This article has been accepted for publication in IEEE Transactions on Big Data. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2024.3403370

zkFL: Zero-Knowledge Proof-based Gradient Aggregation for Federated Learning

Zhipeng Wang, Nanqing Dong, Jiahao Sun, William Knottenbelt, and Yike Guo *Fellow, IEEE*

***Abstract*—Federated learning (FL) is a machine learning paradigm, which enables multiple and decentralized clients to collaboratively train a model under the orchestration of a central aggregator. FL can be a scalable machine learning solution in *big data* scenarios. Traditional FL relies on the trust assumption of the central aggregator, which forms cohorts of clients honestly. However, a malicious aggregator, in reality, could abandon and replace the client's training models, or insert fake clients, to manipulate the final training results. In this work, we introduce `zkFL`, which leverages zero-knowledge proofs to tackle the issue of a malicious aggregator during the training model aggregation process. To guarantee the correct aggregation results, the aggregator provides a proof per round, demonstrating to the clients that the aggregator executes the intended behavior faithfully. To further reduce the verification cost of clients, we use blockchain to handle the proof in a zero-knowledge way, where miners (*i.e.*, the participants validating and maintaining the blockchain data) can verify the proof without knowing the clients' local and aggregated models. The theoretical analysis and empirical results show that `zkFL` achieves better security and privacy than traditional FL, without modifying the underlying FL network structure or heavily compromising the training speed.**

***Index Terms*—Federated Learning, Security, Trustworthy Machine Learning, Zero-Knowledge Proof**

I. Introduction

FEDERATED learning (FL) is a privacy-preserving machine learning paradigm that allows multiple clients to collaboratively train a global model without sharing their raw data [1], [2], [3], [4]. FL can be a scalable solution for machine learning in *big data* scenarios [5], where large-scale data are generated and stored by multiple clients in different physical locations. In FL, each participant (*i.e.*, client) performs local training on its own private dataset and communicates only the model updates to the central server (*i.e.*, aggregator). This decentralized approach minimizes the need to transfer large volumes of data to the aggregator. The aggregator then aggregates the model updates and sends the updated global model back to the clients. This process repeats iteratively until the global model converges or a stopping criterion is fulfilled. During the cross-device FL process, participants need to place their trust in the aggregator to create cohorts of clients in a fair and unbiased manner. However, a potential vulnerability is that an actively malicious adversary with control over the aggregator could exploit this trust [6]. For instance, adversaries could carry out a Sybil attack [7] by simulating numerous fake client devices, and the adversary could also selectively favor previously compromised clients' model updates from the pool of available participants. These attacks have the potential to enable the adversary to manipulate the final training results in Federated Learning (FL), compromising the integrity of the learning process. Safeguarding against such threats is imperative to maintain the effectiveness and security of cross-device FL.

A. Contributions

In this work, we present `zkFL` (*cf.* Fig. 1), an innovative approach that integrates zero-knowledge proofs (ZKPs) into FL. Without changing the learning setup of the underlying FL method, this integration guarantees the integrity of aggregated data from the centralized aggregator. Zero-knowledge proofs (ZKPs) [8], [9], [10], [11] are widely recognized cryptographic tools that enable secure and private computations while safeguarding the underlying data. In essence, ZKPs empower a prover to convince a verifier of a specific fact without revealing any information beyond that fact itself. Within the context of `zkFL`, ZKPs play a pivotal role in addressing the challenge posed by a potentially malicious aggregator during the model aggregation process. To achieve accurate aggregation results, the aggregator must provide a proof for each round, demonstrating to the clients that it has faithfully executed the intended behavior for aggregating the model updates. By verifying these proofs, the clients can ensure the aggregator's actions are transparent and verifiable, instilling confidence that the aggregation process is conducted with utmost honesty.

Furthermore, in order to minimize the verification burden on the clients, we propose a blockchain-based `zkFL` solution to handle the proof in a zero-knowledge manner. As shown in Fig. 2, in this approach, the blockchain acts as a decentralized and trustless platform, allowing *miners*, the nodes validating and maintaining the blockchain data [12], [13], [14], to verify the authenticity of the ZKP proof without compromising the confidentiality of the clients' models. By incorporating blockchain technology into our `zkFL` system, we establish a

This work was supported in part by FLock.io under the FLock Research Grant.

Z. Wang and W. Knottenbelt are with the Department of Computing, Imperial College London, London, SW7 2AZ, UK. (emails: zhipeng.wang20@imperial.ac.uk, w.knottenbelt@imperial.ac.uk)

N. Dong is with the Department of Computer Science, University of Oxford, Oxford, OX1 3QD, UK; and also with the Shanghai Artificial Intelligence Laboratory, Shanghai 200232, China. (email: nanqing.dong@cs.ox.ac.uk)

J. Sun is with the Data Science Institute, Imperial College London, SW7 2AZ, UK; and also with FLock.io, London, WC2H 9JQ, UK. (email: jiahao.sun@imperial.ac.uk)

Y. Guo is with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong SAR, China; and also with the Data Science Institute, Imperial College London, London, SW7 2AZ, UK. (email:yikeguo@ust.hk)

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

This article has been accepted for publication in IEEE Transactions on Big Data. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2024.3403370

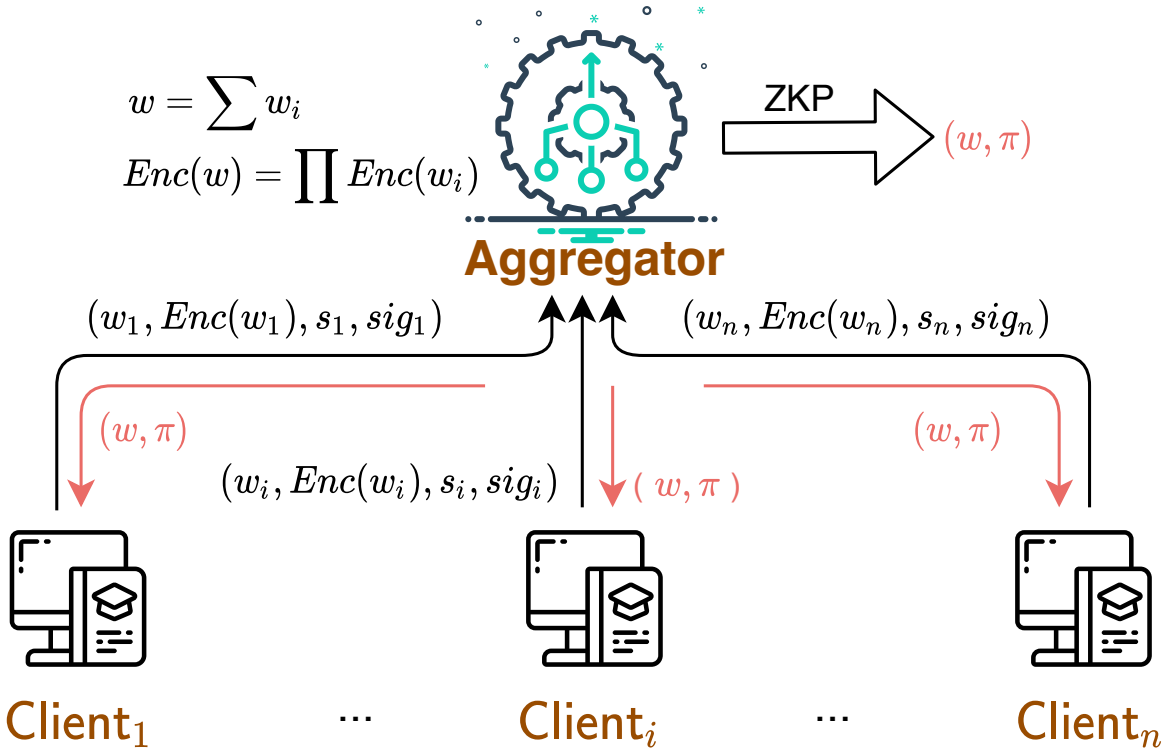


Fig. 1: Overview of zkFL system. During each round, the clients send their local model update w_i, the encrypted model update $Enc(w_i)$, a chosen random number s_i, and their signature sig_i to the aggregator. The aggregator generates the aggregated model update w and leverages zero-knowledge proofs (ZKPs) to generate a proof π, and the clients will verify the proof to ensure that the training model updates are correctly aggregated. zkFL guarantees the integrity of the data computed by the aggregator, and enhances security and trust in the collaborative FL setting.

robust and scalable framework for conducting zero-knowledge proof verification in a decentralized and transparent manner. This not only enhances the overall efficiency of the zkFL system but also reinforces the confidentiality of the participants' data, making it a promising solution for secure and privacy-conscious cross-device FL.

Our contributions can be summarized as follows:

- We present zkFL, an innovative ZKP-based FL system that can be integrated with existing FL methods. zkFL empowers clients to independently verify proofs generated by the centralized aggregator, thereby ensuring the accuracy and validity of model aggregation results. zkFL effectively addresses the threats posed by the malicious aggregators during the training model aggregation process, enhancing security and trust in the collaborative FL setting.
- We integrate zkFL with blockchain technology to minimize clients' computation costs for verification. Leveraging the *zero-knowledge* property of ZKPs, our blockchain-based zkFL significantly improves overall efficiency while preserving clients' model privacy, thereby rendering it more scalable for FL in big data environments.
- We present rigorous theoretical analysis on the security, privacy, and efficiency of zkFL. We further evaluate these properties under benchmark FL setups. The results of these experiments demonstrate the practical feasibility and effectiveness of zkFL in real-world scenarios.

Paper Organization. The remaining part of this paper is structured as follows. Section II reviews related work in Zero-Knowledge Proofs (ZKPs) and Blockchain-based Federated Learning (FL). Section III outlines the preliminary concepts essential for our zkFL and blockchain-based zkFL frameworks. Our system and threat models are detailed in Section IV, followed by our methodology for developing zkFL and blockchain-based zkFL in Section VI. Sections VI and VII provide theoretical and empirical analyses of our constructions, respectively. Future directions are discussed in Section VIII, and the paper concludes in Section IX.

