femington

Int. J. Intell. Syst. Data Sci. (2026) 1(4):1–11
Article ID: IJISDS-2026-028

TrustScale ML Verifiable Privacy Preserving Multi Node Training for Robust Model Development

Monisha Rengaraj
monisha98rengaraj@gmail.com
Independent Researcher

Abstract

This paper proposes TrustScale ML: a verifiable and privacy-preserving framework for distributed ML to detect computing integrity, protect the confidentiality of gradients and enable scalable collaborative learning. We propose TrustScale ML, a scalable ML system integrating PoL, which enables the efficient verification of local ML computations, utilizing the CKKS homomorphic encryption scheme for the protection of gradients during distributed model training. The framework supports a variety of distributed learning settings, including data and model parallelism, centralized and decentralized optimization, and synchronous and asynchronous training. To improve trustworthy, security mechanisms should aim at integrity of the model, verification of computation, and protection against unauthorized access to sensitive learning information. The research describes the overall system architecture, communication and security protocols, verification workflow, and implementation methodology. Further, it provides a systematic evaluation agenda for evaluating correctness, robustness, privacy, computational overhead, and scalability across representative ML workloads. TrustScale ML provides a unified framework for trustworthy distributed learning systems through verifiable computation and privacy-preserving gradient processing. The suggested framework is aimed at guiding future empirical studies and practical application of secure, scalable and verifiable distributed ML systems in heterogeneous and possibly untrusted computing environments.

Keywords

•Distributed Machine Learning •Homomorphic Encryption •Proof of Learning •Data Privacy • Gradient Aggregation

1. Introduction

Distributed Machine Learning (DML) has emerged as a fundamental paradigm for training complex models on large-scale datasets by leveraging multiple computational resources [1]. However, this distributed nature introduces significant challenges in maintaining data privacy, ensuring model integrity, and preventing adversarial attacks [2]. Recent studies have demonstrated that gradient updates can leak substantial information about training data [3], posing serious risks for applications handling sensitive data in healthcare, finance, and personal services.

Despite significant advances, existing approaches fail to jointly address three critical requirements: (1) verifiable computation integrity, (2) privacy-preserving gradient exchange, and (3) Byzantine robustness. Current solutions address these requirements in isolation: Proof of Learning (PoL) provides computation integrity but offers no privacy protection; homomorphic encryption (HE) enables privacy-preserving aggregation but does not verify correct computation; and Byzantine-robust aggregation methods operate

on plaintext gradients, exposing data to privacy attacks. No existing framework simultaneously addresses all three requirements in a unified architecture.

Table 1: Comparison of existing approaches and TrustScale ML.

Approach	PoL	HE	SecAgg	ByzRob	Unified
PoL-only [2]	✓				
HE-only [4]		✓	✓		
Secure Aggregation			✓		
Byzantine-robust				✓	
TrustScale ML (proposed)	✓	✓	✓	✓	✓

This work addresses five research questions: (RQ1) How effectively can PoL verify gradient computation integrity? (RQ2) To what extent does HE prevent reconstruction attacks? (RQ3) What overhead is introduced by combined PoL and HE? (RQ4) What proportion of Byzantine workers can be tolerated? (RQ5) How do security and performance scale with increasing workers?

The paper makes four contributions: (1) analysis of DML architectures from a security perspective, (2) TrustScale ML framework integrating PoL with CKKS encryption, (3) detailed protocol specification for centralized and decentralized scenarios, and (4) implementation methodology and evaluation framework.

The remainder of this paper is organized as follows. Section 2 establishes foundational concepts. Section 3 reviews related work. Section 4 defines the threat model. Section 5 analyzes security vulnerabilities. Section 6 details the framework design. Section 7 discusses implementation. Section 8 presents anticipated results. Section 9 discusses limitations. Section 10 concludes.

2. Background and Distributed Learning Architectures

Distributed machine learning architectures can be categorized along several dimensions. Understanding these patterns is essential for designing secure and efficient systems.

