Membership Proof in Federated Learning via Cryptographic Accumulators

Published Version:
Mazzocca, C., Mora, A., Romandini, N., Montanari, R., Bellavista, P. (2024). Membership Proof in Federated Learning via Cryptographic Accumulators. IEEE [10.1109/bcca62388.2024.10844447].

Availability:
This version is available at: https://hdl.handle.net/11585/1003180 since: 2025-03-19

Published:
DOI: http://doi.org/10.1109/bcca62388.2024.10844447

Membership Proof in Federated Learning via Cryptographic Accumulators

Carlo Mazzocca, Alessio Mora, Nicolò Romandini, Rebecca Montanari, Paolo Bellavista
Department of Computer Science and Engineering
University of Bologna, Bologna, Italy
{name.surname}@unibo.it

***Abstract*—Federated Learning (FL) is a decentralized training paradigm where clients collaboratively train a Machine Learning (ML) model without outsourcing their raw data. Each participant locally trains a model utilizing their data, and these models are periodically aggregated to build a global model. In this scenario, clients may be willing to verify the inclusion of their contributions to ensure accurate and fair computation of the global model. This verification also serves to confirm that their participation has been adequately rewarded. Therefore, FL frameworks should allow clients to efficiently verify their proof of inclusion (i.e., *membership proof*) in the training without affecting privacy. This paper presents a protocol for efficient membership proof in FL (MPFL). Our protocol leverages cryptographic accumulators, which enable clients to verify membership proof with minimum overhead, and a smart contract deployed on a blockchain to ensure the correct generation of the global model and membership proofs. We implemented MPFL and conducted evaluations across various datasets, ML models, and varying the number of clients. The experimental findings reveal that a client only requires 96 bytes to maintain the necessary information for verifying their inclusion, achieving this task in approximately 20 ms.**

***Index Terms*—Federated Learning, Blockchain, Cryptographic Accumulator, Smart Contract, Membership Proof**

I. INTRODUCTION

With the increasing digitalization of our lives and recent technological advancements such as the Internet of Things (IoT), we are experiencing an unprecedented surge in data generation. These large datasets are the cornerstone for developing Machine Learning (ML) models [1]. However, traditional ML approaches rely on centralized data collection, necessitating the transfer of data from where it is generated to central servers. This centralization raises serious privacy concerns, as data could be potentially leaked.

To address these challenges, Federated Learning (FL) has emerged as a promising solution. FL is a privacy-preserving ML method that allows clients to collaboratively train a model without sharing their local data with any third parties [2]. Each client trains a local ML model with their on-premises data and only sends model updates (e.g., gradients or weights) to a parameter server that aggregates them to build a global model.

In FL, participants may seek to verify the inclusion of their model updates in generating the global model. This is motivated by the need to ensure the accurate computation of the global model, incorporating contributions from all clients [3]. Moreover, one of the main challenges in FL is to incentivize clients to actively contribute to the collaborative training process [4]. Consequently, participants may be willing to verify if their contributions are included in the global model to claim corresponding rewards.

Given these considerations, FL frameworks should allow clients to efficiently verify their *membership proof.* With the term membership proof, we refer to the evidence confirming that a model update has contributed to the build of the global model. It is worth noting that one of the key features of FL is safeguarding client privacy [5], meeting the following criteria: (i) client data is not directly accessed and (ii) participants in the FL process do not have to be necessarily identifiable. Therefore, FL platforms should implement membership-proof mechanisms that enable the efficient verification of inclusion in the training without compromising privacy. Numerous studies [6]–[8] have focused on addressing the problem of ensuring the correctness of the global model. However, they overlook the need for clients to efficiently verify whether their contributions have been utilized in generating the global model or not.

