Cluster Computing (2025) 28:942
https://doi.org/10.1007/s10586-025-05668-4

I Trained That! Client-Side Proof of Participation in Federated Learning

Carlo Mazzocca[1] · Alessio Mora[2] · Nicolò Romandini[2] · Rebecca Montanari[2] · Paolo Bellavista[2]

Received: 29 April 2025 / Revised: 8 June 2025 / Accepted: 23 July 2025 / Published online: 9 October 2025

Abstract
In Federated Learning (FL) clients collaboratively train a Machine Learning (ML) model by sharing updates computed over their private data. These updates are aggregated to form a global model, which requires clients to trust the central server. However, clients have no direct means to verify whether their updates were incorporated into the global model. This lack of transparency raises challenges that may discourage participation in the federation. For instance, a malicious server might exclude legitimate clients to deny them rewards or recognition. Even in benign scenarios, updates may be disregarded due to network failures or predefined aggregation conditions (e.g., a quorum). This highlights the need for mechanisms that let clients independently verify their inclusion. To this specific purpose, we propose a novel approach called Membership Proof in Federated Learning (MPFL), which enables client-side verifiability of participation. In MPFL, model updates are aggregated via a smart contract, which also generates a unique cryptographic proof of participation for each update by using cryptographic accumulators. By leveraging blockchain and smart contracts, our approach enhances system trustworthiness, while cryptographic proofs provide an efficient and privacy-preserving method for clients to verify inclusion. In addition, the paper reports on how we have implemented MPFL and extensively evaluated it across three diverse datasets and ML architectures, thus demonstrating its effectiveness and practical viability.

Keywords Federated Learning · Privacy-Preserving Machine Learning · Decentralized learning · Privacy

1 Introduction

Privacy-preserving distributed learning paradigms are gaining significant traction thanks to their ability to collaboratively train models without exposing raw data. This shift is particularly relevant in today's landscape, where the rapid digitization of society and the proliferation of connected devices, such as those in the Internet of Things (IoT), are fueling unprecedented volumes of data generation. These vast datasets are foundational for training effective Machine Learning (ML) models [1]. However, conventional ML approaches typically rely on data centralization, requiring sensitive information to be transferred to centralized servers. This practice introduces major privacy risks, including the potential for data leaks or misuse.

To mitigate such concerns, Federated Learning (FL) has emerged as a compelling alternative. FL enables multiple clients to collaboratively train a shared ML model without exposing their local data to any external entity [2]. In this decentralized setup, each client performs local training on its private dataset and transmits only the resulting model updates (e.g., gradients or weights) to a central aggregator, which constructs the global model.

In this setting, clients may seek to verify that their contributions were indeed included in the aggregation process. Membership verification mechanisms serve multiple purposes: (i) they promote fairness, ensuring that all participating clients are treated equally and that no updates

✉ Carlo Mazzocca
cmazzocca@unisa.it

Alessio Mora
alessio.mora@unibo.it

Nicolò Romandini
nicolo.romandini@unibo.it

Rebecca Montanari
rebecca.montanari@unibo.it

Paolo Bellavista
paolo.bellavista@unibo.it

[1] Department of Information and Electrical Engineering and Applied Mathematics, University of Salerno, Fisciano, Italy

[2] Department of Computer Science and Engineering, University of Bologna, Bologna, Italy

are arbitrarily excluded from the global model; (ii) they play a key role in incentive mechanisms, where clients are rewarded for their participation–verifying inclusion allows clients to confidently claim rewards based on actual contribution [3, 4]; (iii) even in the absence of incentives, such mechanisms support auditing and regulatory compliance, especially in sensitive domains where transparency and trust are critical. By enabling inclusion checks, clients can detect misbehavior (e.g., a server selectively ignoring updates), verify correct execution of the protocol, and build greater confidence in the fairness and integrity of the FL process.

As a result, FL frameworks should support efficient and privacy-preserving membership proof mechanisms. A membership proof provides verifiable evidence that a client update was used in the global model. It is worth noting that this must be achieved without undermining one of the key privacy guarantees of FL: the system must protect the content of local data and avoid the need to identify clients explicitly [5]. While several works have focused on verifying the correctness of global models [6–8], they often overlook the ability of individual clients to confirm their participation.

To address this gap, we build on our prior work and introduce the first protocol for efficient Membership Proof in FL (MPFL), originally presented in [9]. Our method empowers clients with the possibility to verify that their model update contributed to the global model, with minimal computational and communication overhead, and without compromising privacy. At the core of our approach is a cryptographic accumulator based on Elliptic Curve Cryptography (ECC) [10], which compactly encodes multiple contributions into a constant-size digest. This accumulator value allows the generation of concise proofs, known as witnesses, which clients can use to verify inclusion.

To guarantee the integrity and trustworthiness of the global model and its related cryptographic artifacts, MPFL replaces the traditional centralized parameter server of FL with a decentralized smart contract deployed on a blockchain, which coordinates and verifies the aggregation of model updates. This decentralized architecture mitigates common FL issues such as single points of failure, scalability bottlenecks, and potential tampering by untrusted servers [11, 12]. The blockchain serves as an immutable ledger for managing aggregations and issuing proofs, improving the overall trust and auditability of the training process. Although the individual components of MPFL (i.e., cryptographic accumulators, blockchain, and smart contracts) are known, their integration in MPFL addresses a critical and previously underexplored need in FL: enabling the transparent generation of participation proofs that are uniquely bound to each client's contribution, without compromising efficiency or privacy.

We have implemented and evaluated our MPFL solution by using various datasets and three ML models of different sizes, with varying numbers of participating clients. Our experiments demonstrate that the overhead introduced by the accumulator and membership proof creation is minimal. On the client side, only 96 bytes of local storage are needed to store the membership proof and the corresponding accumulator value. The verification process is lightweight, completing in approximately 20 milliseconds, making MPFL a practical and scalable solution for FL scenarios. The main contributions of the paper can be summarized as follows:

- We present a comprehensive analysis highlighting the importance of verifiable proof of participation as a critical challenge in FL;
- We introduce MPFL, the first client-side protocol designed to verify whether individual contributions are incorporated into the global model;
- We implement and rigorously evaluate MPFL across a range of datasets and learning scenarios, demonstrating its scalability and practical efficiency. We also open-sourced our code to the research community[1].