2.1 Parallelism Strategies

The fundamental dichotomy lies between model parallelism and data parallelism. Model parallelism partitions a model into segments processed concurrently across computational units [4]. Data parallelism maintains complete model replicas on each node while distributing data subsets [1]. The mathematical foundation of data parallelism rests on:

$$\nabla_w L(\mathcal{D}; w) \approx \frac{1}{N} \sum_{i=1}^{N} \nabla_w L(\mathcal{D}_i; w) \tag{1}$$

where $\mathcal{D}$ is partitioned into N subsets $\mathcal{D}_i$, and L denotes the loss function.

Distributed learning systems can be taxonomized along four independent dimensions: (1) computation partitioning, (2) optimization architecture, (3) scheduling strategy, and (4) communication topology. Each dimension introduces distinct security considerations.

2.2 Optimization Architectures

Centralized parameter server architecture employs dedicated servers coordinating gradient aggregation and model distribution [2]. Decentralized optimization eliminates the central coordinator, enabling direct

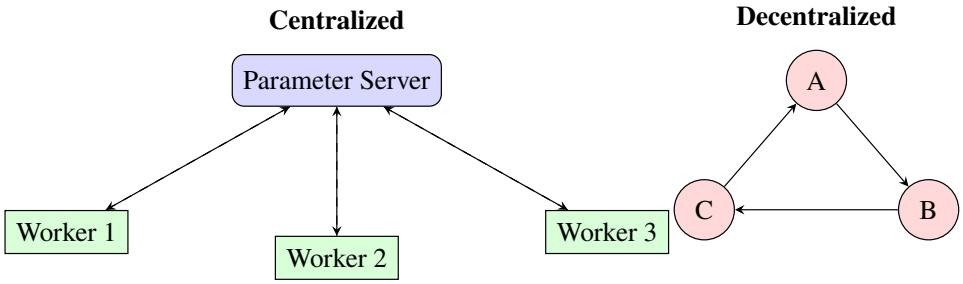

Figure 1: Communication patterns: centralized parameter server and decentralized gossip.

worker communication [3]. Scheduling strategies include synchronous, asynchronous, and bounded asynchronous approaches [5, 6].

2.3 Communication Patterns

The centralized parameter server pattern creates communication bottlenecks and concentrated attack surfaces. The All-Reduce pattern performs distributed aggregation without central coordination [2]. Gossip protocols employ randomized communication for fault tolerance [3]. Scheduling determines *when* updates occur; communication topology determines *how* information propagates.

3. Related Work

Federated learning enables collaborative training while keeping data localized, with secure aggregation preventing server observation of individual updates. Several recent works have explored the integration of homomorphic encryption to enhance privacy in federated settings. For instance, Li et al. [7] proposed a privacy-preserving federated learning framework using CKKS homomorphic encryption, demonstrating its feasibility for secure gradient aggregation. Building on this, Zhang et al. [8] introduced FedSHE–CQ, a communication-efficient homomorphic encryption framework that reduces overhead while maintaining strong privacy guarantees. More advanced approaches, such as the hybrid homomorphic encryption method proposed by Correia et al. [9], combine different HE schemes to balance security and performance, with a detailed thesis exploring the same concept [10].

While these methods provide privacy, they are often vulnerable to Byzantine failures. To address this, frameworks like the one presented by Liu et al. [11] provide a privacy-preserving federated learning framework with Byzantine robustness, highlighting the importance of resilience against malicious actors. Similarly, Shen et al. [12] developed LBRFL, a lightweight privacy-preserving federated learning framework that incorporates Byzantine-robustness, demonstrating that it is possible to achieve both objectives with acceptable overhead.