II. Related Work

A. Zero-Knowledge Proofs

Zero-knowledge proofs (ZKPs) have emerged as a revolutionary cryptographic concept that allows one party (the prover) to demonstrate the truth of a statement to another party (the verifier) without revealing any information beyond the statement's validity [8], [15]. One of the most notable applications of ZKPs is in privacy-preserving authentication, where a user can prove to an aggregator that they know a secret without revealing the secret itself. ZKPs have also been applied in other areas, including blockchains [9], [16], machine learning [11], [17], [18] and FL.

For instance, Burkhalter *et al*. introduce RoFL [10], which is a secure FL with ZKPs that enables the aggregator to

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

This article has been accepted for publication in IEEE Transactions on Big Data. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2024.3403370

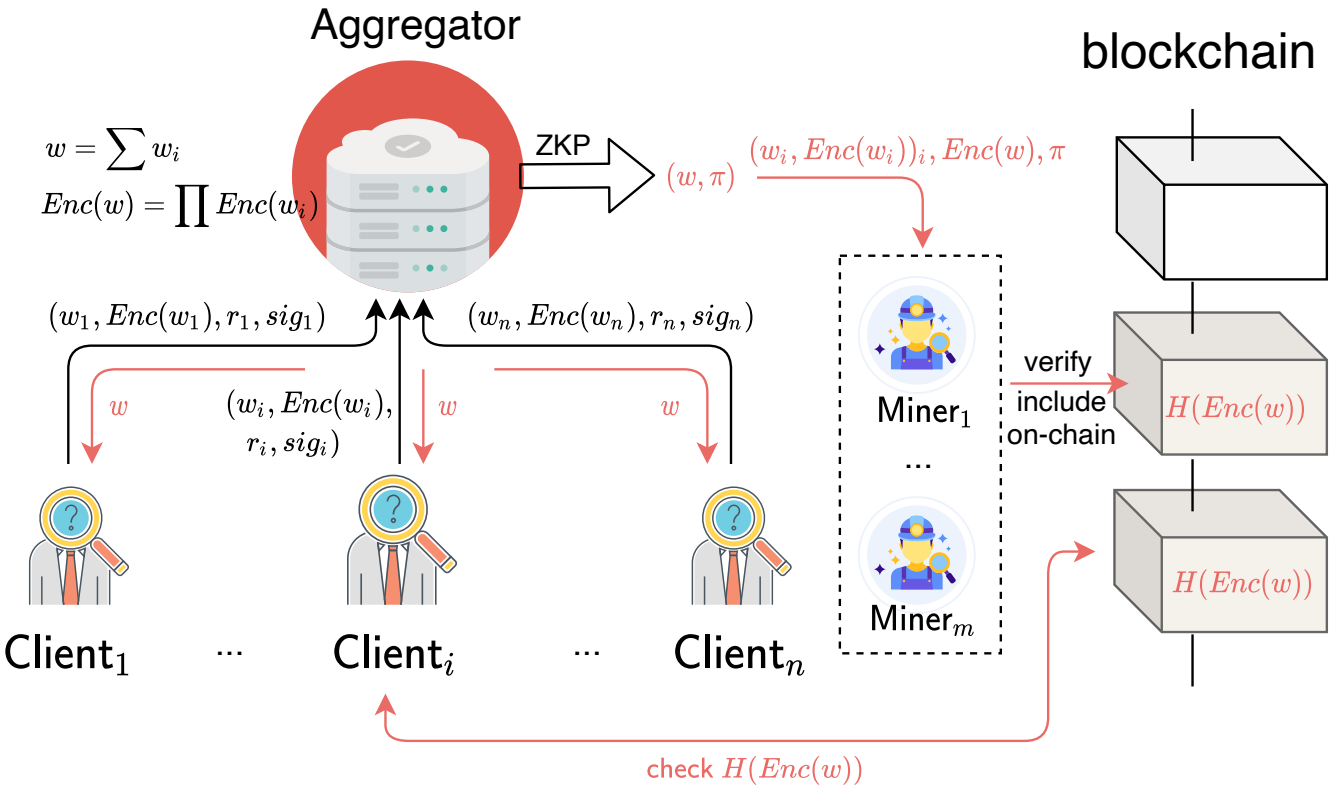


Fig. 2: Overview of zkFL system with blockchain. The proof π generated by the aggregator will be verified by the blockchain miners given the encrypted model updates $Enc(w_1), ..., Enc(w_n), Enc(w)$. This approach can reduce the computation cost of the clients, without compromising the privacy of the clients' models.

TABLE I: Comparison of zkFL and blockchain-based zkFL with existing work.

Framework	Techniques	Without Trust in Aggregator	Without On-Chain Aggregation
RoFL [10]	ZKPs	✗	-
RiseFL [19]	ZKPs	✗	-
PZKP-FL [20]	ZKPs + blockchain	✓	✗
stake-based FL [21]	blockchain	✓	✗
zkFL	ZKPs	-	✓
blockchain-based zkFL	ZKPs + blockchain	✓	✓

enforce and verify constraints on client updates. Zhu *et al*. propose RiseFL [19] to ensure input data privacy and integrity from the FL clients. Their approach employs a probabilistic integrity checking mechanism within ZKPs, combined with a hybrid commitment scheme, to enhance system performance effectively Xing *et al*. propose PZKP-FL [20] which adopts ZKPs to validate the computation process without disclosing the clients' local data in plaintext.

However, as shown in Table I, existing research of ZKPs-based FL designs [10], [19], has predominantly focused on using ZKPs to address malicious client behaviors or enhance the client privacy, while assuming the existence of a "honest-but-curious" (*i.e.*, semi-honest) aggregator. In this work, we break away from this assumption and leverage ZKPs to ensure the honest aggregation of the centralized aggregator without requiring trust in the aggregator itself.

B. Blockchain-based Federated Learning

Blockchain is a decentralized and immutable distributed ledger technology [22], [13] that underpins cryptocurrencies such as Bitcoin [12] and Ethereum [23]. It provides a secure and transparent way to record and verify transactions across a network of nodes. Blockchain has been integrated with FL to tackle the security and privacy issues in existing FL [24], [25], [26], [27], [28]. Specifically, blockchain is used to store and manage the training model's updates and the associated metadata securely and transparently. Instead of relying solely on a centralized aggregator to manage the model updates, the blockchain enables a decentralized and distributed consensus mechanism among the participating clients. However, existing blockchain-based FL designs rely heavily on the on-chain computation. For example, in the design of PZKP-FL [20], a secure sum protocol on blockchain is used to achieve the public verification of global aggregation. Dong *et al*. [21] propose a FL framework with blockchain-based staking and voting scheme to mitigate the malicious behaviors from FL clients, in which the aggregation process is performed on-chain with smart contract (*i.e.*, self-executing programs that run on a blockchain network and are triggered by blockchain events). Such on-chain aggregation will cause expensive costs for FL networks with a large number of parameters. As shown

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

This article has been accepted for publication in IEEE Transactions on Big Data. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2024.3403370

in Table I, we propose blockchain-based zkFL, which can address the scalability issue of blockchain-based FL by using ZKPs to remove the on-chain aggregation process.

III. Preliminaries

In this section, we present the cryptographic building blocks for our zkFL systems.

A. Hash Functions

In this work, we consider a cryptographic hash function H characterized by $H : \{0,1\}^* \rightarrow \{0,1\}^\lambda$, capable of mapping inputs of arbitrary length to outputs of fixed length. The collision resistance of H is defined such that for any probabilistic polynomial-time (PPT) algorithm $\mathcal{A}$, the probability that $\mathcal{A}$ finds distinct inputs x and x' such that $H(x) = H(x')$ is negligible. This property holds even when $\mathcal{A}$ has knowledge of other hash outputs. We will employ this collision-resistant hash function H in the construction of our blockchain-based zkFL. For detailed insights into hash functions, readers are referred to [29].

B. Commitments

A commitment scheme enables an entity to conceal a value while committing to it, with the ability to disclose the value at a later time if desired. The commitment scheme typically comprises two rounds: the committing round and the revealing round. In the committing round, the client commits to specific values while ensuring their confidentiality from others. The client retains the option to reveal the committed value in the subsequent revealing round. A commitment scheme includes two algorithms:

- $\mathsf{cm} \leftarrow \textsc{Commit}(m, r)$ accepts a message m and a secret randomness r as inputs and returns the commitment cm.
- $0/1 \leftarrow \textsc{Verify}(m, r, \mathsf{cm})$ accepts a message m, a commitment cm and a decommitment value r as inputs, and returns 1 if the commitment is opened correctly and 0 otherwise.

In this paper, we leverage Pedersen commitments [30], [31] to compute the clients' commitments/encryption on their local training model updates. Specifically, given the model update w_i, a client will encrypt the update by computing $Enc(w_i) = g^{w_i} \cdot h^{s_i}$, where g and h are public parameters and s_i is a random number generated by the client.

C. Zero-Knowledge Proof

Zero-knowledge proof [32], [33], [15] is a cryptographic primitive that allows a prover to convince the verifier about the correctness of some assertions without providing any meaningful information to the verifier. A zero-knowledge proof of some statement satisfies the following three properties:

- *Completeness:* If the statement *st* is true, an honest verifier will always be convinced by an honest prover.
- *Soundness:* For false statements, a prover cannot convince the verifier (even if the prover deviates from the protocol).
- *Zero-knowledge:* No verifier learns anything other than the fact that *st* is valid if it is true. In other words, knowing the statement, but not the secret, is enough to construct a scenario in which the prover knows the secret.