This paper presents the first protocol for efficient membership proof in FL (MPFL), which is designed to enable clients to verify the inclusion of their updates in the global model with minimal overhead, all while preserving their privacy. Specifically, we leverage an Elliptic Curve Cryptography (ECC)-based [9], a powerful tool that condenses multiple elements into a unique, constant-size value. The latest value of the accumulator is used to generate membership proofs, commonly known as *witnesses*. In this work, the aggregated elements within the accumulator represent the contributions of clients that are selected to build the global model. Furthermore, to ensure the accurate generation of both the global model and the associated accumulators and witnesses, these tasks are performed by a smart contract deployed on a blockchain. Replacing the parameter server with blockchain and smart contracts enables more secure FL [10]. Indeed, traditional FL platforms based on the client-server paradigm suffer from a single point of failure, scalability challenges, and susceptibility to tampering since the server may alter the generation of the global model [11]. These vulnerabilities can compromise the integrity of the global model, potentially introducing biases that favor certain model updates over others.

We implemented and evaluated MPFL across different datasets and ML models, varying the number of participants in the FL training. Experimental results show that a client needs a mere 96 bytes of data storage to maintain their membership proof and the accumulator value, which is necessary to verify their inclusion in the global model. This verification is performed in approximately 20 ms, proving the efficiency of MPFL for FL clients.

In the following, we summarize the main contributions of this paper:

- We provide a detailed analysis that motivates why verifying inclusion is a fundamental property that should be implemented by any FL framework;
- We propose the first efficient membership proof scheme that enables clients to directly verify the inclusion of their contributions in the global model;
- We implement and evaluate MPFL under different datasets, ML models, and the number of FL participants; thus, demonstrating its efficiency in enabling clients to verify their inclusion in the global model.

The remainder of this paper is structured as follows. Section II introduces our background, while Section III presents the related work in the field. Section IV motivates why enabling clients to verify inclusion in FL is important in FL. In Section V, we describe our protocol which is then evaluated in Section VI. Finally, Section VII concludes the paper.

II. BACKGROUND

A. Federated Learning

FL trains a shared ML model by optimizing a global objective function that can be formulated as a weighted average over all participants' private datasets. Supposing that there are N participants in the federation, with each client i holding a local dataset D_i, the objective is to minimize the function $f(w)$ defined as follows:

$$\min_{w} f(w) := \sum_{i=1}^{N} \frac{n_i}{n} F_i(w), \tag{1}$$

where w represents the parameters of the shared model to be trained, n_i represents the cardinality of D_i, and n is the total amount of samples across participants. F_i is the client's objective function such as cross-entropy loss for a supervised classification task.

Federated Averaging (FedAvg) [12] is the baseline solution to perform FL. FedAvg is a synchronous FL algorithm, where a central entity, i.e., the server, coordinates a federation of learning devices, i.e., the clients. FedAvg operates in rounds and follows a series of steps to approximate the optimization problem of Eq. 1. At each round, the global model is distributed to a random subset of participant devices, which in turn fine-tune the model locally for a certain number of epochs. Next, such activated clients transmit back the difference between the locally computed and the received model parameters, also referred to as updates. The server gathers the updates and aggregates them using weighted averaging based on the number of local data points held by clients. Finally, the server applies the averaged updates to the current global model. The process starts again until certain metrics (e.g., accuracy or total number of FL rounds) are reached.

B. Blockchain

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

C. Cryptographic Accumulators

Cryptographic accumulators aggregate many different data into a fixed-length digest called accumulator value v. Accumulators offer an efficient approach for verifying whether an item is part of a set. This verification is accomplished by leveraging membership and non-membership witnesses. For each element e_i, its corresponding witness p_i is derived by aggregating all the values except e_i. Accumulator schemes 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*.

Cryptographic accumulators are also classified according to underlying cryptography. Among existing implementations, Elliptic Curve Cryptography (ECC)-based accumulators offer several advantages such as smaller witnesses and efficient verification time [14]. Given the focus of this paper, we leverage a positive ECC-based accumulator to verify the inclusion of client model updates in the global model.

III. RELATED WORK

Despite the need for clients to efficiently verify the inclusion of their updates into the global model, current literature lacks mechanisms to address this issue. Related research mainly focuses on verifying the correct creation of the global model. One of the first frameworks that offers verifiable FL is VerifyNet [6]. In VerifyNet, each client encrypts their update and submits it to the server. Once a sufficient number of contributions are received, the model updates are aggregated and the result is returned to the participants along with a cryptographic proof that allows clients to verify the correctness of the global model. This proof is computed using homomorphic hash functions and pseudorandom technologies. This approach does not ensure that a client's contributions have been included in the global model, and heavily relies on the engagement of central servers, whose limitations have been underscored previously.