The remainder of this paper is organized as follows. Section 2 introduces the needed background, while Section 3 highlights the relevance of enabling clients to verify participation in FL. Section 4 presents the reference threat model and Section 5 describes our original approach and protocol, whose security analysis is discussed in Section 6. Section 7 extensively evaluates our solution, while Section 8 review related work in the field. Finally, Section 9 draws our conclusions.

2 Background

To support the design of our proposed inclusion verification mechanism in FL, we rely on three foundational technologies: blockchain, cryptographic accumulators, and FL itself. This section provides essential background on these building blocks. We first introduce blockchain and smart contracts as a decentralized infrastructure. We then present cryptographic accumulators as a tool for set membership proofs, and finally, we summarize key aspects of FL relevant to our solution.

2.1 Blockchain

Blockchain technology offers a secure and decentralized platform to share and process data in a network of unknown

[1] https://github.com/MMw-Unibo/MPFL

entities. Its data structure is a continuously expanding chain of blocks containing transactions [13]. Each block is cryptographically linked to the previous one through hashes. Using hashes guarantees immutability since any attempt to alter a block would produce a different hash, disrupting the entire chain. Additionally, blockchain operates on a peer-to-peer paradigm, being resilient against single points of failure. In this decentralized ecosystem, each node maintains a consistent copy of the ledger synchronized through consensus protocols such as Proof of Work (PoW) or Proof of Stake (PoS). The absence of a central authority that controls the entire network underscores blockchain's democratized nature. These features make it the natural solution for many applications such as FL, where numerous unknown participants collaborate on shared tasks without relying on any trusted central intermediary.

Furthermore, integrating smart contracts paves the way for novel applications by enabling programmable, self-executing agreements directly on the blockchain. This makes smart contracts trusted by all involved participants as the transparent nature of blockchain ensures that all transactions and program details are visible and immutable. For example, in the context of FL, smart contracts are exploited to securely aggregate client updates.

2.2 Cryptographic accumulators

Cryptographic accumulators aggregate many different data into a fixed-length digest called accumulator value v, enabling efficient set membership verification. For each element e_i, a witness p_i is derived by aggregating all the values except e_i. Accumulators are categorized according to their support for different types of witnesses. Specifically, those enabling membership witnesses are referred to as *positive*, while those that support non-membership witnesses are known as *negative*. Finally, accumulators capable of supporting both functionalities are called *universal*.

Over the years, several types of accumulators have been proposed. Among them, ECC-based accumulators have shown superior performance compared to alternatives like RSA-based accumulators and Merkle Trees (MTs) [14, 15]. ECC-based accumulators offer smaller membership proofs and faster verification due to the reduced number of cryptographic operations. In contrast, RSA-based accumulators require large modulus sizes to maintain equivalent security levels, resulting in larger proofs and slower computations. MTs have logarithmic proof sizes that grow with the size of the dataset, leading to scalability limitations in FL settings involving many clients.

In this work, we adopt a positive ECC-based accumulator [10] that supports batch operations, including efficient witness generation. This design aligns well with the heterogeneity of FL clients, enabling lightweight, scalable, and verifiable inclusion proofs of client model updates into the global model.

2.3 Federated learning

FL enables the training of deep learning models without requiring raw data to be centralized. Instead, a federation of nodes, such as edge devices or IoT devices, collaborates to train a shared model that generalizes across the entire data distribution. Unlike traditional cloud-centric training, FL shifts the training process to the devices where the data resides. Only processed information (e.g., model updates) is transmitted, preserving data privacy and reducing communication overhead.

The de facto standard algorithm in FL is Federated Averaging (FedAvg) [16], which operates in a star-shaped topology [2] consisting of multiple clients (devices) and a central server (orchestrator). FedAvg begins by initializing a global model, either randomly or from a pre-trained checkpoint, and structures the training into communication rounds. In each round, a subset of clients is selected to participate. These clients compute local weight updates using their private data and send the results back to the server. The server then aggregates the updates and apply them to refine the global model. FedAvg uses a weighted average based on the number of data points per client to aggregate model updates. The next round then begins using the updated global model. This iterative process continues until a predefined stopping criterion is met, such as reaching a maximum number of rounds or achieving a target metric, like accuracy on a validation or test dataset, if available.

In this context, membership proof mechanisms can serve several purposes. From a participant's perspective, being able to verify inclusion in the training process (i.e., "Was my contribution actually used?") is crucial for transparency and accountability. This is especially important in incentive-driven settings, where clients may be rewarded based on their participation [17]. It also helps detect potential misbehavior by the server, such as ignoring certain clients' updates. Moreover, verifying membership can support auditing and compliance in sensitive applications where trust is essential. The next Section further describes the reasons that motivate the investigation of strategies to verify the client inclusion in FL processes.

3 Motivation

Ensuring that clients' contributions are genuinely included in the generation of the global model is a foundational principle in FL. A mechanism to verify inclusion is not only

a technical enhancement but a necessary component to support trust, fairness, and accountability in collaborative model training. Clients may seek to verify their inclusion for several interconnected reasons, which we detail below.

1. **Fairness and Transparency.** A core motivation lies in promoting fairness: all participating clients deserve to be treated equally, with their updates considered in the aggregation process. Since many FL systems rely on partial participation schemes to improve communication efficiency, clients are not always selected in each round. While this is often expected, the lack of transparency in selection may lead to suspicions of unfair treatment or exclusion. Verifiable inclusion ensures that no client is arbitrarily excluded, intentionally or otherwise. This transparency fosters trust in the FL process, particularly important in collaborative or cross-silo settings, where participants may not fully trust the coordinating server or aggregator. Moreover, clients benefit from understanding how their data influences the global model. This is particularly relevant in applications such as personalized recommendations, where users expect the model to reflect their individual data. The ability to verify inclusion gives clients assurance that their participation has a tangible impact.
2. **Incentive Mechanisms.** Participation in FL is often driven by incentives–clients may be compensated through monetary rewards, improved local performance, or access to better global models. In such settings, verifying inclusion becomes a prerequisite for fair compensation. Clients must be able to confirm that their updates have been incorporated into the global model before claiming rewards. Without a mechanism for verifiable inclusion, the incentive structure becomes vulnerable to disputes and abuse. Recent works explicitly leverage inclusion proofs to enable accountable reward distribution in decentralized FL environments [3, 4].
3. **Auditing, Accountability, and Anomaly Detection.** Even in the absence of explicit incentives, verifying inclusion serves as a powerful auditing tool. In regulated or sensitive domains (e.g., healthcare and finance) verifiable records of participation support compliance with legal and ethical requirements. Additionally, clients can use inclusion verification to detect anomalies in the training process. For instance, if a client's contribution is missing from the global model, it may indicate communication failures, adversarial filtering, or rejection due to low-quality updates. The ability to confirm or contest inclusion enables proactive debugging and enhances system robustness.
4. **Emerging Paradigms: Federated Unlearning.** Beyond traditional FL, verifiability of contribution inclusion (or exclusion) is also critical in newer paradigms like Federated Unlearning (FU) [18]. FU aims to remove a client's influence from a model upon request, often for privacy or compliance reasons (e.g., GDPR's right to be forgotten). In this context, clients may require proof that their data *is no longer* present in the model. While this paper focuses on inclusion verification, not unlearning, we note that the MPFL mechanism introduced here lays the foundation for future extensions to verify exclusion as well.