A zero-knowledge Succinct Non-interactive ARgument of Knowledge (zk-SNARK) is a "succinct" non-interactive zero-knowledge proof (NIZK) for arithmetic circuit satisfiability. The construction of zk-SNARK is based on a field $\mathbb{F}$ and an arithmetic circuit C. We adopt the definition of zk-SNARK from [9]: An arithmetic circuit satisfiability problem of a circuit $C : \mathbb{F}^n \times \mathbb{F}^h \rightarrow \mathbb{F}^l$ is captured by the relationship $R_C : \{(x, \mathsf{wit}) \in \mathbb{F}^n \times \mathbb{F}^h : C(x, \mathsf{wit}) = 0^l\}$, with the language $\mathcal{L}_C = \{x \in \mathbb{F}^n : \exists\ \mathsf{wit} \in \mathbb{F}^h\ s.t.\ C(x, \mathsf{wit}) = 0^l\}$. A zk-SNARK for an arithmetic circuit satisfiability problem consists of the following algorithms:

- $(\mathsf{pk}, \mathsf{vk}) \leftarrow \textsc{KeyGen}(1^\lambda, C)$: On input the security parameter 1^λ and the arithmetic circuit C, this algorithm outputs a proving key pk and a verification key vk.
- $\pi \leftarrow \textsc{Prove}(\mathsf{pk}, x, \mathsf{wit})$: Given the proving key pk and $(x, \mathsf{wit}) \in R_C$, this algorithm outputs a proof π for the statement $x \in \mathcal{L}_C$.
- $\mathsf{True}/\mathsf{False} \leftarrow \textsc{Verify}(\mathsf{vk}, \pi, x)$: Taking the verification key vk, the proof π, and the statement x as input, this algorithm outputs True if π is a valid proof for the statement $x \in \mathcal{L}_C$; otherwise, it outputs False.

In addition to the fundamental properties of *correctness, soundness,* and *zero-knowledge* inherent in ZKPs, a zk-SNARK can exhibit additional essential characteristics. One such crucial aspect is *succinctness* [9], which demonstrates that the computational complexity of the $\textsc{Verify}(\cdot)$ algorithm is linear with the size of x, denoted as $O_\lambda(|x|)$[1]. Furthermore, the proof π generated by an honest prover maintains a constant size in relation to x, denoted as $O_\lambda(1)$.

IV. System and Threat Models

In this section, we outline our system model, threat model, and system goals.

A. System Model

- **Clients**: In the context of FL, clients represent individual devices, such as smartphones, tablets, or computers, each possessing its local dataset. These datasets remain secure and never leave the clients' devices. Instead, the clients independently train their machine learning models on their local data and communicate only the model updates to the central aggregator.
- **Aggregator**: The aggregator acts as a central entity responsible for aggregating these model updates from multiple clients and computing a global model. This global model is then sent back to the clients, ensuring that each client benefits from the collective knowledge of the entire network while preserving data privacy and security.

[1]We adhere to the notation introduced in [9]: $O_\lambda(\cdot)$ conceals a fixed polynomial factor in λ.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

This article has been accepted for publication in IEEE Transactions on Big Data. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2024.3403370

B. Threat Model

We consider a malicious aggregator that can choose not to honestly aggregate the local model updates from clients. The malicious aggregator can deviate from the protocol by:

- *Abandoning* the updates generated from one or several honest clients.
- Creating fake model updates to *replace* the updates generated from honest clients,
- *Inserting* fake model updates to the updates generated from honest clients.

We would like to remark that the scope of this work centers around the malicious aggregator rather than the honest-but-curious one, with a specific emphasis on ensuring aggregation integrity. We also acknowledge the potential for the aggregator to carry out model inversion attacks on the clients, a topic we intend to delve into in a future study.

C. System Goals

- **Security**: The aggregator cannot abandon or replace the local model updates generated from honest clients, nor insert any fake model updates into the final aggregated model update. Otherwise, the clients will detect the malicious behaviors of the aggregator and halt the FL training process.
- **Privacy**: Only the participants (*i.e.*, the aggregator and clients) in the FL system can know the aggregated model updates during each round.

V. Methodology

A. zkFL

As shown in Fig. 1, our zkFL system works as follows:

1) **Setup:** N clients and one aggregator generate their private/public key pairs and set up communication channels. Each client knows the public keys of the other $n-1$ clients, and this setup can be achieved by using a public key infrastructure (PKI).
2) **Local Training, Encrypting, and Signing:** During each round, the n clients train their models locally to compute the local model updates $w_1, w_2, \ldots, w_n$. Each client encrypts their update $Enc(w_i) = g^{w_i} \cdot h^{s_i}$ using Pedersen commitment, where g and h are public parameters and s_i is a random number generated by the client. The client signs the encrypted updates with their private key to generate a signature sig_i. The client then sends the tuple of local model update, the randomly generated number, encrypted local model update, and signature $(w_i, s_i, Enc(w_i), sig_i)$ to the aggregator.
3) **Global Aggregation and ZKP Generation:** The aggregator aggregates the received local model updates $w_1, w_2, \ldots, w_n$ to generate the aggregated global model update $w = \sum_{i=1}^{n} w_i$. The aggregator also computes the aggregated value of the encrypted global model update $Enc(w) = \prod_{i=1}^{n} Enc(w_i)$ and signs it with its private key to generate the signature sig. The aggregator then leverages zk-SNARK to issue a proof π for the following statement and witness:

$$\begin{cases} statement = (Enc(w_1), sig_1, Enc(w_2), sig_2, \\ \qquad\qquad\qquad ..., Enc(w_n), sig_n, Enc(w)) \\ witness = (w_1, s_1, w_2, s_2, ..., w_n, s_n, w) \end{cases}$$

where the corresponding circuit $C(statement, witness)$ outputs 0 if and only if:

$$\begin{cases} \text{(i)} & \forall 1 \le i \le n, Enc(w_i) = g^{w_i} \cdot h^{s_i} \\ \text{(ii)} & w = \sum_{i=1}^{n} w_i \\ \text{(iii)} & sig_i \text{ is signed by the client } i \end{cases}$$

4) **Global Model Transmission and Proof Broadcast:** The aggregator transfers the aggregated global model update w, its encryption $Enc(w)$ and the proof π to the n clients.
5) **Verification:** Upon receiving the proof π and the encrypted global model update $Enc(w)$ from the aggregator, the clients verify if π is valid. When the verification is passed, the clients start their local training based on the aggregated global model update w.

B. Blockchain-based zkFL

To decrease the computation burden on clients, we incorporate blockchain technology into our zkFL system. In this approach, the verification of proofs generated by the aggregator is entrusted to blockchain *miners*. Illustrated in Fig. 2, the blockchain-based zkFL operates as follows:

1) **Setup:** N clients and one aggregator generate their private/public key pairs, which correspond to their on-chain addresses.
2) **Selection:** For each round, n clients are selected from the N clients via Verifiable Random Functions [34], [35]. The n selected clients' public keys are broadcasted to the underlying P2P network of the blockchain, which will be received and verified by the miners.
3) **Local Training, Encrypting, and Signing:** The n selected clients train their models locally to compute the local model updates $w_1, w_2, \ldots, w_n$. Each client encrypts their update $Enc(w_i) = g^{w_i} \cdot h^{s_i}$ using Pedersen commitment, where g and h are public parameters and s_i is a random number generated by the client. The client signs the encrypted updates with their private key to generate a signature sig_i. The client then sends the tuple of local model update, the randomly generated number, encrypted local model update, and signature $(w_i, s_i, Enc(w_i), sig_i)$ to the aggregator.
4) **Global Aggregation and ZKP Generation:** The aggregator aggregates the received local model updates $w_1, w_2, \ldots, w_n$ to generate the aggregated global model update $w = \sum_{i=1}^{n} w_i$. The aggregator also computes the aggregated value of the encrypted global model update $Enc(w) = \prod_{i=1}^{n} Enc(w_i)$ and signs it with its private key to generate the signature sig. The aggregator then leverages

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

This article has been accepted for publication in IEEE Transactions on Big Data. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2024.3403370

zk-SNARK to issue a proof π for the following statement and witness:

$$\begin{cases} statement = (Enc(w_1), sig_1, Enc(w_2), sig_2, \\ \qquad\qquad ..., Enc(w_n), sig_n, Enc(w)) \\ witness = (w_1, s_1, w_2, s_2, ..., w_n, s_n, w) \end{cases}$$

where the corresponding circuit $C(statement, witness)$ outputs 0 if and only if:

$$\begin{cases} \text{(i)} & \forall 1 \leq i \leq n, Enc(w_i) = g^{w_i} \cdot h^{s_i} \\ \text{(ii)} & w = \sum_{i=1}^{n} w_i \\ \text{(iii)} & sig_i \text{ is signed by the client } i \end{cases}$$

5) **Global Model Transmission and Proof Broadcast:** The aggregator transfers the aggregated global model update w and its encryption $Enc(w)$ to the n clients, and broadcasts the proof π, and the encrypted global model update $Enc(w)$ to the miners over the P2P network.
6) **On-Chain Verification:** Upon receiving the proof π and the encrypted global model update $Enc(w)$ from the aggregator, the miners verify π and append the hash value of $H(Enc(w))$ to the blockchain if π is valid.
7) **On-Chain Reading:** When the next round starts, the newly selected n clients read the blockchain to check if $H(Enc(w))$ is appended on-chain. When the check is valid, the clients start their local training based on the aggregated global model update w.

VI. Theoretical Analysis

In the following, we provide analyses to show that our zkFL and blockchain-based zkFL systems can achieve the goals of security and privacy, while merely bringing an acceptable decrease in the FL training efficiency.