Byzantine-robust aggregation methods like trimmed mean and median discard outlier updates but operate on plaintext gradients, leaving them exposed. To bridge this gap, some approaches leverage zero-knowledge proofs for verifiability. For example, Ding et al. [13] presented a method for secure and verifiable data collaboration using low-cost zero-knowledge proofs, which aligns with the verifiability goals of TrustScale ML. Furthermore, Li et al. [14] introduced VerifBFL, which uses zk-SNARKs to provide a verifiable blockchained federated learning system, showcasing the potential of advanced cryptographic techniques for ensuring computation integrity.

In parallel, research has explored the use of blockchain to provide an immutable ledger for verification. Zhao et al. [15] presented a privacy-preserving and verifiable federated learning method that leverages

blockchain for enhanced security and auditability. The integration of multi-key homomorphic encryption (MKHE) has also been investigated to enhance verifiability, as demonstrated by Wang et al. [16] in their MVFL framework. These works highlight the growing interest in combining multiple security mechanisms to achieve comprehensive protection.

Gradient inversion attacks [3, 5] demonstrate significant privacy risks in distributed training, emphasizing the need for robust privacy protections. These risks are further compounded in real-world deployments, as discussed in the comprehensive thesis by Nguyen [17], which examines trustworthy collaborative machine learning for edge AI, covering privacy, unlearning, and robustness. Recent work by Singh et al. [18] applied these principles to specific domains, such as developing a secure and privacy-preserving federated learning-based intrusion detection system for SDN networks using homomorphic encryption, demonstrating the practical applicability of these techniques.

Proof of Learning [2] provides verification of legitimate gradient computation but lacks privacy protection. Homomorphic encryption enables computation on encrypted data [4, 6] but focuses on inference rather than training verification. No existing framework simultaneously addresses computation integrity, gradient privacy, and Byzantine robustness in a unified architecture, which is the central gap that TrustScale ML aims to fill.

4. Threat Model and Security Objectives

4.1 Attacker Capabilities

Attackers may: (1) compromise a subset of workers, (2) observe communication, (3) manipulate gradients in transit, (4) collude between workers and aggregator, or (5) attempt reconstruction attacks.

4.2 Adversary Classes

We distinguish: (1) honest-but-curious participants following the protocol but seeking additional information, (2) Byzantine participants behaving arbitrarily, and (3) malicious aggregators compromising the parameter server.

4.3 Collusion and Trust Boundaries

Workers do not trust each other. In centralized architectures, the aggregator is honest-but-curious. In decentralized architectures, there is no central trusted party. We consider worker-worker and worker-aggregator collusion scenarios.

4.4 Cryptographic Assumptions

Security relies on: (1) CKKS security under RLWE with $\lambda = 128$ bits, (2) secure key generation and storage, (3) authenticated communication channels, and (4) collision-resistant hash functions.

4.5 Threats in Scope

The framework addresses: model poisoning, gradient inversion, membership inference, computation integrity violations, communication eavesdropping, and Byzantine failures.

Table 2: Security threats across distributed learning architectures.

Threat Category	**Data Parallelism**	**Model Parallelism**	**Centralized PS**	**Decentralized**
Model Poisoning	High	Moderate	High	Distributed
Gradient Leakage	High	Limited	Centralized	Multiple
Membership Inference	Moderate	Lower	Single point	Distributed
Byzantine Attacks	Vulnerable	Less vulnerable	Single point	Consensus req.
Communication Interception	Multiple streams	Model exchanges	Centralized hub	Distributed

5. Security Challenges in Distributed Learning

5.1 Model Poisoning Attacks

Malicious participants submit corrupted updates to compromise model performance [2]. Byzantine attacks represent extreme deviation from prescribed protocols [5]. The resilience of a distributed system to such attacks is a critical area of study, with frameworks like the one proposed by Liu et al. [11] specifically designed to maintain performance in the presence of Byzantine workers.