4 Threat model

We consider an FL setting involving a set of clients $\mathcal{C} = \{c_1, c_2, \ldots, c_n\}$, each holding a private local dataset D_c that remains on-device. The goal is to collaboratively train a global model g without sharing raw data. In each training round t, a client $c \in \mathcal{C}$ trains a local model m_c^t on its dataset D_c, and submits the resulting model update to a central aggregator. The aggregator computes the global model using the FedAvg protocol:

$$g^{t+1} \leftarrow \sum_{c \in \mathcal{C}} \frac{|D_c|}{\sum_{c' \in \mathcal{C}} |D_{c'}|} \cdot m_c^t, \tag{1}$$

where $|D_c|$ denotes the size of the local dataset held by client c. In this context, it becomes crucial for each participating client $c \in \mathcal{C}$ to be able to efficiently verify whether its contribution m_c^t has been correctly included in the aggregated global model.

In our threat model, we focus on attacks that interfere with the ability of clients to verify the inclusion of their contributions in the global model. We do not consider attacks that manipulate the global model itself, such as poisoning attacks, or those aimed at compromising client privacy. Instead, we concentrate on adversarial behaviors that exclude clients' contributions or impersonate other clients, making it difficult for legitimate clients to verify their actual participation. The primary adversaries in our model are: (i) a malicious aggregator, capable of manipulating both the aggregation process and the mechanisms used by clients to verify inclusion, and (ii) a malicious client, who impersonates other clients, further complicating the verification of legitimate contributions.

- **Intentional Exclusion**: A malicious aggregator may intentionally exclude a legitimate client's model update from being included in the global model. This exclusion could be used to prevent the client from receiving rewards or recognition for their contribution.

- **Forgery of Inclusion Proofs**: A malicious aggregator may forge or manipulate the proof of inclusion provided to the client, leading the client to believe their contribution has been included in the global model when it has not.
- **Model Update Forgery**: A malicious client may forge model updates and falsely attribute them to a legitimate client. This attack creates the false appearance that the legitimate client contributed, while in reality they did not. Such forgery complicates the verification process, as it may prevent the legitimate client from immediately recognizing whether their genuine contribution has been correctly included.

5 Membership proof in Federated Learning

This section describes MPFL, our novel method that enables efficient membership proof verification in FL by leveraging blockchain, smart contracts, and ECC-based accumulators. Specifically, the aggregator is implemented through a smart contract that also produces a proof of inclusion, i.e., a witness, for each client update m_c used during the generation of the global model. The use of blockchain ensures the transparency of the entire aggregation process, allowing all participants to independently verify the inclusion of their contributions, without relying on external parties. In the following, we first introduce the preliminaries to understand our proposal. Then, we describe how MPFL operates. Figure 1 offers a visual overview of MPFL.

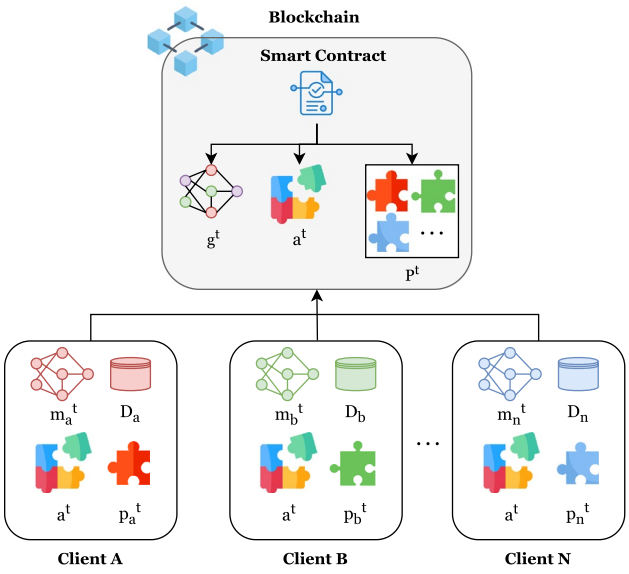

Fig. 1 Overview of MPFL

5.1 Preliminaries

This subsection provides a detailed explanation of how the accumulator and witnesses are utilized in MPFL to prove a client's inclusion in g^t. The concept of membership proof is closely linked to the notion of a witness, which is formally defined as follows:

Definition 1 (**Membership Proof**) In federated learning, a *membership proof* refers to the ability of a client to efficiently verify that their contribution, specifically their local model update, has been successfully included in the global model for a given training round t.

Inputs of the Accumulator. The values included in a^t are derived from the hash conversion of $m_c^t \in M^t$. The security of a^t is thus dependent on the chosen hash function. In our implementation, we utilize the SHA-256 hash function, known for its security and collision resistance.

Functions. Below, we present the key functions employed by MPFL to facilitate membership proof in federated learning:

- $m_c^t \leftarrow \text{ComputeLocalModel}(\mathsf{D_c}, \mathsf{g^{t-1}})$: Each client c trains an updated local model m_c^t based on their private data $\mathsf{D_c}$ and the global model from the previous round $\mathsf{g^{t-1}}$.
- $g^t \leftarrow \text{ComputeGlobalModel}(\mathsf{M^t})$: The aggregator computes the new global model g^t by averaging all local model updates $m_c^t \in M^t$, where M^t is the set of all updates from the clients in round t.
- $a_{info} \leftarrow \text{Setup}(\text{K})$: This function sets up the parameters for the accumulator a_j, using a set of public values K, which represent the random generators of the elliptic curve used for the accumulator.
- $a^t \leftarrow \text{ComputeAccumulator}(\mathsf{a_{info}}, \mathsf{M_c^t})$: This function accumulates the collected model updates M_c^t and computes the corresponding point on the elliptic curve, resulting in the accumulator a^t.
- $p_c^t \leftarrow \text{ComputeMembershipProof}(\mathsf{a^t}, \mathsf{m_c^t})$: This function generates the membership proof p_c^t for the model update m_c^t, providing evidence that the contribution has been included in the accumulator a^t.
- $0, 1 \leftarrow \text{Verify}(\mathsf{a^t}, \mathsf{m_c^t}, \mathsf{p_c^t})$: Each client c verifies whether the membership proof p_c^t associated with their model update m_c^t belongs to the accumulator a^t. If the proof is valid, the result is 1 (true), otherwise 0 (false).