A. Security Analysis

Our zkFL and blockchain-based zkFL can achieve the security goal through the following techniques:

- The signatures sig_i of each encrypted local update $Enc(w_i)$ ensure the local updates' integrity from the clients and prevent the aggregator from tampering with their content and the statement $statement = (Enc(w_1), sig_1, ..., Enc(w_n), sig_n, Enc(w))$.
- The *completeness* and *soundness* properties of the ZKP proof π play a critical role in safeguarding the aggregation process from adversarial manipulation by the aggregator. These properties ensure that the aggregator cannot deviate from the intended behaviors:
  - If the aggregator abandons the model update generated from one client j, then the aggregated results will be $\hat{w} = \sum_{i=1, j\neq i}^{n} w_i$. In this case, the corresponding circuit $C(statement, witness)$ outputs 0 and the proof π will be invalid.
  - If the aggregator replaces the model update generated from one client j by $\hat{w_j}$, then the aggregated results will be $\hat{w} = \sum_{i=1, j\neq i}^{n} w_i + \hat{w_j}$. In this case, the corresponding circuit $C(statement, witness)$ outputs 0 and the proof π will be invalid.
  - If the aggregator inserts one fake model update generated $\hat{w_0}$, then the aggregated results will be $\hat{w} = \sum_{i=1}^{n} w_i + \hat{w_0}$. In this case, the corresponding circuit $C(statement, witness)$ outputs 0 and the proof π will be invalid.

Therefore, the proof π will only be valid if the aggregator honestly conducts the local model updates aggregation to generate the global model update $w = \sum_{i=1}^{n} w_i$.

B. Privacy Analysis

In zkFL, privacy is inherently ensured as only the aggregator and the participating clients are involved in the training and aggregation process, eliminating the need for external parties. As a result, only these authorized entities possess knowledge of the aggregated model updates at each round.

In the context of the blockchain-based zkFL system, the blockchain miners receive encrypted the local model updates $Enc(w_i)$ $(1 \leq i \leq n)$, the encrypted global model update $Enc(w)$, and the ZKP proof π from the aggregator. However, due to the *zero-knowledge* property of ZKP, the miners can only verify whether $Enc(w)$ is correctly executed or not, without gaining any access to information about the individual local model updates w_i $(1 \leq i \leq n)$ or the global model update w. Additionally, storing the encrypted data of $Enc(w)$ on the blockchain does not compromise the privacy of the global model update w. Our system maintains a robust level of privacy throughout the blockchain-based zkFL process.

C. Efficiency Analysis

In the following, we calculate the expected computation time of the aggregator and a client per round, to analyze the system efficiency of zkFL and blockchain-based zkFL.

In both zkFL and blockchain-based zkFL systems, the aggregator is responsible for aggregating the local model updates and generating the ZKP proof. The expected computation time of the aggregator is:

$$\mathbb{E}_{aggregator} = \mathbb{E}(aggr) + \mathbb{E}(ZKP.gen)$$

In the zkFL system, a client needs to train the local model, encrypt the local model update, and verify the ZKP proof generated by the aggregator. The expected computation time of a client is:

$$\mathbb{E}_{client} = \mathbb{E}(train) + \mathbb{E}(enc) + \mathbb{E}(ZKP.ver)$$

In the blockchain-based zkFL system, a client still needs to train the local model and encrypt the local model update. However, the blockchain miners will verify the ZKP proof generated by the aggregator, and the clients only need to read the data on the blockchain. The expected computation time of a client is:

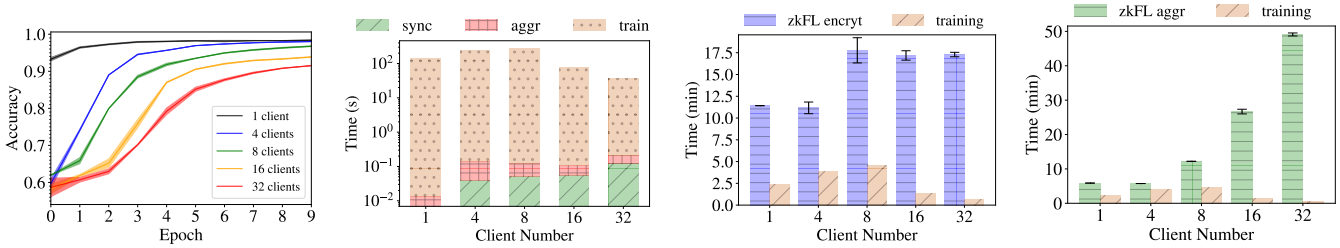
$$\mathbb{E}^{block}_{client} = \mathbb{E}(train) + \mathbb{E}(enc) + \mathcal{O}(1)$$

VII. Empirical Analysis

In this section, we quantify the overhead of zkFL and show that it can be used to train practical FL models.

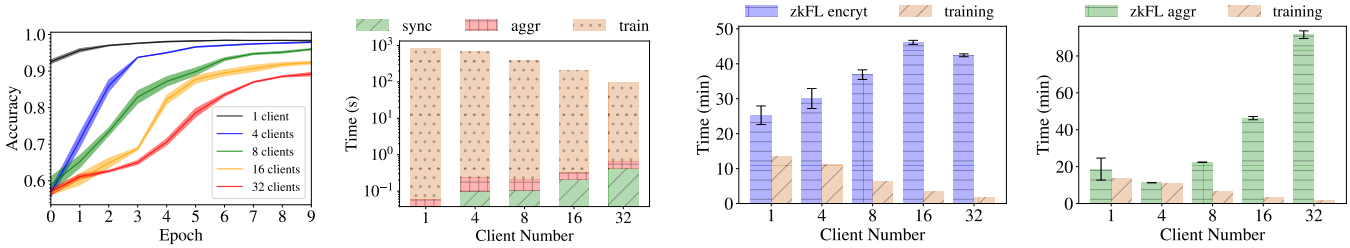
This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

This article has been accepted for publication in IEEE Transactions on Big Data. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2024.3403370



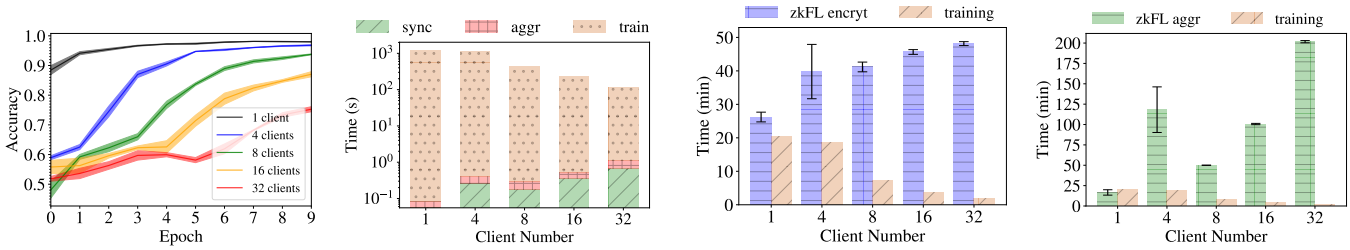
(a) Training accuracy. (b) Total training time. (c) zkFL encryption time. (d) zkFL aggregation time.

Fig. 3: Accuracy, training, encryption and aggregation time for Resnet18 on CIFAR-10 with zkFL system.



(a) Training accuracy. (b) Total training time. (c) zkFL encryption time. (d) zkFL aggregation time.

Fig. 4: Accuracy, training, encryption, and aggregation time for Resnet34 on CIFAR-10 with zkFL system.



(a) Training accuracy. (b) Total training time. (c) zkFL encryption time. (d) zkFL aggregation time.

Fig. 5: Accuracy, training, encryption, and aggregation time for Resnet50 on CIFAR-10 with zkFL system.

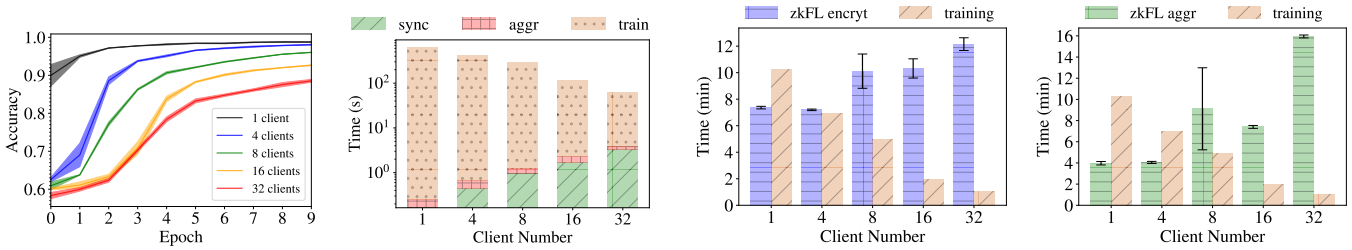
A. *Experiment Setup*

1) Data and Task: We consider two benchmark machine learning tasks under the common federated setup. We use FedAVG [1] as the base FL method to evaluate zkFL. In each epoch, clients train their models separately with local data. Then, the aggregator synchronizes local models and performs the model evaluation. The training set is split into K subsets of equal size and distributed across K clients. For each model of interest, we conduct the tasks for zkFL with 1, 4, 8, 16, and 32 client(s). For each task, we record the training time without using zkFL, the encryption and aggregation time with zkFL, as well as the ZKP generation and verification time under zkFL. We also evaluate the performance of each task with various network backbones and client numbers.