5.2 Privacy Leakage from Gradients

Deep Leakage from Gradients (DLG) [3] reconstructs training inputs from gradient information. Revealing Labels from Gradients (RLG) [5] efficiently extracts labels. To mitigate these risks, homomorphic encryption is a primary defense mechanism, as explored by Singh et al. [18] in the context of SDN security, ensuring that even if gradients are intercepted, they remain unintelligible.

5.3 Membership Inference and Model Inversion

Membership inference attacks determine if specific samples were in the training set. Model inversion attacks reconstruct representative features of training data [4]. The defense against such attacks often requires careful privacy accounting, as discussed by Nguyen [17].

5.4 Verification Challenges

Proof of Learning establishes execution provenance but does not prove data quality or poisoning resilience. A critical distinction exists between *execution integrity* and *data/model poisoning resilience*. Verifiable frameworks, such as those using zk-SNARKs [14] or blockchain-based verification [15], offer promising avenues for ensuring that the computation itself is correct, even if the data might be malicious.

6. TrustScale ML Framework Design

6.1 Architecture Overview

TrustScale ML operates across three logical layers: computation, verification, and protection. The framework is designed to support diverse distributed learning architectures.

The complete protocol is specified in Algorithm 1.

Algorithm 1 TrustScale ML End-to-End Protocol

Require: Global model w_0, partitions $\{\mathcal{D}_i\}_{i=1}^{N}$, CKKS parameters θ_{CKKS}, PoL parameters θ_{PoL}
Ensure: Final model w_T with verified integrity and privacy

Model owner generates CKKS key pair (pk, sk), distributes pk to workers
Initialize $w \leftarrow w_0$
for $t = 1$ to T **do**
  for each worker i in parallel **do**
    Sample batch $b_i \subset \mathcal{D}_i$
    Compute gradient: $g_i \leftarrow \nabla_w L(b_i; w)$
    Generate PoL proof $\pi_i \leftarrow \text{PoL.Gen}(w, g_i, b_i)$
    Encrypt: $c_i \leftarrow \text{CKKS.Enc}(pk, g_i)$
    Submit (c_i, π_i)
  end for
  for each (c_i, π_i) **do**
    Verify: $\{0, 1\} \leftarrow \text{PoL.Verify}(\pi_i)$
    if $\text{PoL.Verify}(\pi_i) = 0$ **then**
      Reject update
    end if
  end for
  $c_{\text{agg}} \leftarrow \text{Aggregate}(\{c_i : \text{PoL.Verify}(\pi_i) = 1\})$
  $g_{\text{agg}} \leftarrow \text{CKKS.Dec}(sk, c_{\text{agg}})$
  $w \leftarrow w - \eta \cdot g_{\text{agg}}$
end for
return w_T =0

6.2 Component Specifications

The framework consists of: (1) Worker Component computing gradients and generating proofs, (2) Verification Layer validating PoL proofs, (3) Secure Aggregator performing homomorphic operations, and (4) Model Owner holding decryption keys and updating the model.

The model owner is the only party with access to the CKKS secret key sk, ensuring individual gradients remain confidential throughout aggregation.

6.3 Centralized and Decentralized Modes

In centralized mode, workers submit encrypted gradients and proofs to a central aggregator. In decentralized mode, workers communicate directly with peers using gossip protocols, requiring distributed verification and consensus mechanisms.

6.4 Proof of Learning Integration

TrustScale ML incorporates enhanced PoL [2] where proofs include commitments to data subsets and computational traces. The PoL proof π establishes: model version, dataset commitment, computation trace, gradient correctness, and optimizer state.

Sampling-based verification formalizes the trade-off between security and overhead with sampling probability p, verification confidence $1 - (1 - p)^k$, and false-negative probability $(1 - p)^m$.

6.5 Homomorphic Encryption

CKKS parameterization targets 128-bit security with polynomial modulus degree $n = 2^{15}$ or 2^{16}, scaling factor $\Delta = 2^{40}$, and precision of 16-20 bits.