5.2 Protocol

In each round t of FL, the participating clients $c_1, c_2, \ldots, c_n$ collaboratively train by computing their local model updates

m_c^t using their respective local datasets D_c. This is achieved through the ComputeLocalModel() function, which also takes the previous global model g^{t-1} as input. The resulting local updates are then sent to the smart contract, which orchestrates the construction of the new global model g^t.

Once the aggregation conditions are met, the smart contract invokes the ComputeGlobalModel() function, using the set of received updates M^t and the previous model g^{t-1} as input. To guarantee the integrity of client contributions and produce cryptographic membership proofs, this process also involves executing the Setup(), ComputeAccumulator(), and ComputeMembershipProof() functions.

Data: Model updates M^t
Result: Global model g^t and membership proofs P^t
Function `ComputeGlobalModel(`M^t`):`
  $K \leftarrow$ random generator elliptic curve;
  $a_{info} \leftarrow$ Setup(K)
  $a^t \leftarrow$ ComputeAccumulator(a_{info}, M^t)
  for $m_c^t \in M^t$ **do**
    $p_c^t \leftarrow$ ComputeMembershipProof(a^t, m_c^t);
    push p_c^t into P^t
  end
  $N \leftarrow$ size of M^t
  $g^t \leftarrow \frac{\sum_{c=1}^{N} |D_c| \cdot m_c^t}{N}$
  return g^t, P^t;

Algorithm 1 Smart contract that generates global model and membership proofs

The ComputeMembershipProof() procedure plays a crucial role in allowing clients to verify the inclusion of their updates in the global model. When a model update $m_c^t \in M^t$ is incorporated into g^t, it is first hashed and then accumulated into a^t using the ComputeAccumulator() function. After computing a^t, the smart contract invokes ComputeMembershipProof() for each m_c^t, generating the corresponding membership proof p_c^t. These proofs, along with the newly computed global model g^t, are subsequently distributed to the respective clients.

To verify whether their update m_c^t was included in g^t, a client c runs the Verify() function, as illustrated in Figure 2. This function takes m_c^t and the associated proof p_c^t as input. If the result of the verification matches the accumulator a^t, the client can confirm that their update was correctly incorporated into the global model. The pseudocode for ComputeGlobalModel() is provided in Algorithm 1.

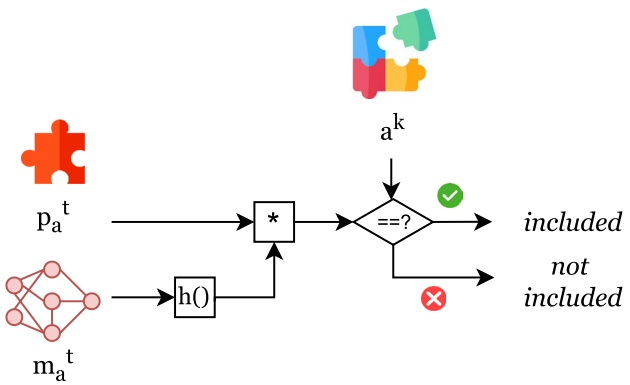

Fig. 2 Verification of inclusion

5.3 Discussion

Leveraging blockchain and smart contracts enhances the trustworthiness of the FL process by enabling transparent and auditable model aggregation. In MPFL, the global model is generated directly on-chain, allowing all participants to independently verify the correctness of the aggregation logic detect potential flaws or biases encoded in the smart contract. However, such transparency does not guarantee that a specific client's update has been included in the global model. Several factors may prevent a client's update from being incorporated, such as network connectivity issues or conditions embedded in the smart contract logic. For instance, generating the global model only after receiving a certain number of updates. In such cases, verifying inclusion becomes difficult. Although clients could attempt to analyze blockchain transactions to confirm their participation, this is impractical due to the volume and structure of on-chain data. MPFL addresses this challenge through ECC accumulators that encode all valid contributions into a constant-size digest published on-chain. The smart contract also generates lightweight cryptographic witnesses to prove client inclusion.

This design achieves an efficient and privacy-preserving membership proof mechanism that scales across heterogeneous clients and avoids the overhead of exhaustive on-chain analysis. While the individual components of MPFL are established tools, their integration in our framework fills a critical and previously underexplored gap in FL by enabling verifiable participation, thereby enhancing both transparency and accountability in decentralized learning environments. Moreover, MPFL can be readily extended to support quality-weighted contribution proofs, enabling more nuanced verification of each client's impact on the learning process. This enhancement could be realized by integrating smart contracts with contribution valuation techniques, such as Shapley value approximations [19] or influence functions [20], which estimate the utility of individual updates. These quantified values can then be securely bound to the corresponding membership proofs, allowing clients not only to verify their inclusion but also to assess the relative significance of their contributions in the final model.

Finally, it is important to note that MPFL is not inherently limited to gradient-sharing federated learning schemes such as FedAvg. The core requirement for applying MPFL is a star-shaped communication topology, where a central coordinator collects client contributions. MPFL replaces this central entity with a blockchain-based smart contract,

which transparently manages, aggregates, and verifies the contributions. These architectural conditions are met by a wide range of federated and distributed ML paradigms. As a result, MPFL can be naturally extended to other settings, such as Split Learning [21] or FL variants that exchange model outputs or intermediate representations instead of raw weights (e.g., [22, 23]). This flexibility highlights MPFL's broader applicability in ensuring transparent and verifiable participation across diverse collaborative learning scenarios. In Split Learning, clients process the initial layers of a model locally and send the resulting activations to a central server, which completes the forward and backward passes. In FL via model outputs, clients share predictions or intermediate representations instead of parameter updates, often to preserve privacy or reduce the computation overhead associated with local training. Despite differences in the data exchanged, both paradigms meet the structural requirements necessary for MPFL.