- **Image Classification**: We consider image classification on CIFAR-10 dataset, a benchmark computer vision task [36]. We use the default train-test split of CIFAR-10. For each client, 10% of distributed data is assigned as the validation set. We use a standard Adam optimizer [37] with fixed learning rate 0.001 and batch size 50. We test our zkFL system with two families of network architectures: ResNets [38] (ResNet18, ResNet34, and ResNet50) and DenseNets [39] (DenseNet121, DenseNet169, and DenseNet201). We use this setup to evaluate the sensitivity of zkFL to network architectures and number of parameters. We use the *area under the receiver operating curve* (AUROC) to evaluate the accuracy.
- **Language Understanding**: We consider language modeling (word prediction) on the Penn Treebank (PTB) dataset, a benchmark natural language processing task [36]. We use the default train-validation-test split. We test our zkFL system with three different network architectures. We use long short-term memory (LSTM) network [40] to model the language. Each LSTM consists of an embedding layer, a few LSTM layers, and a linear layer. To perform a similar sensitivity analysis of zkFL as above, we consider LSTMs with one, two, and three LSTM layers, with 650 neurons in

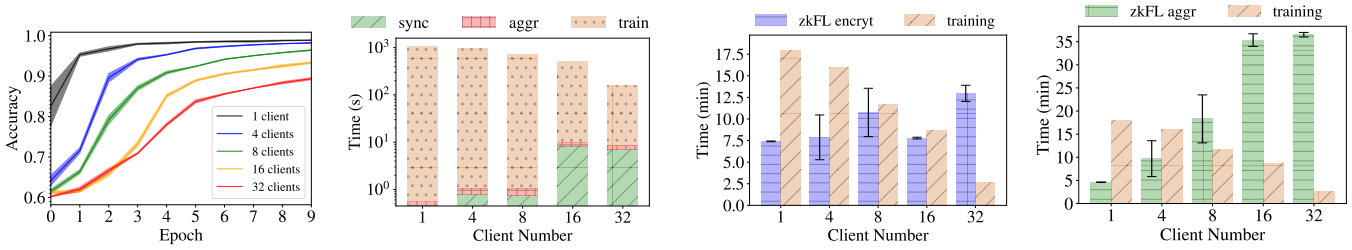
This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

This article has been accepted for publication in IEEE Transactions on Big Data. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2024.3403370



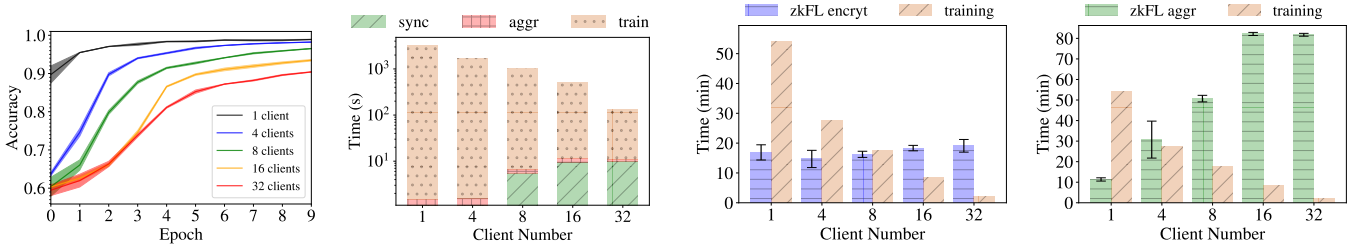
(a) Training accuracy. (b) Total training time. (c) zkFL encryption time. (d) zkFL aggregation time.

Fig. 6: Accuracy, training, encryption, and aggregation time for Densenet121 on CIFAR-10 with zkFL system.



(a) Training accuracy. (b) Total training time. (c) zkFL encryption time. (d) zkFL aggregation time.

Fig. 7: Accuracy, training, encryption, and aggregation time for Densenet169 on CIFAR-10 with zkFL system.



(a) Training accuracy. (b) Total training time. (c) zkFL encryption time. (d) zkFL aggregation time.

Fig. 8: Accuracy, training, encryption, and aggregation time for Densenet201 on CIFAR-10 with zkFL system.

each layer. We use a standard Adam optimizer with fixed learning rate 0.001 and batch size 20. The dropout ratio is 0.5.

2) Implementation: We develop a prototype of zkFL. We implement the prototype in Rust and interface it with Python to train deep learning models with PyTorch 1.13. We adopt the elliptic curve Curve25519 (i.e. 126-bit security) implementation from the dalek curve25519 library for cryptographic encryption operations. We build Pedersen commitments [31] over the elliptic curves and integrate it with Halo2 [41], a ZKP system that is being used by the Zcash. All tests are implemented in PyTorch 1.10.1 on an NVIDIA Tesla T4 GPU with 16GB memory.

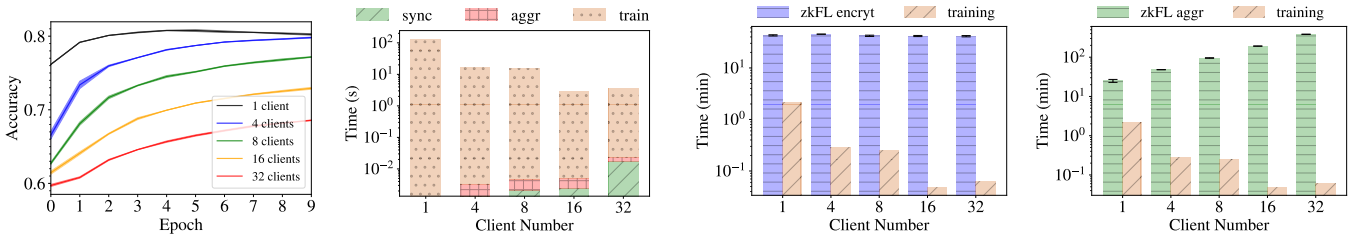
B. Results of zkFL

1) Training Time: We commence our evaluation by conducting an in-depth analysis of the total training time per epoch for each network backbone, focusing on the conventional FL approach without the integration of zkFL. This evaluation encompasses three crucial components that contribute to the overall training time: the local training time of each individual client, the time required for aggregating local model updates on the central aggregator, and the synchronization time. As shown in Figs. 3(b) 4(b) 5(b) 6(b) 7(b) 8(b) 9(b) 10(b) 11(b), our findings indicate that the local training time of each client is the primary contributor to the total training time. Moreover, as the number of clients increases, the local training time for individual clients decreases due to the effective distribution of training tasks among them.

2) Encryption and Aggregation Time: We conduct a thorough evaluation of the encryption time for clients and the aggregation time for the central aggregator within the zkFL system. In addition to the computational costs associated with the FL training protocol, the client-side tasks involve computing commitments, such as encryption computations for each model update parameter. Figs 3(c) 4(c) 5(c) 6(c) 7(c) 8(c) 9(c) 10(c) 11(c) demon-

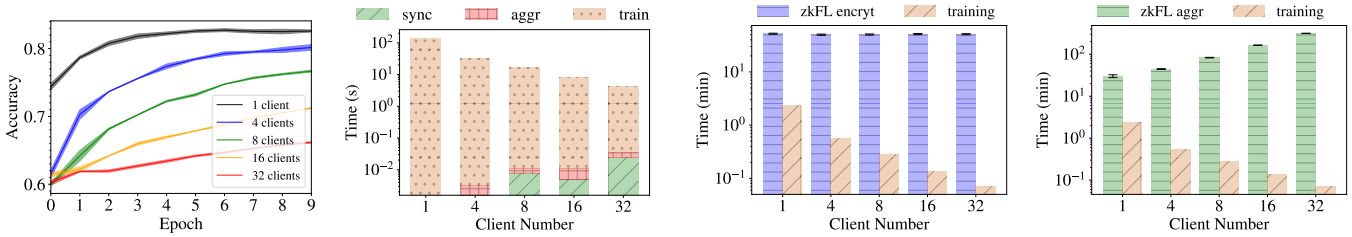
This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

This article has been accepted for publication in IEEE Transactions on Big Data. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2024.3403370



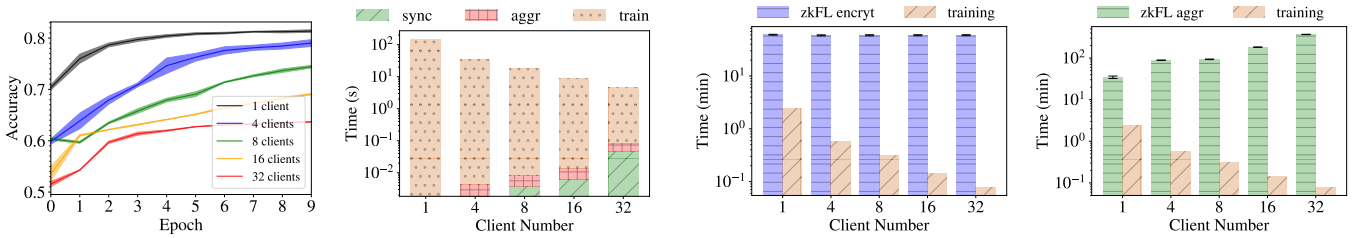
(a) Training perplexity. (b) Total training time. (c) zkFL encryption time. (d) zkFL aggregation time.

Fig. 9: Perplexity, training, encryption, and aggregation time for LSTM (one layer) on PTB with zkFL system.



(a) Training perplexity. (b) Total training time. (c) zkFL encryption time. (d) zkFL aggregation time.

Fig. 10: Perplexity, training, encryption, and aggregation time for LSTM (two layers) on PTB with zkFL system.



(a) Training perplexity. (b) Total training time. (c) zkFL encryption time. (d) zkFL aggregation time.

Fig. 11: Perplexity, training, encryption, and aggregation time for LSTM (three layers) on PTB with zkFL system.

strate that this additional cost varies based on the choice of underlying network backbones and increases with the number of parameters in the network. For instance, the encryption time for ResNet18 (i.e., $10-20$ mins) is lower than that for ResNet34 (i.e., $25-45$ mins), as the latter has a larger number of network parameters. Moreover, as illustrated in Figs 3(d) 4(d) 5(d) 6(d) 7(d) 8(d) 9(d) 10(d) 11(d), the aggregation time of the central aggregator in zkFL is influenced by the network parameters and exhibits an approximately linear relationship with the number of clients in the FL system, which has a critical effect on the whole system's efficiency.

