence. Their transparency and compactness make them well-suited for privacy-preserving financial systems, such as range proofs in cryptocurrencies. However, Bulletproofs face challenges in computational efficiency, particularly during proof generation and verification. Future advancements in aggregation techniques and compression methods could expand their applicability to more complex systems while preserving their inherent strengths.

The findings of this study have several practical implications for the adoption and development of ZKP protocols:

1.Privacy-Preserving AI Training Models: zkSTARKs and Bulletproofs offer scalable and transparent solutions for federated learning systems, enabling secure model validation without exposing sensitive data. zkSNARKs could complement these protocols in scenarios requiring compact proofs and rapid verification. **2.Blockchain Systems:** zkSNARKs' compactness and efficiency make them ideal for transaction validation in blockchain systems, while zkSTARKs' scalability supports cross-chain interoperability and large-scale decentralized networks. **3.Financial Applications:** Bulletproofs' moderate proof sizes and setup independence align well with privacy-preserving financial systems, enabling secure and efficient range proofs in cryptocurrencies.

B. Challenges and Recommendations

Despite their advantages, each protocol faces challenges that must be addressed to maximize their potential. zkSNARKs require further innovations to eliminate trusted setups and enhance scalability. zkSTARKs must prioritize reducing proof sizes and computational costs to improve usability in bandwidth-limited environments. Bulletproofs need optimizations in prover algorithms and verification efficiency to handle larger computations. Collaborative research efforts focusing on hybrid systems, proof aggregation, and modular designs are recommended to address these challenges and drive the next generation of ZKP protocols.

C. Future Prospects

The future of ZKP protocols lies in their ability to adapt to emerging technological challenges and integrate with advanced systems. With ongoing advancements in cryptographic techniques and increasing demand for privacy-preserving solutions, zkSNARKs, zkSTARKs, and Bulletproofs are expected to play a central role in shaping the next era of secure and scalable applications. By addressing current limitations and exploring innovative designs, these protocols can achieve broader adoption in domains such as AI training, decentralized identity systems, and quantum-resistant cryptographic frameworks.

Closing Remarks: This study highlights the transformative potential of ZKP protocols and the critical role of ongoing research and collaboration in unlocking their full potential. As the field evolves, zkSNARKs, zkSTARKs, and Bulletproofs will continue to redefine the boundaries of cryptographic systems, driving innovation in privacy, security, and scalability.

REFERENCES

[1] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof systems," in *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*, pp. 186–208, ACM, 1985. Available: http://people.csail.mit.edu/silvio/Selectedpers/Proof.

[2] B. Oude Roelink, M. El-Hajj, and D. Sarmah, "Systematic review: Comparing zk-SNARK, zk-STARK, and bulletproof protocols for privacy-preserving authentication," *Security and Privacy*, 2024. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/spy2.401.

[3] "Benchmarking the zk-SNARK, zk-STARK, and Bulletproof Non-Interactive Zero-Knowledge Proof Protocols in an Equivalent Practical Application," unpublished.

[4] M. Blum, P. Feldman, and S. Micali, "Non-interactive zero-knowledge and its applications," in *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, pp. 103–112, ACM, 1988. Available: https://dl.acm.org/doi/10.1145/62212.62222.

[5] S. Garg *et al.*, "Experimenting with Zero-Knowledge Proofs of Training," in *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS '23)*, ACM, 2023. Available: https://doi.org/10.1145/3576915.3623202.

[6] K. Abbaszadeh *et al.*, "Zero-Knowledge Proofs of Training for Deep Neural Networks," in *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security (CCS '24)*, ACM, 2024. Available: https://doi.org/10.1145/3658644.3670316.

[7] A. R. Block *et al.*, "Fiat-Shamir Security of FRI and Related SNARKs," unpublished, 2024.

[8] J. M. Pollard, "Monte Carlo Methods for Index Computation (mod p)," *Mathematics of Computation*, vol. 32, no. 143, pp. 918, 1978. Available: https://www.jstor.org/stable/2006496?origin=crossref.

[9] A. Gabizon, Z. J. Williamson, and O. Ciobotar, "PlonK: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge," unpublished, 2024.

[10] A. Berentsen, J. Lenzi, and R. Nyffenegger, "A walk-through of a simple zk-stark proof," unpublished, 2022. Available: https://ssrn.com/abstract=4308637.