6 Security analysis

MPFL builds upon cryptographic accumulators and smart contract logic to ensure the verifiability and integrity of model updates in FL. In this section, we describe how MPFL addresses the threats identified in Section 4.

6.1 Intentional exclusion

In MPFL, the global model g^t is computed by a smart contract based on the set of updates $M^t = \{m^t_{c_1}, \ldots, m^t_{c_n}\}$ collected on-chain. This mitigates the risk of intentional exclusion by a malicious central server, as the smart contract logic is deterministic and publicly verifiable. Each model update m^t_c is hashed and included in the accumulator $a^t = \text{ComputeAccumulator}(a_{\text{info}}, M^t)$. A client's update is said to be *verifiably included* if there exists a witness $p^t_c = \text{ComputeMembershipProof}(a^t, m^t_c)$ such that:

$$\text{Verify}(a^t, m^t_c, p^t_c) = 1$$

Hence, exclusion can only occur if $m^t_c \notin M^t$, which implies the update was never received by the smart contract–either due to submission failure or unmet aggregation conditions. This limitation also applies in centralized deployments, where the server is assumed to be honest but still may not be able to include an update if it fails to meet the necessary conditions or encounters technical issues.

6.2 Forgery of inclusion proofs

In MPFL, the process of aggregating clients' updates into the global model is accompanied by the generation of proof of inclusion p^t_c. When a client's contribution is successfully included in the global model, the smart contract not only updates the model but also generates a corresponding membership proof. Therefore, the generation of a membership proof p^t_c is tightly bound to both m^t_c and a^t, and is performed on-chain by the smart contract. We formally define the security guarantee as follows:

Definition 2 (Unforgeability of Membership Proofs) A membership proof scheme is unforgeable if no probabilistic polynomial-time adversary $\mathcal{A}$, given only a^t, can generate a tuple (m^*, p^*) such that $m^* \notin M^t$ and:

$$\text{Verify}(a^t, m^*, p^*) = 1$$

except with negligible probability in the security parameter λ.

This proof, or witness, enables the client to directly and efficiently verify their inclusion in the updated global model. Because proof generation is executed within the smart contract, tampering with p^t_c would require manipulating the blockchain state, which is computationally infeasible under standard consensus assumptions.

6.3 Model update forgery

In MPFL, clients verify their inclusion in the global model by combining their original model update with the corresponding membership proof. This verification step is crucial: even if a malicious client forges an update in the name of another, the legitimate client can detect the fraud by verifying that no proof of inclusion matches their genuine contribution.

A malicious client may attempt to forge an update $\hat{m}^t_c \neq m^t_c$ attributed to another legitimate client c, aiming to deceive the system into accepting it as valid. However, the design of MPFL ensures that such attempts can be detected and rejected. Because membership proofs p^t_c are computed by the smart contract and are cryptographically linked to the submitted m^t_c, the legitimate client can locally verify their own contribution by checking:

$$\text{Verify}(a^t, m^t_c, p^t_c) = 1$$

If an adversary replaces m^t_c with $\hat{m}^t_c$, the verification will fail unless the adversary can forge a valid proof $\hat{p}^t_c$, which contradicts the unforgeability property. Thus, impersonation attempts are readily detectable.

7 Performance evaluation

To evaluate the feasibility and performance of our membership proof protocol, we implemented it through a smart contract [24] written in NodeJS that implements FedAvg, as well as the generation of membership proofs for all clients whose contributions are included in the global model. We used Hyperledger Fabric as the underlying blockchain framework for our deployment. The network consisted of six organizations, each hosting a single peer node, and one ordering node leveraging the Raft consensus protocol. The endorsement policy of the smart contract was based on a majority rule, requiring approval from at least four of the six peer nodes for a transaction to be validated. The network was hosted on a machine with an Intel(R) Core(TM) i5-3470 CPU at 3.20 GHz and 32 GB of RAM.

7.1 In-the-field experimentation

We have extensively evaluated the performance and the overhead introduced by MPFL via a series of experiments, with varying degrees of complexity regarding the ML models utilized, the datasets employed, and the number of clients involved. Specifically, our experiments aim to evaluate the overhead introduced by MPFL when generating membership proofs during the global model generation.

Initially, we used the Fashion-MNIST dataset [25], which consists of Zalando item images, comprising a training set of 60,000 examples and a test set of 10,000. Each example is represented by a 28x28 grayscale image associated with one of 10 classes. The neural network architecture used in this experiment features three layers: a Flatten input layer, a Dense layer with 128 neurons, and an output layer with 10 neurons, corresponding to the 10 classes. The network comprises a total of 101,770 parameters. We then increased the complexity by employing the CIFAR-10 dataset [26] together with more sophisticated neural network architectures. CIFAR-10 is a widely recognized dataset comprising 60,000 color images of size 32 by 32, distributed across ten classes with 6,000 images per class. The dataset is divided into 50,000 training images and 10,000 test images. In this phase, we first used the MobileNetV2 architecture proposed by Sandler et al. [27], which contains 2,270,794 parameters, representing approximately twenty times the size of the model used in the initial experiment. In addition, we evaluated the EfficientNetV2 [28] architecture, a more recent and high-performing model designed to balance accuracy and computational efficiency. EfficientNetV2 contains 5,932,122 parameters, nearly two times the number found in MobileNetV2, allowing us to further stress-test MPFL in scenarios involving large-scale and high-capacity models.

In all experiments, we run 10 FL rounds, with each client locally training the neural network for 5 epochs. The datasets were equally partitioned among all FL clients to ensure a fair distribution of computational load. Since the generation of membership proofs depends on the number of FL participants, we conducted experiments varying the number of clients from 10 to 1000. Each experiment was executed 10 times, and the outcomes were aggregated.