To overcome the challenges of central servers, several works leverage the integration of blockchain, smart contracts, and FL to facilitate global model verification. Kalapaaking et al. [7] employ a blockchain to aggregate local models computed within a trusted execution environment. Consensus is achieved when the hashes of global models generated by all blockchain aggregation nodes match. Subsequently, the global model is transmitted to another blockchain, where all nodes endorse it via a multi-signature scheme before storing it. However, using two blockchains introduces a remarkable overhead, and this platform fails to guarantee the inclusion of client contributions in the global model. VFChain [8] is a verifiable and auditable framework for FL. When the number of received updates is larger than θ, a smart contract aggregates client contributions and generates a global model. For each global model update, a signature is recorded in the blockchain. Furthermore, VFChain employs a novel data structure that enables efficient retrieval of information stored in the blockchain. This framework does not offer a mechanism that enables FL participants to verify their inclusion in the global model. For example, a client should be able to assess if their updates fall in the interval defined by θ.

Cryptographic accumulators are commonly used to generate privacy-preserving membership proofs [15]; however, only Jian et al. [16] use them in the FL context. The authors presented PFLM, a privacy-preserving FL framework with membership proof based on a (t, N)-threshold secret sharing scheme. This scheme works under the assumption that the threshold t is greater than $\lfloor n/2 \rfloor + 1$ (n denotes the number of clients in FL) with a maximum allowance of $\lceil n/2 \rceil - 1$ clients permitted to exit. To facilitate verification of client connectivity, the server maintains an accumulator of available user IDs, issuing corresponding membership proofs on a public blockchain. In this context, membership proofs confirm client connectivity rather than efficiently verifying the inclusion of their partial models in the global model.

IV. VERIFY INCLUSION IN FL: WHY IS IT CRUCIAL?

Ensuring the inclusion of clients' contributions in the generation of the global model stands as a key principle for the design of any FL framework. Clients may be willing to verify their inclusion for several reasons.

Firstly, guaranteeing inclusion is crucial for fostering the widespread adoption of FL. Participants are interested in ensuring the correct generation of the global model, incorporating contributions from all clients. This confirms that the model is computed without bias and empowers clients to understand how their data impacts the global model. For clients seeking tailored experiences, such as those in recommendation systems, verifying inclusion grants assurance of their influence on the global model, thereby enhancing user satisfaction.

Furthermore, knowing that their contributions directly impact the global model encourages clients to actively participate in FL training. To join FL, participants are often incentivized with some rewards such as data monetization or enhanced model performance. Therefore, they may be seeking to verify that their updates are used in the global model generation to redeem corresponding rewards.

Additionally, verifying inclusion can also serve to detect potential anomalies. Instances may arise where clients are excluded due to factors such as the poor quality of their model updates or network issues hindering the collection of their contributions. By assessing their inclusion status, clients can identify and address such anomalies and investigate to address them.

Finally, verifying inclusion holds promise for applications beyond traditional FL, extending to emerging paradigms like Federated Unlearning (FU) [17]–[19]. FU's objective consists of removing specific clients' contributions from the global model. Here, the aim is to enable data owners to verify that their contributions *are not included* in the global model. It is worth noting that providing a mechanism for clients to verify the absence of their contributions from the global model lies beyond the scope of this paper. MPFL can successfully be employed to verify inclusion for the other motivations outlined in this subsection.

V. MEMBERSHIP PROOF IN FEDERATED LEARNING

We consider a FL process where clients $c_i \in \mathcal{C}$ collaborate to train a global model g^k. During each round k, each c_i processes their local dataset D_i and produces a model update m_i^k. All model updates M^k collected during k are aggregated by a smart contract s executed on the blockchain b. The aggregation M^k results in the build of g^k. A client c_i may want to verify if their contributions have been included in the global model. This membership proof is accomplished by using an accumulator value a^k and a witness p_i^k, which is associated with each m_i^k. Figure 1 offers an overview of MPFL.