[11] M. Petkus, "Why and How zk-SNARK Works," unpublished, 2019. Available: http://arxiv.org/abs/1906.07221.

[12] M. Petkus, "Why and how zk-snark works," *arXiv preprint*, 2019. Available: http://arxiv.org/abs/1906.07221.

[13] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev, "Scalable Zero Knowledge with no Trusted Setup," unpublished.

[14] E. Ben-Sasson *et al.*, "Short PCPs Verifiable in Polylogarithmic Time," in *Proceedings of the 20th Annual IEEE Conference on Computational Complexity*, pp. 120–134, 2005.

[15] H. T. Larasati and H. Kim, "Quantum Cryptanalysis Landscape of Shor's Algorithm for Elliptic Curve Discrete Logarithm Problem," in *Information Security Applications*, Springer, 2021, pp. 91–104. Available: https://doi.org/10.1007/978-3-030-89432-0_8.

[16] E. Ben-Sasson *et al.*, "Scalable, transparent, and post-quantum secure computational integrity," unpublished, 2018. Available: https://eprint.iacr.org/2018/046.

[17] X. Dong *et al.*, "Quantum Attacks on Hash Constructions with Low Quantum Random Access Memory," in *Advances in Cryptology – ASIACRYPT 2023*, Springer Nature, 2023, pp. 3–33. Available: https://doi.org/10.1007/978-981-99-8727-6_1.

[18] E. Ben-Sasson *et al.*, "libSTARK: a library for zero knowledge (ZK) scalable transparent argument of knowledge (STARK)," unpublished. Available: https://github.com/elibensasson/libSTARK.

[19] L. Lovesh, "bulletproofs-r1cs-gadgets/src/gadgetmimc.rs at master," unpublished, 2019. Available: https://github.com/lovesh/bulletproofs-r1cs-gadgets/blob/master/src/gadget$_{mimc.rs}$.

[20] "bulletproofs::generators::AggregatedGensIter - Rust," unpublished, 2021. Available: https://doc-internal.dalek.rs/bulletproofs/generators/struct.AggregatedGensIter.html.

[21] B. Bünz *et al.*, "Bulletproofs: Short Proofs for Confidential Transactions and More," in *39th IEEE Symposium on Security and Privacy 2018*, 2017. Available: https://eprint.iacr.org/2017/1066.

[22] The Monero Project, "Moneropedia: Bulletproofs," unpublished, 2018. Available: https://www.getmonero.org/resources/moneropedia/bulletproofs.html.

[23] T. P. Pedersen, "Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing," in *Advances in Cryptology — CRYPTO '91*, Springer, 1992, pp. 129–140. Available: https://doi.org/10.1007/3-540-46766-1_9.

[24] D. Hankerson and A. Menezes, "Elliptic Curve Discrete Logarithm Problem," in *Encyclopedia of Cryptography and Security*, Springer, 2011, pp. 397–400. Available: https://doi.org/10.1007/978-1-4419-5906-$5_2$46.

[25] "Increasing transparency in AI security," unpublished, 2023. Available: https://security.googleblog.com/2023/10/increasing-transparency-in-ai-security.html.

[26] E. Ben-Sasson *et al.*, "elibensasson/libSTARK," unpublished, 2024. Available: https://github.com/elibensasson/libSTARK.

[27] "bulletproofs - crates.io: Rust Package Registry," unpublished, 2021. Available: https://crates.io/crates/bulletproofs/4.0.0.

[28] "bellman - crates.io: Rust Package Registry," unpublished, 2023. Available: https://crates.io/crates/bellman/0.14.0.

[29] A. Banerjee, M. Clear, and H. Tewari, "Demystifying the Role of zk-SNARKs in Zcash," in *2020 IEEE Conference on Application, Information and Network Security (AINS)*, 2020, pp. 12–19. Available: https://arxiv.org/abs/2008.00881.

[30] S. Ames *et al.*, "Ligero: Lightweight Sublinear Arguments Without a Trusted Setup," in *ACM CCS 2017*, pp. 2087–2104, 2017. Available: https://doi.org/10.1145/3133956.3134104.

[31] M. Chase *et al.*, "Post-Quantum Zero-Knowledge and Signatures from Symmetric-Key Primitives," in *ACM CCS 2017*, pp. 1825–1842, 2017. Available: https://doi.org/10.1145/3133956.3133997.