7.2 Results

Figures 3, 4, and 5 show the overhead introduced by MPFL across the datasets and ML models under study, as well as when varying the number of involved clients. Specifically, we report the latency, expressed in *ms*, incurred in computing the accumulator value and generating membership proofs for clients to verify their inclusion in the global models. Additionally, we examine the amount of time required for hashing model updates. This operation is necessary because the functions that accumulate client contributions and generate corresponding membership proofs require hashes as input. To effectively evaluate the impact of MPFL on federated learning aggregation, we also measure the latency of the standard FedAvg algorithm, labeled as `FedAvg` in the figures. In addition, we assess the overhead introduced by blockchain-related operations, such as the consensus protocol, which is labeled as `Consensus`. As shown in the

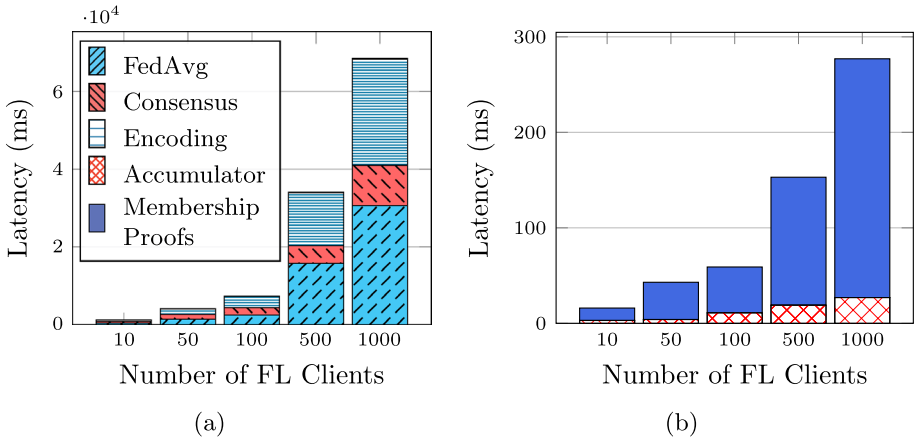

Fig. 3 Latency analysis with Fashion-MNIST and 1-layer neural network, varying the number of FL clients. (**a**) Shows total latency for global model generation, encoding, accumulator, and membership proof creation. (**b**) Zooms in on accumulator and membership proof latency

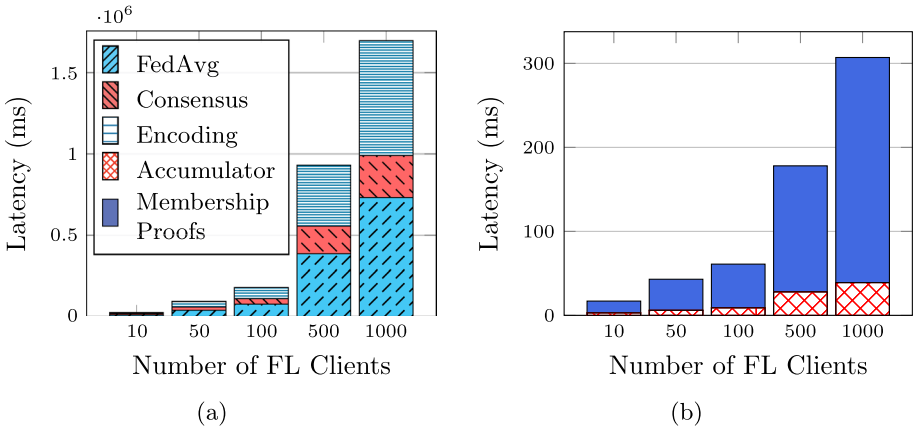

Fig. 4 Latency analysis with CIFAR-10 and MobileNetV2, varying the number of FL clients. (**a**) Shows total latency for global model generation, encoding, accumulator, and membership proof creation. (**b**) Zooms in on accumulator and membership proof latency

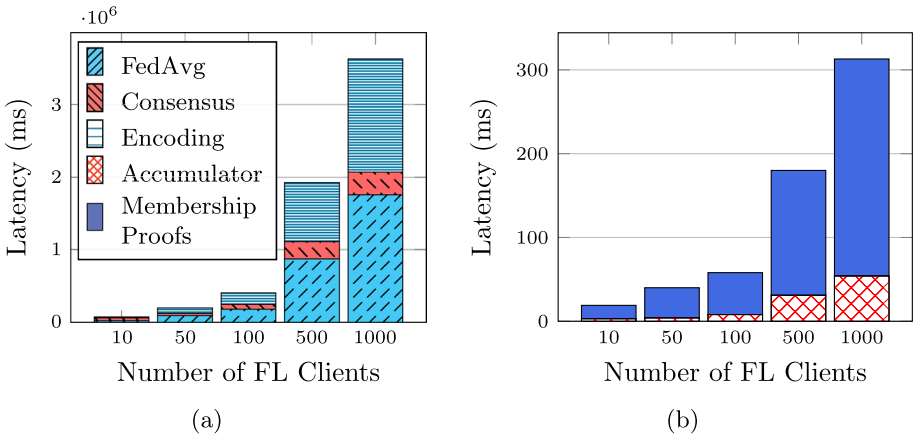

Fig. 5 Latency analysis with CIFAR-10 and EfficientNetV2, varying the number of FL clients. (**a**) Shows total latency for global model generation, encoding, accumulator, and membership proof creation. (**b**) Zooms in on accumulator and membership proof latency

figures, in most cases, the FedAvg aggregation algorithm is the most time-consuming operation in the process. Blockchain-related operations introduce only minimal overhead, demonstrating that the decentralized infrastructure can be integrated efficiently. Notably, generating hashes for each model is the second most time-intensive task following FedAvg. The time required to process MobileNetV2 and EfficientNetV2 is an order of magnitude greater than that for a 1-layer neural network. Interestingly, this pattern appears to correlate with the number of weights in the models. Specifically, as mentioned, MobileNetV2 and EfficientNetV2 have an order of magnitude more weights than the 1-layer neural network. Moreover, EfficientNetV2 has roughly double the number of weights compared to MobileNetV2, and it exhibits approximately double the latency.

Instead, the time required to generate the accumulator and membership proofs is negligible compared to the other operations across all three experiments. This is why subfigures (b) are used to display the latency separately. These figures show that as the number of participating clients increases, latency increases only marginally. Moreover, with the same number of clients, latency remains almost unchanged across the three model configurations. This is expected, as these operations rely on hashes and are not dependent on the model's size. Furthermore, it is worth noting that the latency associated with membership proof scales with the number of proofs required: generating p membership proofs requires at least $\Omega(p)$ operations [29]. In general, with a higher number of clients, there is an expected increase in the number of model updates. Consequently, the figures outline a remarkable uptick in the time required to convert the model updates and generate the corresponding hashes. Although MPFL introduces latency overhead in generating the global model, it is worth noting that this time increase is acceptable since clients' contributions are not continuously aggregated; thus, overhead in the order of minutes or a few hours does not constitute a serious concern. For example, in a scenario where participants are smartphones, the aggregation of clients' updates may occur once a day according to a specified timeline (e.g., during the night) [30]. The experiments highlight that the security, transparency, and decentralization benefits provided by MPFL and blockchain are achieved with limited impact on overall system performance, making the trade-offs acceptable within the context of typical FL scenarios.