A. Preliminaries

This subsection offers all the details related to how the accumulator and witnesses are used in MPFL to prove have been included in g^k. Membership proof is associated with the concept of witness and it is defined as follows:

Definition 1 (Membership Proof). *In federated learning, membership proof refers to the capability of a client to efficiently verify whether their contributions have been included in the global model for a given round k.*

Inputs of the Accumulator. The values included in a^k are obtained by the hash conversion of $m_i^k \in M^k$. Thus, the security of a^k depends on the hash function adopted. We employ the SHA-256 hash functions given its capability of being secure and collision-resistant.

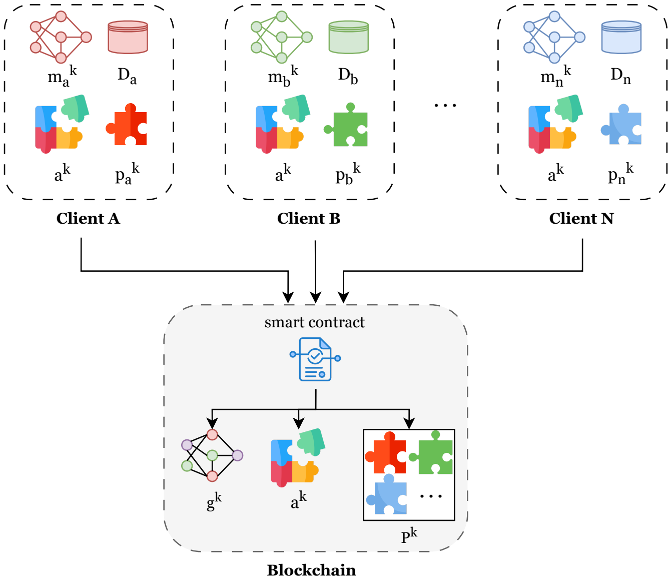


Fig. 1: Overview of MPFL.

Functions. In the following, we present the functions needed by MPFL to achieve membership proof in FL:

- $m_i^k \leftarrow \mathsf{ComputePartialModel}(\mathsf{D_i}, \mathsf{g^{k-1}})$: each c_i train an updated local model given their private data $\mathsf{D_i}$ and the global model of the previous round $\mathsf{g^{k-1}}$.
- $g^k \leftarrow \mathsf{ComputeGlobalModel}(\mathsf{M^k})$: s computes the novel global model by averaging all $m_i^k \in M^k$.
- $a_{info} \leftarrow \mathsf{Setup}(\mathsf{K})$: setups the parameters of $\mathsf{a_j}$ by taking a set of public values K, which represent the random generators of the elliptic curve.
- $a^k \leftarrow \mathsf{ComputeAccumulator}(\mathsf{a_{info}}, \mathsf{M_i^k})$: accumulates collected model updates $\mathsf{M_i^k}$ obtaining the corresponding point of a curve.
- $p_i^k \leftarrow \mathsf{ComputeMembershipProof}(\mathsf{a^k}, \mathsf{m_i^k})$: generates membership proof value $\mathsf{p_i^k}$ for $\mathsf{m_i^k}$, providing proof that they have been included in $\mathsf{a_k}$.
- $0, 1 \leftarrow \mathsf{Verify}(\mathsf{a^k}, \mathsf{m_i^k}, \mathsf{p_i^k})$: each c_i verifies whether $\mathsf{p_i^k}$ associated with $\mathsf{m_i^k}$ belongs to $\mathsf{a^k}$.

B. Protocol

During each round k of FL, all participating c_i engaged in the collaborative training compute m_i^k on their respective local D_i. This computation is performed by executing the $\mathsf{ComputePartialModel}()$ function, which also takes as input the previous version of the global model g^{k-1}. Subsequently, model updates are then transmitted to the smart contract s, which is responsible for building g^k.