3) ZKP Proof Generation and Verification Time: As depicted in Fig. 12, the time required for Halo2 ZKP generation and verification exhibits variations depending on the chosen network and increases with the size of the network parameters. Among the six networks evaluated, ResNet50 demands the longest time for proof generation, taking approximately 54.72 ± 0.12 mins. Notably, the proof verification time is approximately half of the generation time. This favorable characteristic makes zkFL more practical, as the aggregator, equipped with abundant computing power, can efficiently generate the proof, while the source-limited clients can verify it without significant computational overhead. This highlights the feasibility and applicability of zkFL in real-world scenarios, where the FL clients may have constrained resources compared to the central aggregator.

4) Communication Costs: Compared to the traditional FL, zkFL will also increase the communication costs for the clients and the aggregator, as the encrypted local training model updates $Enc(w_i)$ $(1 \leq i \leq n)$ and the ZKP proof π. As shown in Table II, for each network backbone, we compare the size of encrypted data with the size of the model updates in plaintext. Compared to traditional FL, zkFL will cause additional communication costs (e.g., the encrypted data). We estimate the proof size based on the network parameter size and the data provided in the Halo2 original paper [41]. We show that the size of encrypted data grows linearly in the number of parameters. Thus, the communication costs are dominated by the encrypted data. For ResNet50, the network backbone with the largest number of model parameters in our

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

This article has been accepted for publication in IEEE Transactions on Big Data. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2024.3403370

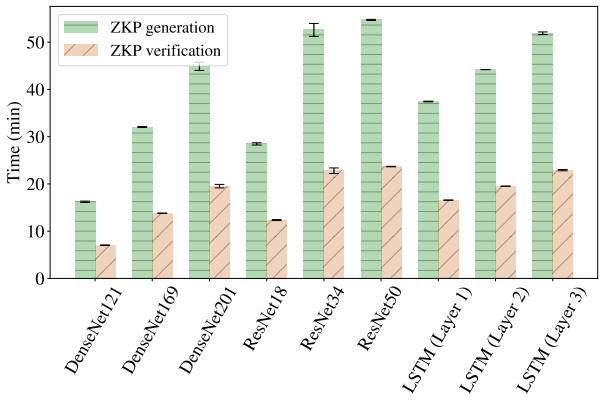


Fig. 12: Halo2 ZKP proof generation and verification time for a zkFL system with various network backbones.

TABLE II: Communication costs for zkFL with various network backbones. The increased communication costs in zkFL are dominated by the encrypted data.

Models	# Parameters	Model Updates in Plaintext	Encrypted Model Updates	Estimated ZKP Proof Size
DenseNet121	6,964,106	146MB	146MB	186KB
DenseNet169	12,501,130	262MB	262MB	334KB
DenseNet201	18,112,138	380MB	380MB	484KB
ResNet18	11,181,642	238MB	238MB	299KB
ResNet34	21,289,802	452MB	452MB	569KB
ResNet50	23,528,522	497MB	497MB	628KB
LSTM (one layer)	16,395,200	374MB	374MB	438KB
LSTM (two layers)	19,780,400	415MB	415MB	528KB
LSTM (three layers)	23,165,600	486MB	486MB	619KB

experiment, the additional data transmitted to the centralized aggregator per client is approximately $2\times$ compared to the plaintext). However, it is of utmost importance to underscore that although the relative size increase might appear significant, the absolute size of the updates is merely equivalent to a few minutes of compressed HD video. As a result, the data size remains well within the capabilities of the FL clients utilizing wireless or mobile connections. For instance, let us examine the most substantial communication overhead as outlined in Table II, specifically, a total data transfer volume of 491MB + 491MB + 628KB = 982.61MB, associated with the network architecture of ResNet50. In a scenario where a client transmits model updates in plaintext, alongside the encrypted model updates and the ZKP proof to a centralized aggregator, all over a network bandwidth of 1 Gbps, the resulting communication latency can be calculated as follows: $982.61 \times 8/1000 = 7.86$ seconds. This latency is approximately $2\times$ of the one in FL under the same communication network conditions.

5) Training Performance and Convergence Analysis: In addition to analyzing the additional computation and communication costs introduced by zkFL, we thoroughly investigate its potential impact on training performance, including accuracy and convergence speed. Theoretically, we demonstrate that compared to traditional FL, zkFL solely affects the data output from the clients, leaving the training process unaffected. Our experimental results provide strong evidence supporting this claim. Figs 3(a) 4(a) 5(a) 6(a) 7(a) 8(a) 9(a) 10(a) 11(a) present the final training results' accuracy/perplexity and convergence speed for both traditional FL and zkFL. We observe the accuracy and speed do not exhibit any difference between FL settings with and without zkFL in terms of epoch number. Furthermore, we observe that the convergence speed is primarily influenced by the number of clients involved in the process. This reaffirms the practical viability of zkFL as it maintains training performance while enhancing security and privacy in the federated learning framework.

C. Results of Blockchain-based zkFL

In this subsection, we present the results to show how blockchain-based zkFL affects the performance of the system.

1) Single Client Running Time: To understand the efficiency gains of blockchain-based zkFL over traditional zkFL, we first focus on the average runtime for a single client. As depicted in Figs. 13, 14, and 15, blockchain-based zkFL shows reduced client running time. As detailed in Section VI-C, this improvement stems from blockchain-based zkFL clients not having to verify ZKP proofs produced by the aggregator. Instead, blockchain miners undertake the verification, allowing clients to simply access the validated data, specifically the hash of the encrypted aggregated model updates.

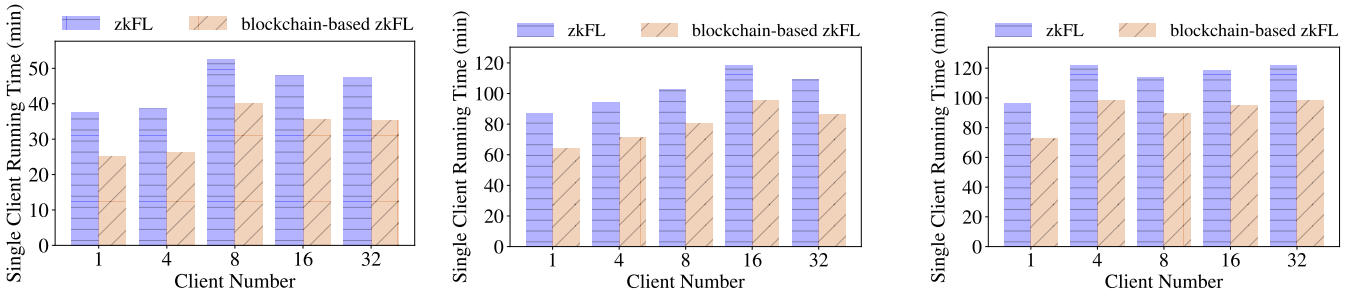
2) Training Security, Performance and Convergence Analysis: We then analyze how blockchain-based zkFL will affect the training convergence of FL. The clients in a blockchain-based zkFL rely on the blockchain miners to verify the ZKP proofs and then append the hash value of the encrypted aggregated model update, $H(Enc(w))$, into the blockchain. To ensure that the miners have correctly performed the verification, the clients need to wait for the transaction that contains $H(Enc(w))$ to be finalized, to guarantee the security of the system. We adopt the two prominent blockchains, Bitcoin and Ethereum, as examples. For Bitcoin, it takes approximately six blocks to ensure the finality of a transaction, which is about one hour [42]. On Ethereum, it takes about 15 minutes for a block to finalize[2]. Given these parameters, We graphically represent the training accuracy over time for standard zkFL, alongside its Bitcoin and Ethereum counterparts. Figs 16, 17, and 18 demonstrate that, despite the delays caused by the blockchain transaction finalization, blockchain-based zkFL achieves convergence in training accuracy, analogous to zkFL without blockchain.

3) On-Chain Costs: To compare the scalability of our blockchain-based zkFL with other blockchain-based FL such as [21], we analyze their on-chain costs on smart-contract enabled blockchain, i.e., Ethereum. The design in [21] involves performing the aggregation process on-chain and storing the aggregated model on-chain as well. This approach incurs an on-chain computation cost of at least $O(n \cdot m)$ and a storage cost of $O(m)$, where n represents the number of clients and m signifies the size of the aggregated model. Conversely, our blockchain-based zkFL framework optimizes resource

[2]https://ethereum.org/fil/roadmap/single-slot-finality

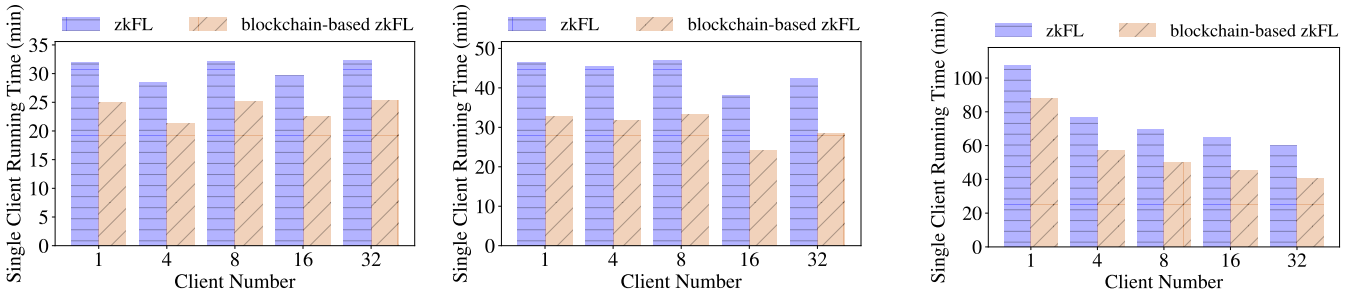
This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

This article has been accepted for publication in IEEE Transactions on Big Data. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2024.3403370



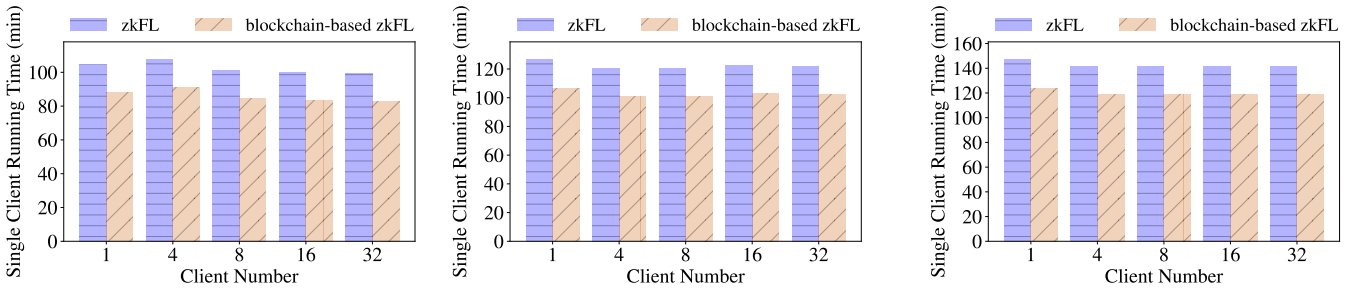
(a) Resnet18. (b) Resnet34. (c) Resnet50.