Client-side. MPFL imposes minimal overhead on the client side. Local training proceeds exactly as in traditional FL settings, with no modifications required to the training process or the way updates are shared. The only overhead introduced relates to storing a compact cryptographic proof

and verifying whether a model update has been included or excluded from the global model. Each client needs to store just 96 bytes to maintain the accumulator value and the corresponding membership proof. Furthermore, the verification latency remains consistently low, around 20 milliseconds. It is worth noting that, due to ue to the unique features of the ECC-based accumulator, both storage requirements and verification time remain unaffected and do not grow with the number of updates or the complexity of the underlying model. This makes MPFL particularly well-suited for FL deployments, which are characterized by heterogeneous participants, who may rely on resource-constrained devices.

8 Related work

Recent efforts to address the verifiability of global model updates in FL have increasingly turned to blockchain and smart contract technologies. These decentralized architectures aim to reduce reliance on centralized servers while offering transparency and auditability. For instance, Kalapaaking et al. [7] propose a dual-blockchain infrastructure where local models, computed within trusted execution environments, are aggregated by blockchain nodes. Once a consensus is reached, based on the alignment of global model hashes, the result is broadcast to a second blockchain. This secondary layer finalizes the update using a multi-signature protocol before committing it. Although this setup offers strong consistency guarantees, it incurs significant overhead due to the dual-chain structure and still does not offer a way for individual clients to verify whether their own contributions were actually included in the final global model.

Similarly, VFChain [8] proposes a secure and verifiable blockchain-based framework for FL where smart contracts manage aggregation once a threshold θ of client updates is collected. Each global model update is signed and recorded on-chain, and an efficient indexing structure is introduced for streamlined data access. While VFChain ensures the integrity and authenticity of the overall training process, it overlooks client-side verifiability. In particular, it does not enable individual clients to directly verify whether their updates were included in the final global model, nor whether their contributions satisfied the threshold-based selection criteria enforced by the smart contract.

In contrast to blockchain-based systems, earlier efforts such as VerifyNet [6] explore verifiability through cryptographic primitives within a centralized architecture. In VerifyNet, clients encrypt their model updates and transmit them to a central server, which performs aggregation and returns the global model accompanied by a cryptographic proof of correctness. This proof is constructed using homomorphic hashing and pseudorandom functions, ensuring that the aggregation process can be externally verified. However, VerifyNet does not empower individual clients to verify whether their specific updates were included in the global model, nor does it remove the reliance on a trusted aggregator. These are crucial limitations in FL scenarios, where accountability and transparency at the participant level are essential

Cryptographic accumulators are a powerful primitive for enabling efficient membership proofs. While they are widely adopted in privacy-preserving protocols [31], their application in the context of FL remains limited. In FL, accumulators have primarily been leveraged to verify user participation or to support authentication, rather than to confirm a client's actual contribution to the global model. For instance, PFLM [32] introduces a membership-proof mechanism based on threshold secret sharing, where the server maintains an accumulator of active user identifiers, and the resulting proofs are anchored on a public blockchain. Similarly, PrSeFL [33] utilizes accumulators for lightweight and anonymous user authentication. However, in both cases, the generated proofs demonstrate client participation in the system but do not offer guarantees regarding the inclusion of a client's model update in the final aggregated model. As such, the fundamental problem of verifiable contribution in FL remains largely unaddressed.

9 Conclusive remarks

FL clients require efficient mechanisms to verify whether their contributions have been incorporated into the global model. This paper presents MPFL, i.e., a novel method that enables client-side proofs of participation in FL. MPFL leverages a blockchain smart contract to aggregate client updates and uses an ECC-based cryptographic accumulator to generate verifiable proofs. The reported experimental results show that MPFL clients can efficiently verify their participation with minimal computational and storage overhead, regardless of the underlying ML model.

Acknowledgements This work was partially supported by the project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan program.

Author Contributions Carlo Mazzocca, Alessio Mora, and Nicolò Romandini contributed to the design, implementation, and empirical evaluation of the proposed mechanism, and were primarily responsible for drafting the manuscript. Rebecca Montanari and Paolo Bellavista contributed to the design and provided critical revisions, helping shape the final version of the manuscript.

Funding Open access funding provided by Università degli Studi di Salerno within the CRUI-CARE Agreement.

Data Availability No datasets were generated or analysed during the current study.

Declarations

Conflicts of Interest The author declares that there are no financial or non-financial conflicts of interest related to this work.

Research involving Human Participants and/or Animals This research did not involve human participants or animals.

Informed Consent Not applicable. The study exclusively uses publicly available datasets for evaluation, and no personal or sensitive data were collected or processed.

Competing interests The authors declare no competing interests.