Upon satisfying the aggregation criteria, s generates g^k by using all the collected M^k and g^{k-1} as inputs for the $\mathsf{ComputeGlobalModel}()$ function. This function additionally executes the $\mathsf{Setup}()$, $\mathsf{ComputeAccumulator}()$, and $\mathsf{ComputeMembershipProof}()$ procedures essential for enabling clients to verify the inclusion of their updates in the FL. Specifically, when a $m_i^k \in M^k$ is included in g_k, it undergoes hashing and subsequent accumulation in a^k through the $\mathsf{ComputeAccumulator}()$ function. Following the computation of a^k, s executes the $\mathsf{ComputeMembershipProof}()$ for each m_i^k accumulated, generating the corresponding membership proof p_i^k. These proofs are then disseminated to c_i along with the novel version g^k. The pseudocode of the $\mathsf{ComputeGlobalModel}()$ function is reported in Algorithm 1.

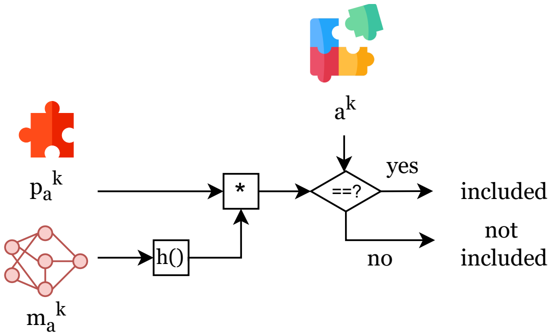


Fig. 2: Verification of inclusion.

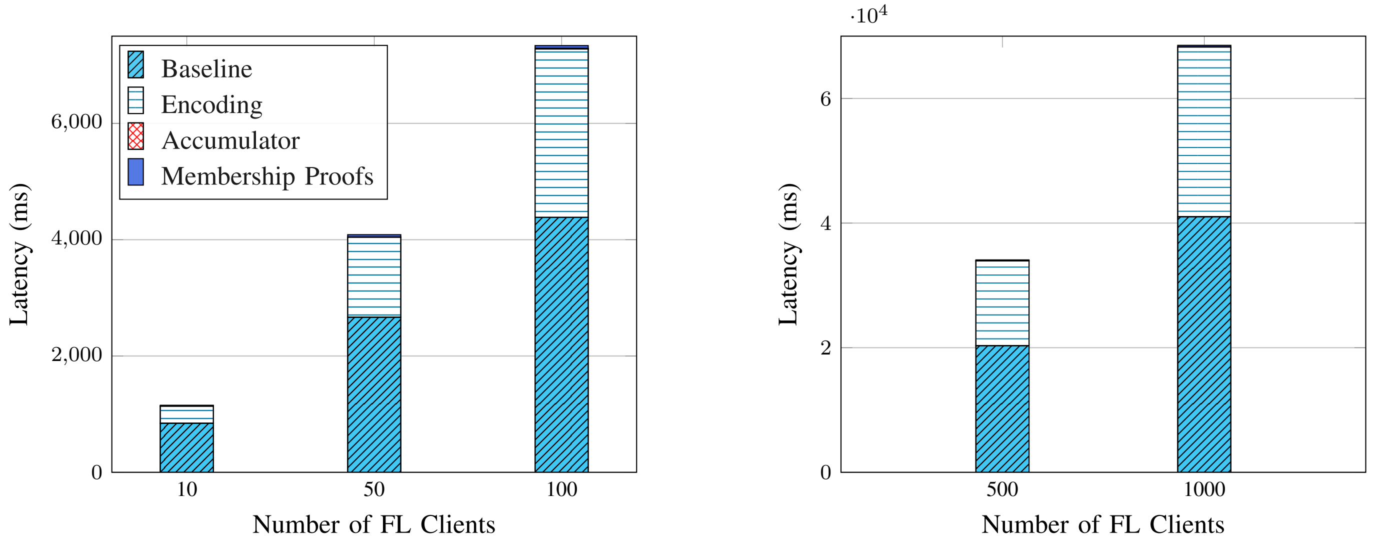
Algorithm 1 Smart contract that generates global model and membership proofs.

```
Data: Model updates $M^k$
Result: Global model $g^k$ and membership proofs $P^k$
Function ComputeGlobalModel($M^k$):
    $K \leftarrow$ random generator elliptic curve;
    $a_{info} \leftarrow \mathsf{Setup(K)}$
    $a^k \leftarrow \mathsf{ComputeAccumulator(a_{info}, M^k)}$
    for $m_i^k \in M^k$ do
        $p_i^k \leftarrow \mathsf{ComputeMembershipProof(a^k, M^k)}$
        push $p_i^k$ in $P^k$
    end
    $N \leftarrow$ size of $M^k$
    $g^k \leftarrow \frac{\sum_{i=1}^{N} n_i \cdot m_i^k}{n}$
    return $g^k, P^k$
```