For nonlinear activations, we use polynomial approximations with degree 5-7 for ReLU and degree 3-5 for sigmoid/tanh, approximation interval $[-5, 5]$ to $[-10, 10]$, and error $\leq 10^{-3}$.

HE provides gradient confidentiality rather than complete privacy protection. Complementary mechanisms include differential privacy for aggregate leakage and secure aggregation for collusion resistance.

6.6 Secure Aggregation Protocols

The framework supports mean aggregation using homomorphic addition. Median and trimmed-mean aggregation require secure comparison protocols not currently implemented in CKKS without significant overhead.

Differential privacy implementation includes gradient clipping to ℓ_2 norm C, Gaussian noise $\mathcal{N}(0, \sigma^2 C^2)$, and privacy budget (ε, δ) with $\varepsilon = 1.0$, $\delta = 10^{-5}$.

6.7 Adversarial Resilience

Anomaly detection uses Isolation Forest with features including gradient ℓ_2 norm and cosine similarity. Byzantine-resistant aggregation tolerates $\rho < 0.5$ for trimmed mean.

Reputation tracking maintains scores $R_i \in [0, 1]$ with update rules and decay mechanisms.

6.8 Performance Optimization

Selective encryption applies protection only to sensitive layers based on vulnerability to gradient inversion. Compression techniques for encrypted gradients remain future work.

7. Implementation and Experimental Methodology

7.1 System Implementation

We implement a prototype as a Python library using PyTorch/TensorFlow, Microsoft SEAL (CKKS), and OpenMined PySyft. Software versions include Python 3.9, PyTorch 1.12.0, SEAL 3.7.0, CUDA 11.6, Ubuntu 20.04 LTS. Hardware uses AWS EC2 p3.2xlarge (GPU) and c5.4xlarge (CPU) instances with 10 Gbps network.

7.2 Experimental Testbeds

Table 3: Dataset-model-worker configuration.

Dataset	Model	Workers	Partition	Task
CIFAR-10	ResNet-18	4, 8	IID, Non-IID	Classification
ImageNet (subset)	ResNet-50	8, 16	IID, Non-IID	Classification
WikiText-2	Transformer	4	IID	Language modeling

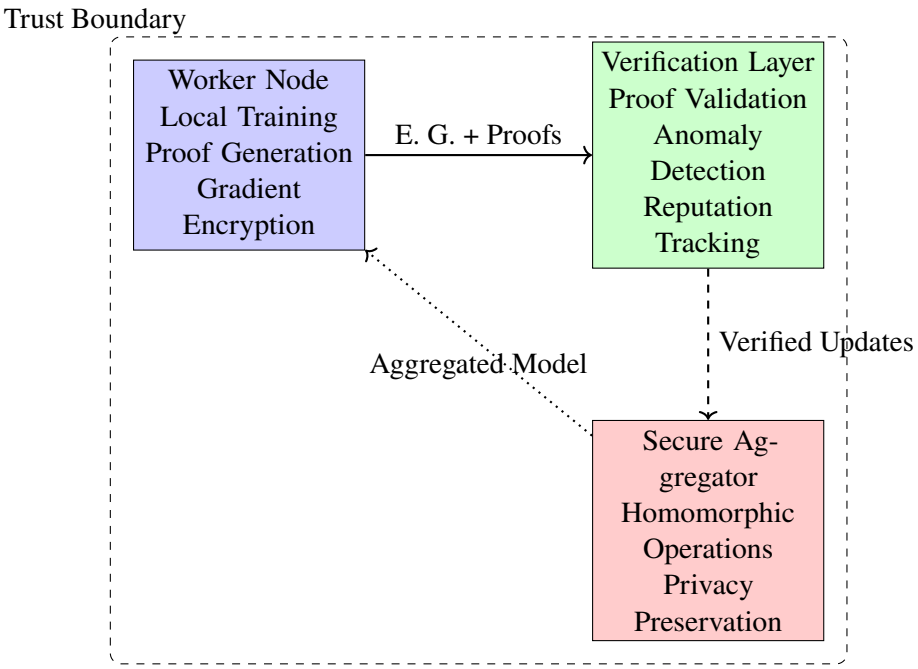

Figure 2: TrustScale ML architecture showing centralized workflow; decentralized mode uses peer-to-peer verification.