Fig. 13: Single client running time for Resnet18, Resnet34, and Resnet50 with zkFL and blockchain-based zkFL systems.



(a) Densenet121. (b) Densenet169. (c) Densenet201.

Fig. 14: Single client running time for Densenet121, Densenet169, and Densenet201 with zkFL and blockchain-based zkFL systems.



(a) LSTM Layer 1. (b) LSTM Layer 2. (c) LSTM Layer 3.

Fig. 15: Single client running time for LSTM Layer 1, 2, and 3 with zkFL and blockchain-based zkFL systems.

utilization by conducting the aggregation process off-chain and storing only the hash of the aggregated model on-chain, which reduces the cost to a constant $O(1)$.

For a practical comparison, we consider a scenario where the model size is 1MB, and the hash function employed is SHA-256. According to Ethereum's specifications [23], storing a 256-bit word requires 20,000 gas. Therefore, the method described in [21] would necessitate at least $\frac{1\times10^6\times8}{256}\times 20{,}000 = 625$M gas for storage alone. In stark contrast, our blockchain-based zkFL requires a mere 20,000 gas for storing the hash, highlighting a significant efficiency improvement. This disparity is further accentuated when accounting for the on-chain computation costs associated with [21].

VIII. Discussion

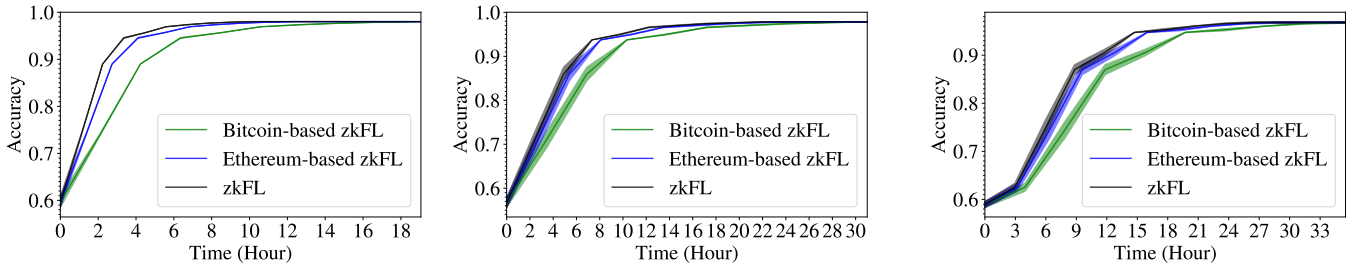
In the following, we discuss the limit of our zkFL designs and potential future work for improvement.

Decentralized Storage. In our blockchain-based zkFL design, the miners will only store the hash value of the encrypted aggregated model update $H(Enc(w))$ on-chain, rather than $Enc(w)$. This approach addresses the impracticality and high cost of storing large data on most existing blockchains. Moreover, the clients can directly receive w and $Enc(w)$ from the centralized aggregator. However, the $Enc(w)$ will be propagated to the blockchain miners and cause communication costs. To reduce the costs, we can leverage decentralized storage platforms, such as IPFS[3], or blockchains for decen-

[3]https://ipfs.tech/

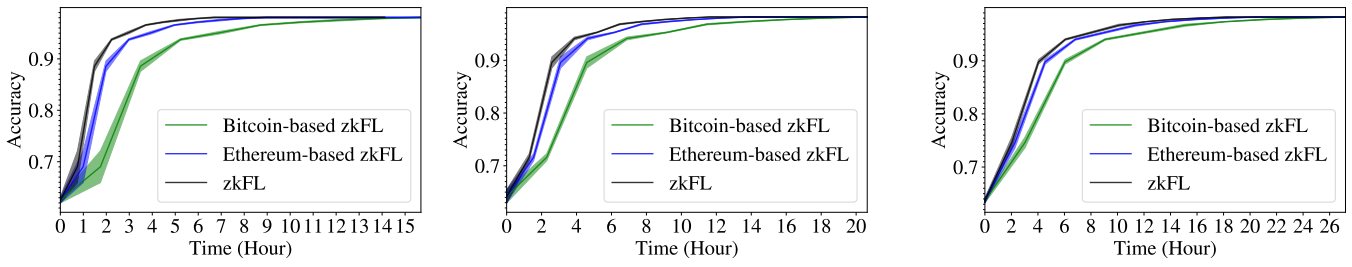
This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

This article has been accepted for publication in IEEE Transactions on Big Data. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2024.3403370



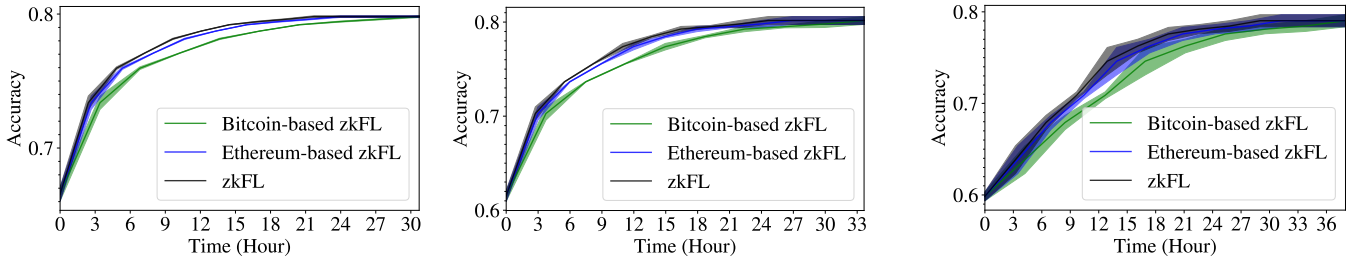
(a) Resnet18. (b) Resnet34. (c) Resnet50.

Fig. 16: Training accuracy over time for Resnet18, Resnet34, and Resnet50 with `zkFL` and blockchain-based `zkFL` systems with 4 clients.



(a) Densenet121. (b) Densenet169. (c) Densenet201.

Fig. 17: Training accuracy over time for Densenet121, Densenet169, and Densenet201 with `zkFL` and blockchain-based `zkFL` systems with 4 clients.



(a) LSTM (one layer). (b) LSTM (two layers). (c) LSTM (three layers).

Fig. 18: Training perplexity over time for LSTMs on PTB with `zkFL` and blockchain-based `zkFL` systems with 4 clients.

tralized storage, such as Filecoin [4]. These platforms enable storage of large-sized encrypted model updates, accessible to miners without the need to broadcast them repeatedly across the blockchain's P2P network with which miners connect.

Recursive Proofs for ZKPs. We have demonstrated that `zkFL` enhances the security of traditional FL at the expense of additional computation for ZKP proof generation and verification. To mitigate these computational costs, recursive zero-knowledge proofs [43] could be utilized. By employing recursion in ZKPs, complex computations can be broken down into smaller, more manageable sub-proofs. This is particularly advantageous in scenarios involving multiple layers of computation or verifying a sequence of computations, where each step can be proven individually and then combined. This approach could be beneficial in FL, where model aggregation often involves processing and verifying large, complex datasets. The application of recursive ZKPs in this context could enhance efficiency, making the overall process more manageable and scalable.

Power Consumption for Large-Scale Computing. Our results show that as the number of clients increases, both the synchronization time and aggregation time will experience an increase. This, in turn, also places a heightened computational burden on the centralized aggregator, too. It's worth noting that in practical scenarios, the centralized aggregator can be a sizable corporation (e.g., Google) equipped with the necessary resources to manage the computational costs and system power consumption efficiently. In this work, system power consumption is beyond the scope of discussion as a FL study. However, system power consumption is a non-trivial

[4]https://filecoin.io/

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

This article has been accepted for publication in IEEE Transactions on Big Data. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2024.3403370

topic for system study and shall be discussed in future work, especially in large-scale computing setups.

IX. Conclusion

We present a novel and pioneering FL approach for the era of big data, zkFL, which utilizes ZKPs to ensure a trustworthy aggregation process on the centralized aggregator. Through rigorous theoretical analysis, we establish that zkFL effectively addresses the challenge posed by a malicious aggregator during the model aggregation phase. Moreover, we extend zkFL to a blockchain-based system, significantly reducing the verification burden on the clients. The empirical analysis demonstrates that our design achieves superior levels of security and privacy compared to traditional FL systems, while maintaining a favorable training speed for the clients. These results showcase the practical feasibility and potential advantages of zkFL and its blockchain-based version in real-world applications.

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *AISTATS*, pp. 1273–1282, PMLR, 2017.

[2] B. Pejó and G. Biczók, "Quality inference in federated learning with secure aggregation," *IEEE Trans. Big Data*, 2023.