When c_i wants to verify if their m_i^k was included in g^k, they execute the $\mathsf{Verify}()$ function, whose overview is sketched in Figure 2. Specifically, this verification function combines m_i^k and p_i^k. If the result of this operation is equal to a^k, the client update was correctly included in the FL process.

C. Discussion

Using blockchain and smart contracts enhances the trustworthiness of clients in the FL process. By executing the generation of the global model directly on the blockchain, every participant can analyze the correctness of the smart contract logic. This transparency empowers clients to detect potential vulnerabilities or biases in the global model generation. Nevertheless, this still does not guarantee that client contributions have been included in the aggregation.



(a)

(b)

Fig. 3: Lantency for generating global model, accumulator value, and membership proofs using Fashion-MNIST dataset, 1-layer neural network, and varying the number of FL clients.

Various factors might impede a client's model updates from being received, such as network connectivity issues or the specific logic within the smart contract that triggers global model generation only under certain conditions (e.g., when a predetermined number of contributions is reached). In these circumstances, verifying the inclusion in the global model becomes extremely challenging. Clients should query the blockchain and inspect the transaction history. Given the continuously expanding nature of blockchain, these operations prove cumbersome.

To overcome these concerns, MPFL offers an efficient membership proof that allows participants to directly verify whether their model has indeed been integrated into the global model with minimal overhead in terms of storage and verification time. Consequently, MPFL aptly accommodates the heterogeneous capabilities of clients involved in the FL training process.

VI. EVALUATION

To evaluate the efficacy of our membership proof protocol, we implemented it through a smart contract [20] 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 our blockchain platform deployed on a node with an Intel(R) Core(TM) i5-3470 CPU running at 3.20GHz and 12 GB of RAM.

A. Experiments

To comprehensively evaluate the overhead introduced by MPFL, we conducted a series of experiments with varying degrees of complexity regarding the ML models utilized, datasets employed, and number of clients involved. Specifically, our experiments aim to evaluate which is the overhead introduced by MPFL to generate membership proof during the generation of the global model.

Initially, we utilized the Fashion-MNIST dataset, consisting 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 employed in this experiment features three layers: a Flatten input layer, a Dense layer consisting of 128 neurons, and an output layer containing 10 neurons, corresponding to the 10 classes. In total, the network comprises 101,770 parameters. Subsequently, we increased the complexity by employing CIFAR-10 and the MobileNetV2 neural network architecture proposed by Sandler et al. [21]. CIFAR-10 is a widely recognized dataset comprising 60,000 32x32 color images distributed across 10 classes, with 6,000 images per class. The dataset is partitioned into 50,000 training images and 10,000 test images. The MobileNetV2 architecture consists of 2,270,794 parameters, approximately 20 times more than the network utilized in the initial experiment.

The neural network is trained locally for 5 epochs in both experiments, followed by 10 FL rounds. Additionally, the datasets were equally partitioned among all FL clients to ensure fair distribution of computational load. Given that the generation of membership proofs inevitably 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.