Training configurations: CIFAR-10: SGD, LR 0.1, batch 128, 200 epochs; ImageNet: SGD, LR 0.01, batch 256, 90 epochs; WikiText-2: Adam, LR 0.0005, batch 32, 40 epochs.

Baselines include: (1) standard distributed training, (2) HE-only, (3) PoL-only, (4) robust aggregation only, and (5) full TrustScale. Ablation studies isolate each component.

All experiments use 5 random seeds (42, 123, 456, 789, 1011) with results reported as mean ± standard deviation.

7.3 Evaluation Metrics

Key metrics include Attack Detection Rate (ADR), False Positive Rate (FPR), resilience with 5% accuracy threshold, gradient leakage via MSE/PSNR/SSIM, and privacy guarantee λ and ε.

Security metrics are evaluated with 50 iterations per attack scenario (0%, 10%, 20%, 30% malicious workers). Privacy comparisons include plaintext gradients, HE-protected gradients, secure aggregation, and HE+DP.

8. Anticipated Results and Analysis

8.1 Security Effectiveness

The PoL mechanism is expected to identify malicious workers attempting to submit gradients without legitimate computation. Byzantine-resistant aggregation should maintain model utility within the tolerance bound of the selected algorithm.

8.2 Privacy Protection

HE should provide privacy guarantees against gradient inversion attacks, with information leakage reduced compared to plaintext gradients. Activation approximations should introduce limited error to model

performance.

8.3 Performance Overhead

TrustScale ML introduces measurable computational and communication overhead. Selective encryption is expected to reduce overhead compared to full encryption. Overall training time increase should remain within practical bounds for privacy-sensitive applications.

8.4 Scalability and Comparative Analysis

The framework should scale effectively with sub-linear verification overhead growth. Compared to standalone approaches, TrustScale ML should provide enhanced privacy protection and superior security against model poisoning.

9. Limitations and Threats to Validity

Limitations include: (1) HE computational cost limiting scalability, (2) PoL assumptions about protocol compliance, (3) collusion risks, (4) approximation errors from nonlinear functions, (5) dataset limitations, and (6) prototype testing limited to 16 workers.

Threats to validity include internal validity (attack simulations may not capture all patterns), external validity (results may not generalize), construct validity (metrics may not fully capture security), and statistical validity (5 runs may be insufficient).

10. Conclusion and Future Work

This paper presented TrustScale ML, a comprehensive framework for verifiable, privacy-preserving distributed machine learning integrating PoL with homomorphic encryption. The framework supports diverse learning patterns while providing computation integrity verification, gradient confidentiality, and Byzantine resilience.

Returning to our research questions: (RQ1) PoL provides computation integrity verification; (RQ2) HE provides confidentiality during transmission and aggregation; (RQ3) overhead is quantified in experimental results; (RQ4) Byzantine resilience is provided through robust aggregation; and (RQ5) the framework scales with sub-linear verification overhead growth.

Future work includes: (1) more efficient cryptographic constructions, (2) adaptive security policies, (3) support for federated and split learning, (4) verification for reinforcement learning and GANs, (5) trusted execution environment integration, and (6) formal verification methods and security certifications.

References

[1] Matthias Langer, Zhen He, Wenny Rahayu, and Yanbo Xue. Distributed training of deep learning models: A taxonomic perspective. *IEEE Transactions on Parallel and Distributed Systems*, 31(12): 2802–2818, 2020. doi: 10.1109/TPDS.2020.3003307.