[3] W. Huang, J. Liu, T. Li, S. Ji, D. Wang, and T. Huang, "Fedcke: Cross-domain knowledge graph embedding in federated learning," *IEEE Trans. Big Data*, 2022.

[4] Z. Jiang, W. Wang, B. Li, and Q. Yang, "Towards efficient synchronous federated training: A survey on system optimization strategies," *IEEE Trans. Big Data*, vol. 9, no. 2, pp. 437–454, 2022.

[5] R. Doku, D. B. Rawat, and C. Liu, "Towards federated learning approach to determine data relevance in big data," in *IEEE IRI*, pp. 184–192, 2019.

[6] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, *et al.*, "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[7] X. Cao and N. Z. Gong, "Mpaf: Model poisoning attacks to federated learning based on fake clients," in *CVPR*, pp. 3396–3404, 2022.

[8] J. Groth, "On the size of pairing-based non-interactive arguments," in *International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 305–326, Springer, 2016.

[9] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from bitcoin," in *IEEE S&P*, pp. 459–474, 2014.

[10] H. Lycklama, L. Burkhalter, A. Viand, N. Küchler, and A. Hithnawi, "Rofl: Robustness of secure federated learning," in *44th IEEE Symposium on Security and Privacy, SP 2023, San Francisco, CA, USA, May 21-25, 2023*, pp. 453–476, IEEE, 2023.

[11] T. Liu, X. Xie, and Y. Zhang, "Zkcnn: Zero knowledge proofs for convolutional neural network predictions and accuracy," in *ACM CCS*, pp. 2968–2985, 2021.

[12] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized business review*, 2008.

[13] J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. A. Kroll, and E. W. Felten, "Sok: Research perspectives and challenges for bitcoin and cryptocurrencies," in *IEEE S&P*, pp. 104–121, 2015.

[14] G. Liu, H. Dong, Z. Yan, X. Zhou, and S. Shimizu, "B4sdc: A blockchain system for security data collection in manets," *IEEE Trans. Big Data*, vol. 8, no. 3, pp. 739–752, 2020.

[15] X. Sun, F. R. Yu, P. Zhang, Z. Sun, W. Xie, and X. Peng, "A survey on zero-knowledge proof in blockchain," *IEEE Network*, vol. 35, no. 4, pp. 198–205, 2021.

[16] Z. Wang, S. Chaliasos, K. Qin, L. Zhou, L. Gao, P. Berrang, B. Livshits, and A. Gervais, "On how zero-knowledge proof blockchain mixers improve, and worsen user privacy," in *Proceedings of the ACM Web Conference*, pp. 2022–2032, 2023.

[17] J. Weng, J. Weng, G. Tang, A. Yang, M. Li, and J.-N. Liu, "pvcnn: Privacy-preserving and verifiable convolutional neural network testing," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 2218–2233, 2023.

[18] H. Duan, L. Xiang, X. Wang, P. Chu, and C. Zhou, "A new zero knowledge argument for general circuits and its application," *IEEE Trans. Inf. Forensics Security*, 2023.

[19] Y. Zhu, Y. Wu, Z. Luo, B. C. Ooi, and X. Xiao, "Robust and secure federated learning with low-cost zero-knowledge proof," 2023.

[20] Z. Xing, Z. Zhang, M. Li, J. Liu, L. Zhu, G. Russello, and M. R. Asghar, "Zero-knowledge proof-based practical federated learning on blockchain," *arXiv preprint arXiv:2304.05590*, 2023.

[21] N. Dong, Z. Wang, J. Sun, M. Kampffmeyer, W. Knottenbelt, and E. Xing, "Defending against poisoning attacks in federated learning with blockchain," *IEEE Trans. Artif. Intell.*, 2024.

[22] H. Huang, W. Kong, S. Zhou, Z. Zheng, and S. Guo, "A survey of state-of-the-art on blockchains: Theories, modelings, and tools," *ACM Computing Surveys*, vol. 54, no. 2, pp. 1–42, 2021.

[23] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, pp. 1–32, 2014.

[24] J. Zhu, J. Cao, D. Saxena, S. Jiang, and H. Ferradi, "Blockchain-empowered federated learning: Challenges, solutions, and future directions," *ACM Computing Surveys*, vol. 55, no. 11, pp. 1–31, 2023.

[25] W. Issa, N. Moustafa, B. Turnbull, N. Sohrabi, and Z. Tari, "Blockchain-based federated learning for securing internet of things: A comprehensive survey," *ACM Computing Surveys*, vol. 55, no. 9, pp. 1–43, 2023.

[26] Y. Qu, M. P. Uddin, C. Gan, Y. Xiang, L. Gao, and J. Yearwood, "Blockchain-enabled federated learning: A survey," *ACM Computing Surveys*, vol. 55, no. 4, pp. 1–35, 2022.

[27] H. Kim, J. Park, M. Bennis, and S.-L. Kim, "Blockchained on-device federated learning," *IEEE Communications Letters*, vol. 24, no. 6, pp. 1279–1283, 2019.

[28] J. Weng, J. Weng, J. Zhang, M. Li, Y. Zhang, and W. Luo, "Deepchain: Auditable and privacy-preserving deep learning with blockchain-based incentive," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 5, pp. 2438–2455, 2019.

[29] P. Rogaway and T. Shrimpton, "Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance," in *International workshop on fast software encryption*, pp. 371–388, Springer, 2004.

[30] T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *CRYPTO*, pp. 129–140, Springer, 1991.

[31] D. F. Aranha, E. M. Bennedsen, M. Campanelli, C. Ganesh, C. Orlandi, and A. Takahashi, "Eclipse: enhanced compiling method for pedersen-committed zksnark engines," in *IACR PKC*, pp. 584–614, Springer, 2022.

[32] J. Kilian, "A note on efficient zero-knowledge proofs and arguments," in *Proceedings of ACM STOC*, pp. 723–732, 1992.

[33] O. Goldreich and Y. Oren, "Definitions and properties of zero-knowledge proof systems," *Journal of Cryptology*, vol. 7, no. 1, pp. 1–32, 1994.

[34] S. Micali, M. Rabin, and S. Vadhan, "Verifiable random functions," in *FOCS*, pp. 120–130, IEEE, 1999.

[35] N. Bitansky, "Verifiable random functions from non-interactive witness-indistinguishable proofs," *Journal of Cryptology*, vol. 33, no. 2, pp. 459–493, 2020.

[36] W. Dai, Y. Zhou, N. Dong, H. Zhang, and E. Xing, "Toward understanding the impact of staleness in distributed machine learning," in *ICLR*, 2019.

[37] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.

[38] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, pp. 770–778, 2016.

[39] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *CVPR*, pp. 4700–4708, 2017.

[40] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[41] S. Bowe, J. Grigg, and D. Hopwood, "Recursive proof composition without a trusted setup," *Cryptology ePrint Archive*, 2019.

[42] M. Carlsten, H. Kalodner, S. M. Weinberg, and A. Narayanan, "On the instability of bitcoin without the block reward," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pp. 154–167, 2016.

[43] A. Kothapalli, S. Setty, and I. Tzialla, "Nova: Recursive zero-knowledge arguments from folding schemes," in *Annual International Cryptology Conference*, pp. 359–388, Springer, 2022.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

This article has been accepted for publication in IEEE Transactions on Big Data. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2024.3403370



Zhipeng Wang is a Ph.D. student at the Department of Computing, Imperial College London, UK, supervised by Prof. William Knottenbelt. Formerly, He received the Bacholor's degree in Information and Computational Science and the Master's degree in System Engineering from Beihang University, China. His research interests span cryptography and blockchain, as well as their relevant application, privacy, and security questions.



Nanqing Dong received the Master's degree from the Department of Statistical Science, Cornell University, USA, and the Ph.D. degree in Computer Science from the University of Oxford, UK. He is currently a Young Principal Investigator and Assistant Professor at the Shanghai Artificial Intelligence Laboratory. He visited the Machine Learning Department, Carnegie Mellon University, USA. His research interests include machine learning, computer vision, optimization, and AI for science.



Jiahao Sun received the Master's degree in Computer Science from the University of Oxford, UK. He is currently serving as an Honorary Research Fellow at the Data Science Institute, Imperial College London, UK. He is the Founder and CEO of FLock.io, an innovative company operating at the intersection of AI and Blockchain. He was the Head of AI at the RBC Wealth Management. His work focuses on the application of artificial intelligence in Fintech.



William Knottenbelt received the BSc degree in Computer Science and Statistics, BSc (Hons) degree in Computer Science, and MSc degree in Computer Science from the Department of Computer Science, University of Cape Town, South Africa, and the Ph.D. degree from the Department of Computing, Imperial College London, UK. He is a Professor of Applied Quantitative Analysis in the Department of Computing, Imperial College London and the Director of the Imperial College Centre for Cryptocurrency Research and Engineering. His broad area of research interest is the application of mathematical modeling techniques to real-life systems. His research interests also encompass all aspects of cryptocurrencies, blockchains, distributed ledgers, and smart contracts.



Yike Guo (Fellow, IEEE) received the degree (Hons.) in Computing Science from Tsinghua University, China, in 1985, and the Ph.D. degree in computational logic from the Imperial College London, U.K., in 1994. He is the Provost of the Hong Kong University of Science and Technology since December 2022 and concurrently a Chair Professor in the Department of Computer Science and Engineering. He is also a Professor of Computing Science in the Department of Computing at Imperial College London. He is the founding Director of the Data Science Institute at Imperial College. His main research interests lie in the field of machine learning and large-scale data analysis management. He is a Fellow of the Institute of Electrical and Electronics Engineers (FIEEE), Fellow of the Royal Academy of Engineering (FREng), Member of Academia Europaea (MAE), Fellow of British Computer Society (FBCS), and a Trustee of the Royal Institution of Great Britain. Professor Guo has published over 250 articles, papers and reports.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/