References

1. Zhou, L., Pan, S., Wang, J., Vasilakos, A.V.: Machine learning on big data: opportunities and challenges. Neurocomputing **237**, 350–361 (2017). https://doi.org/10.1016/j.neucom.2017.01.026
2. Bellavista, P., Foschini, L., Mora, A.: Decentralised learning in federated deployment environments: a system-level survey. ACM Comput. Surv. **54**(1), 1–38 (2021). https://doi.org/10.1145/3429252
3. Zhou, Z., Chu, L., Liu, C., Wang, L., Pei, J., Zhang, Y.: Towards Fair Federated Learning. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining. KDD '21, pp. 4100–4101. Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3447548.3470814
4. Zhan, Y., Zhang, J., Hong, Z., Wu, L., Li, P., Guo, S.: A survey of incentive mechanism design for federated learning. IEEE Trans. Emerg. Top. Comput. **10**(2), 1035–1044 (2022). https://doi.org/10.1109/TETC.2021.3063517
5. Mothukuri, V., Parizi, R.M., Pouriyeh, S., Huang, Y., Dehghantanha, A., Srivastava, G.: A survey on security and privacy of federated learning. Futur. Gener. Comput. Syst. **115**, 619–640 (2021). https://doi.org/10.1016/j.future.2020.10.007
6. Xu, G., Li, H., Liu, S., Yang, K., Lin, X.: VerifyNet: secure and verifiable federated learning. IEEE Trans. Inf. Forensics Secur. **15**, 911–926 (2020). https://doi.org/10.1109/TIFS.2019.2929409
7. Kalapaaking, A.P., Khalil, I., Atiquzzaman, M.: Blockchain-enabled and multisignature-powered verifiable model for securing federated learning systems. IEEE Internet Things J. **10**(24), 21410–21420 (2023). https://doi.org/10.1109/JIOT.2023.3289832
8. Peng, Z., Xu, J., Chu, X., Gao, S., Yao, Y., Gu, R., Tang, Y.: VFChain: enabling verifiable and auditable federated learning via blockchain systems. IEEE Transactions on Network Science and Engineering **9**(1), 173–186 (2022). https://doi.org/10.1109/TNSE.2021.3050781
9. Mazzocca, C., Mora, A., Romandini, N., Montanari, R., Bellavista, P.: Membership Proof in Federated Learning via Cryptographic Accumulators. In: 2024 6th International Conference on Blockchain Computing and Applications (BCCA), pp. 57–63 (2024). https://doi.org/10.1109/BCCA62388.2024.10844447
10. Vitto, G., Biryukov, A.: Dynamic universal accumulator with batch update over bilinear groups. In: Galbraith, S.D. (ed.) Topics in Cryptology - CT-RSA 2022, pp. 395–426. Springer, Cham (2022)
11. Mazzocca, C., Romandini, N., Montanari, R., Bellavista, P.: Enabling Federated Learning at the Edge through the IOTA Tangle. Futur. Gener. Comput. Syst. **152**, 17–29 (2024). https://doi.org/10.1016/j.future.2023.10.014
12. Nguyen, D.C., Ding, M., Pham, Q.-V., Pathirana, P.N., Le, L.B., Seneviratne, A., Li, J., Niyato, D., Poor, H.V.: Federated learning meets blockchain in edge computing: opportunities and challenges. IEEE Internet Things J. **8**(16), 12806–12825 (2021). https://doi.org/10.1109/JIOT.2021.3072611
13. Bhutta, M.N.M., Khwaja, A.A., Nadeem, A., Ahmad, H.F., Khan, M.K., Hanif, M.A., Song, H., Alshamari, M., Cao, Y.: A survey on blockchain technology: evolution, architecture and security. IEEE Access **9**, 61048–61073 (2021). https://doi.org/10.1109/ACCESS.2021.3072849
14. Wang, L., Tian, Y., Zhang, D.: Toward cross-domain dynamic accumulator authentication based on blockchain in internet of things. IEEE Trans. Industr. Inf. **18**(4), 2858–2867 (2021)
15. Kumar, A., Lafourcade, P., Lauradoux, C.: Performances of cryptographic accumulators. In: 39th Annual IEEE Conference on Local Computer Networks, pp. 366–369 (2014). IEEE
16. McMahan, H.B., et al.: Communication-efficient Learning of Deep Networks from Decentralized Data. arXiv preprint arXiv:1602.05629 (2016)
17. Zhan, Y., Zhang, J., Hong, Z., Wu, L., Li, P., Guo, S.: A survey of incentive mechanism design for federated learning. IEEE Trans. Emerg. Top. Comput. **10**(2), 1035–1044 (2021)
18. Romandini, N., Mora, A., Mazzocca, C., Montanari, R., Bellavista, P.: Federated Unlearning: A Survey on Methods, Design Guidelines, and Evaluation Metrics. IEEE Trans. Neural Netw. Learn. Syst, 1–21 (2024) https://doi.org/10.1109/TNNLS.2024.3478334
19. Sun, Q., Li, X., Zhang, J., Xiong, L., Liu, W., Liu, J., Qin, Z., Ren, K.: Shapleyfl: robust federated learning based on shapley value. In: Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 2096–2108 (2023)
20. Xue, Y., Niu, C., Zheng, Z., Tang, S., Lyu, C., Wu, F., Chen, G.: Toward understanding the influence of individual clients in federated learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, **35**, 10560–10567 (2021)
21. Poirot, M.G., Vepakomma, P., Chang, K., Kalpathy-Cramer, J., Gupta, R., Raskar, R.: Split learning for collaborative deep learning in healthcare. arXiv preprint arXiv:1912.12115 (2019)
22. Li, D., Wang, J.: Fedmd: heterogenous federated learning via model distillation. arXiv preprint arXiv:1910.03581 (2019)
23. Jeong, E., Oh, S., Kim, H., Park, J., Bennis, M., Kim, S.-L.: Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data. arXiv preprint arXiv:1811.11479 (2018)
24. MMw-Unibo: MPFL: Membership Proof in Federated Learning via Cryptographic Accumulators. https://github.com/MMw-Unibo/MPFL. Accessed 14 Jul 2024 (2024)

25. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. CoRR arXiv:1708.07747 (2017)
26. Krizhevsky, A.: Learning multiple layers of features from tiny images. Technical report (2009)
27. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.-C.: Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018)
28. Tan, M., Le, Q.: Efficientnetv2: Smaller models and faster training. In: International Conference on Machine Learning, pp. 10096–10106 (2021). PMLR
29. Camacho, P., Hevia, A.: On the impossibility of batch update for cryptographic accumulators. In: Progress in Cryptology–LATINCRYPT 2010: First International Conference on Cryptology and Information Security in Latin America, Puebla, Mexico, August 8-11, 2010, Proceedings 1, pp. 178–188 (2010). Springer
30. Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., Kiddon, C., Konečný, J., Mazzocchi, S., McMahan, B., Van Overveldt, T., Petrou, D., Ramage, D., Roselander, J.: Towards Federated Learning at Scale: System Design. In: Talwalkar, A., Smith, V., Zaharia, M. (eds.) Proceedings of Machine Learning and Systems, **1**, 374–388 (2019). https://proceedings.mlsys.org/paper_files/paper/2019/file/7b770da633baf74895be22a8807f1a8f-Paper.pdf
31. Ren, Y., Liu, X., Wu, Q., Wang, L., Zhang, W.: Cryptographic accumulator and its application: a survey. Secur. Commun. Netw. **2022**, 5429195 (2022). https://doi.org/10.1155/2022/5429195
32. Jiang, C., Xu, C., Zhang, Y.: PFLM: privacy-preserving federated learning with membership proof. Inf. Sci. **576**, 288–311 (2021). https://doi.org/10.1016/j.ins.2021.05.077
33. Xiao, Y., Xu, L., Wu, Y., Sun, J., Zhu, L.: PrSeFL: achieving practical privacy and robustness in blockchain-based federated learning. IEEE Internet Things J. **11**(24), 40771–40786 (2024). https://doi.org/10.1109/JIOT.2024.3454087