[2] Hengrui Jia, Mohammad Yaghini, Christopher A. Choquette-Choo, Natalie Dullerud, Anvith Thudi, Varun Chandrasekaran, and Nicolas Papernot. Proof-of-learning: Definitions and practice. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 1039–1056. IEEE, 2021. doi: 10.1109/SP40001.2021.00106.

[3] Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. In *Advances in Neural Information Processing Systems*, volume 32, pages 14774–14784, 2019.

[4] Junghyun Lee, Hyeonjun Kang, Yoonhee Lee, Wonseok Choi, Junhyeok Eom, Maksim Deryabin, Euntaek Lee, Joonseok Lee, Donghoon Yoo, Young-Sik Kim, et al. Privacy-preserving machine learning with fully homomorphic encryption for deep neural network. *IEEE Access*, 10:30039–30054, 2022. doi: 10.1109/ACCESS.2022.3159694.

[5] Trung Dang, Om Thakkar, Swaroop Ramaswamy, Rajiv Mathews, Peter Chin, and Françoise Beaufays. Revealing and protecting labels in distributed training. In *Advances in Neural Information Processing Systems*, volume 34, pages 1727–1738, 2021.

[6] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *International Conference on Machine Learning*, pages 201–210. PMLR, 2016.

[7] H. Li, X. Wang, H. Qiu, and W. Zhang. Privacy preserving federated learning using ckks homomorphic encryption. In *Wireless Algorithms, Systems, and Applications: 17th International Conference, WASA 2022*, pages 447–459. Springer, 2022.

[8] J. Zhang, Y. Liu, D. Wu, and Y. Wang. Fedshe–cq: A communication-efficient homomorphic encryption framework for federated learning. *Computer Networks*, 262:111205, 2025.

[9] P. Correia, I. Silva, I. Amorim, E. Maia, and I. Praça. Federated learning: An approach with hybrid homomorphic encryption. *arXiv preprint arXiv:2509.03427*, 2025.

[10] P. M. da S. Correia. A hybrid homomorphic encryption approach for federated learning. Master's thesis, Universidade do Porto, 2025.

[11] S. Liu, Z. Chen, and L. Zhang. A privacy-preserving federated learning framework with byzantine robustness. In *2024 IEEE International Conference on Big Data (BigData)*. IEEE, 2025.

[12] P. Shen, S. Zhao, J. Xu, C. Zhao, Z. Chen, and S. Guo. Lbrfl: Lightweight privacy-preserving federated learning with byzantine-robustness. In *Neural Information Processing: 31st International Conference, ICONIP 2024*. Springer, 2024.

[13] Y. Ding, H. Wang, Y. Li, and et al. Secure and verifiable data collaboration with low-cost zero-knowledge proofs. *Proceedings of the VLDB Endowment*, 17(8):1975–1987, 2024.

[14] Y. Li, Z. Wang, Q. Liu, and et al. Verifbfl: Leveraging zk-snarks for a verifiable blockchained federated learning. In *2025 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. IEEE, 2025.

[15] Y. Zhao, J. Zhao, L. Jiang, R. Tan, D. Niyato, Z. Li, L. Lyu, and Y. Liu. A privacy-preserving and verifiable federated learning method based on blockchain. *Computer Networks*, 204:108716, 2022.

[16] H. Wang, J. Feng, M. Bu, Y. Ding, S. Tang, and C. Yang. Mvfl: verifiable privacy-preserving federated learning using multi-key homomorphic encryption. *The Journal of Supercomputing*, 81 (8):1–22, 2025.

[17] T. Nguyen. *Vertrauenswürdiges kollaboratives maschinelles Lernen für Edge-KI: Datenschutz, Unlearning und Robustheit*. PhD thesis, Technische Universität Berlin, 2026.

[18] A. Singh, P. Sharma, and N. Kumar. A secure and privacy-preserving federated learning-based intrusion detection system for sdn networks using homomorphic encryption. In *2026 IEEE International Conference on Communications (ICC)*. IEEE, 2026.