[32] I. Giacomelli, J. Madsen, and C. Orlandi, "ZKBoo: Faster Zero-Knowledge for Boolean Circuits," in *USENIX Security 2016*, pp. 1069–1083.

[33] J. Katz, V. Kolesnikov, and X. Wang, "Improved Non-Interactive Zero Knowledge with Applications to Post-Quantum Signatures," in *ACM CCS 2018*, 2018.

[34] T. Xie *et al.*, "Libra: Succinct Zero-Knowledge Proofs with Optimal Prover Computation," in *Advances in Cryptology—Crypto 2019*, pp. 733–764, 2019.

[35] J. Zhang *et al.*, "Doubly Efficient Interactive Proofs for General Arithmetic Circuits with Linear Prover Time," in *Conf. on Computer and Communications Security*, pp. 159–177, 2021.

[36] A. Golovnev *et al.*, "Brakedown: Linear-Time and Post-Quantum SNARKs for R1CS," unpublished, 2021. Available: https://eprint.iacr.org/2021/1043.

[37] T. Dokchitser and A. Bulkin, "ZERO KNOWLEDGE VIRTUAL MACHINE STEP BY STEP," unpublished, 2023. Available: https://eprint.iacr.org/2023/1032.pdf.

[38] E. Ben-Sasson *et al.*, "Aurora: Transparent Succinct Arguments for R1CS," unpublished, 2019.

[39] E. Ben-Sasson, A. Chiesa, and N. Spooner, "Interactive Oracle Proofs," in *Proceedings of the 14th Theory of Cryptography Conference*, pp. 31–60, 2016.

[40] O. Reingold, R. Rothblum, and G. Rothblum, "Constant-Round Interactive Proofs for Delegating Computation," in *Proceedings of the 48th ACM Symposium on the Theory of Computing*, pp. 49–62, 2016.

[41] C. Lund *et al.*, "Algebraic Methods for Interactive Proof Systems," *Journal of the ACM*, vol. 39, no. 4, pp. 859–868, 1992.

[42] S. Ames *et al.*, "Ligero: Lightweight Sublinear Arguments Without a Trusted Setup," in *Proceedings of the 24th ACM Conference on Computer and Communications Security*, pp. 2087–2104, 2017.

[43] E. Ben-Sasson *et al.*, "Scalable, transparent, and post-quantum secure computational integrity," unpublished, 2018. Available: https://eprint.iacr.org/2018/046.

[44] "Intro to zkML," unpublished, 2022. Available: https://worldcoin.org/blog/engineering/intro-to-zkml.

[45] "Zator Project Repository," unpublished, 2022. Available: https://github.com/lyronctk/zator/tree/main.

[46] B. Feng *et al.*, "ZEN: An Optimizing Compiler for Verifiable, Zero-Knowledge Neural Network Inferences," *Cryptology ePrint Archive*, vol. Report 2021/087, 2021. Available: https://eprint.iacr.org/2021/087.

[47] S. Lee *et al.*, "vCNN: Verifiable Convolutional Neural Network," *Cryptology ePrint Archive*, vol. Report 2020/584, 2020. Available: https://eprint.iacr.org/2020/584.

[48] T. Liu, X. Xie, and Y. Zhang, "zkCNN: Zero Knowledge Proofs for Convolutional Neural Network Predictions and Accuracy," in *ACM CCS 2021*, pp. 2968–2985, 2021. Available: https://doi.org/10.1145/3460120.3485379.

[49] Amazon Web Services, Inc., "Quantum Computer and Simulator – Amazon Braket Pricing," 2024. Available: https://aws.amazon.com/braket/pricing/. [Accessed: Jan. 24, 2024].

[50] K. George, "The Mathematical Mechanics Behind the Groth16 Zero-knowledge Proving Protocol," Dec. 6, 2022.

[51] A. Menezes, "An Introduction to Pairing-Based Cryptography,". Available:**??**

[52] A. Kate, G.M. Zaverucha, I. Goldberg, "Polynomial Commitments,"Dec. 1,2010. Available: https://cacr.uwaterloo.ca/techreports/2010/cacr2010-10.pdf

[53] A.Gabizon, Z.J Williamson, O. Ciobotaru, "PlonK: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge, "Feb. 23,2024.