B. Results

Figures 3 and 4 show the overhead introduced by MPFL across the two datasets and ML models under study and by varying the number of involved clients. Specifically, we report the latency, expressed in ms, incurred in computing

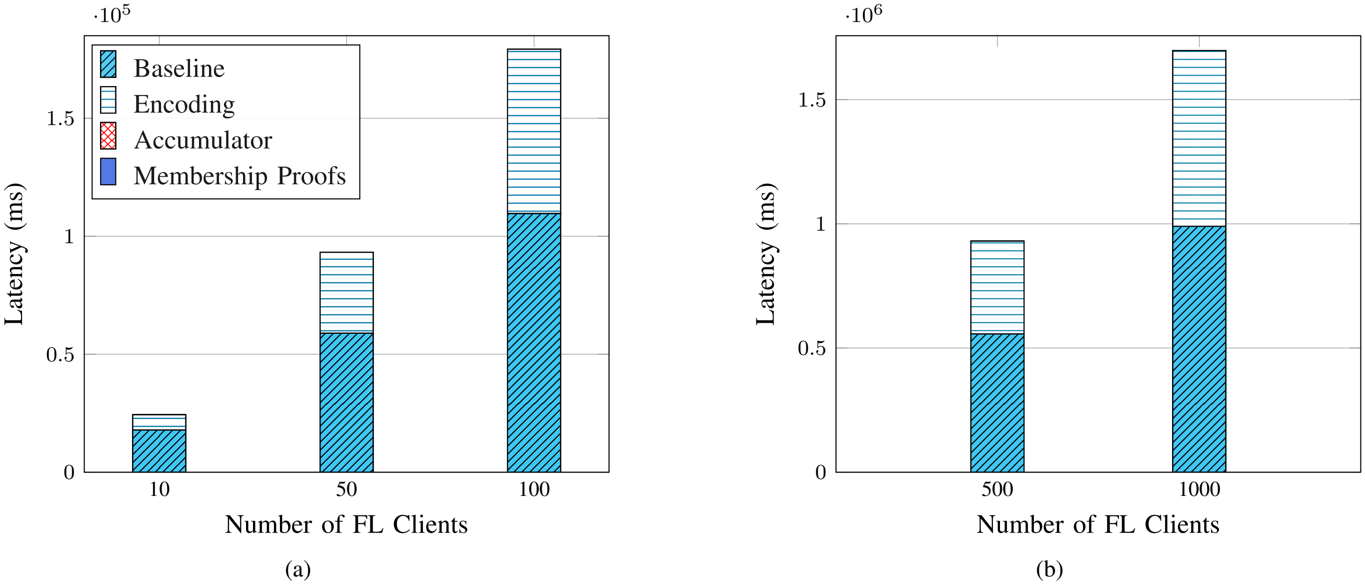


Fig. 4: Lantency for generating global model, accumulator value, and membership proofs using CIFAR-10 dataset, MobileNetV2, and varying the number of FL clients.

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 an FL aggregation, we also measure the FedAvg latency, labeled as `baseline` in the figures.

Notably, the figures illustrate that generating the hashes for each model is the most time-consuming operation, and the time required to convert MobileNetV2 requires one order of magnitude compared to a 1-layer neural network. We do not observe significant differences in terms of latency for computing the accumulator value and generating the membership proofs (in the order of a few milliseconds) while scaling the number of FL clients and considering various datasets and ML models. However, 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. Furthermore, it is worth noting that the latency associated with membership proof scales with the number of proofs required: to generate p membership proof requires at least $\Omega(p)$ operations [22]. 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) [23].

Concerning client-side implications, each FL client must store 96 bytes to maintain the accumulator value and the membership proof associated with a model update. We also estimated the latency for verifying a model update inclusion and exclusion in the global model, which remains consistent at approximately 20 ms. Finally, due to the unique features of the ECC-based accumulator, both storage requirements and verification time remain unaffected by the number and characteristics of accumulated model updates.

VII. CONCLUSIONS

Most existing works focus on the correctness of the global model, while they overlook how clients can directly verify whether their contributions are included or not in the global model itself. While integrating blockchain and smart contracts can enhance the trustworthiness of the FL process, additional measures are necessary to facilitate membership proof.