APPENDIX

Rnds	Protocol	CT (ms)	ST (ms)	PT (ms)	VT (ms)	SC (b)	SP (b)
15	**Bulletproof**	-	-	6.756	0.899	-	-
15	**SNARK (R)**	-	10.467	4.479	1.703	-	-
15	**SNARK (G)**	0.043	3.425	1.299	1.138	-	-
15	**STARK**	-	-	2.060	0.052	120	73
63	**Bulletproof**	-	-	25.210	2.677	-	-
63	**SNARK (R)**	-	18.643	5.563	1.686	-	-
63	**SNARK (G)**	0.227	10.292	2.420	1.195	-	-
63	**STARK**	-	-	0.552	0.142	118	75
255	**Bulletproof**	-	-	102.450	11.069	-	-
255	**SNARK (R)**	-	42.788	12.218	1.709	-	-
255	**SNARK (G)**	1.830	40.888	5.676	1.407	-	-
255	**STARK**	-	-	11.339	0.199	116	74
1023	**Bulletproof**	-	-	499.610	92.663	-	-
1023	**SNARK (R)**	-	132.280	30.268	1.684	-	-
1023	**SNARK (G)**	10.453	150.211	19.867	2.280	-	-
1023	**STARK**	-	-	13.094	0.313	114	73
4095	**Bulletproof**	-	-	3614.500	1271.200	-	-
4095	**SNARK (R)**	-	440.560	96.865	1.695	-	-
4095	**SNARK (G)**	42.937	453.436	61.512	5.733	-	-
4095	**STARK**	-	-	44.876	0.452	112	72

Fig. 2. Time and security level results of the protocols benchmark [3]

- **CT (ms)** - Compile Time; The time required to compile the circuit in milliseconds.
- **ST (ms)** - Setup Time; The time required to perform the setup in milliseconds.
- **PT (ms)** - Proof Time; The time required to generate the proof in milliseconds.
- **VT (ms)** - Verification Time; The time required to verify the proof in milliseconds.
- **SC (b)** - Security Conjectured; The conjectured security level in bits.
- **SP (b)** - Security Proven; The proven security level in bits.

Input: Weights $(W_{t-1})_{t=1}^{L}$ and Batch B_{t-1}
Output: Updated weights $(W_t)_{t=1}^{L}$
Forward pass:
Let $U_{i,0}$ be the concatenation of input features included in B_{t-1}
for $t = 1$ *to* L **do**
  if *the t-th layer is dense* **then**
    $T_{i,t} \leftarrow W_{t-1} \cdot U_{i,t-1}$
    $U_{i,t} \leftarrow \text{act}(T_{i,t})$; `// e.g., ReLU, tanh, Softmax`
  else
    $T_{i,t} \leftarrow W_{t-1} * U_{i,t-1}$
    $Q_{i,t} \leftarrow \text{act}(T_{i,t})$
    $U_{i,t} \leftarrow \text{pool}(Q_{i,t})$; `// e.g., MaxPool, AvgPool`
  end
end

Backward pass:
Let $R_{i,L+1} = \frac{\partial \mathcal{L}(U_{i,L}, Y_i)}{\partial U_{i,L}}$, where Y_i includes labels in B_{t-1}
for $t = L$ *to* 1 **do**
  if *the t-th layer is dense* **then**
    $T'_{i,t} \leftarrow \frac{\partial U_{i,t}}{\partial T_{i,t}} \cdot \text{act}'(T_{i,t})$
    $G_{t,i} \leftarrow (R_{i,t+1} \cdot T'_{i,t}) \cdot U_{i,t}^{\top}$
    $R_{i,t-1} \leftarrow (R_{i,t+1} \cdot T'_{i,t}) \cdot W_{t-1}^{\top}$
  else
    $T'_{i,t} \leftarrow \frac{\partial U_{i,t}}{\partial Q_{i,t}} \cdot \text{pool}^{-1}(Q_{i,t})$
    $Q'_{i,t} \leftarrow \frac{\partial Q_{i,t}}{\partial T_{i,t}} \cdot \text{act}'(T_{i,t})$
    $G_{t,i} \leftarrow (R_{i,t+1} * Q'_{i,t}) \cdot U_{i,t-1}^{\top}$
    $R_{i,t-1} \leftarrow \text{pad}^{-1}((R_{i,t+1} * Q'_{i,t}) \cdot W_{t-1}^{\top})$
  end
end

Update weights:
for $t = 1$ *to* L **do**
  $W_t \leftarrow W_{t-1} - \eta \cdot G_{t,i}$; `//` η `is the learning rate`
end

Algorithm 1: Mini-Batch Gradient Descent