This paper introduces MPFL, an efficient protocol designed to verify whether a client's model update is successfully integrated into the global model, all while preserving participant privacy. We conducted comprehensive implementations and evaluations of MPFL across diverse datasets and ML models, varying the number of participants in the FL training. Our evaluations underscore that through leveraging an ECC-based accumulator, FL clients require only minimal storage for membership proofs. Furthermore, verifying their model updates' inclusion in the global model is accomplished within 20 milliseconds.

ACKNOWLEDGMENTS

This work was partially supported by the project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan program funded by the European Union - NextGenerationEU.

References

[1] L. Zhou, S. Pan, J. Wang, and A. V. Vasilakos, “Machine learning on big data: Opportunities and challenges,” *Neurocomputing*, vol. 237, pp. 350–361, 2017.

[2] P. Bellavista, L. Foschini, and A. Mora, “Decentralised Learning in Federated Deployment Environments: A System-Level Survey,” *ACM Comput. Surv.*, vol. 54, no. 1, feb 2021.

[3] Z. Zhou, L. Chu, C. Liu, L. Wang, J. Pei, and Y. Zhang, “Towards Fair Federated Learning,” in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, ser. KDD ’21. New York, NY, USA: Association for Computing Machinery, 2021, p. 4100–4101.

[4] Y. Zhan, J. Zhang, Z. Hong, L. Wu, P. Li, and S. Guo, “A Survey of Incentive Mechanism Design for Federated Learning,” *IEEE Transactions on Emerging Topics in Computing*, vol. 10, no. 2, pp. 1035–1044, 2022.

[5] V. Mothukuri, R. M. Parizi, S. Pouriyeh, Y. Huang, A. Dehghantanha, and G. Srivastava, “A survey on security and privacy of federated learning,” *Future Generation Computer Systems*, vol. 115, pp. 619–640, 2021.

[6] G. Xu, H. Li, S. Liu, K. Yang, and X. Lin, “VerifyNet: Secure and Verifiable Federated Learning,” *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 911–926, 2020.

[7] A. P. Kalapaaking, I. Khalil, and M. Atiquzzaman, “Blockchain-Enabled and Multisignature-Powered Verifiable Model for Securing Federated Learning Systems,” *IEEE Internet of Things Journal*, vol. 10, no. 24, pp. 21 410–21 420, 2023.

[8] Z. Peng, J. Xu, X. Chu, S. Gao, Y. Yao, R. Gu, and Y. Tang, “VFChain: Enabling Verifiable and Auditable Federated Learning via Blockchain Systems,” *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 1, pp. 173–186, 2022.

[9] G. Vitto and A. Biryukov, “Dynamic Universal Accumulator with Batch Update over Bilinear Groups,” in *Topics in Cryptology – CT-RSA 2022*, S. D. Galbraith, Ed. Cham: Springer International Publishing, 2022, pp. 395–426.

[10] C. Mazzocca, N. Romandini, R. Montanari, and P. Bellavista, “Enabling Federated Learning at the Edge through the IOTA Tangle,” *Future Generation Computer Systems*, vol. 152, pp. 17–29, 2024.

[11] D. C. Nguyen, M. Ding, Q.-V. Pham, P. N. Pathirana, L. B. Le, A. Seneviratne, J. Li, D. Niyato, and H. V. Poor, “Federated Learning Meets Blockchain in Edge Computing: Opportunities and Challenges,” *IEEE Internet of Things Journal*, vol. 8, no. 16, pp. 12 806–12 825, 2021.

[12] H. B. McMahan *et al.*, “Communication-efficient Learning of Deep Networks from Decentralized Data,” *arXiv preprint arXiv:1602.05629*, 2016.

[13] M. N. M. Bhutta, A. A. Khwaja, A. Nadeem, H. F. Ahmad, M. K. Khan, M. A. Hanif, H. Song, M. Alshamari, and Y. Cao, “A Survey on Blockchain Technology: Evolution, Architecture and Security,” *IEEE Access*, vol. 9, pp. 61 048–61 073, 2021.

[14] A. Kumar, P. Lafourcade, and C. Lauradoux, “Performances of cryptographic accumulators,” in *39th Annual IEEE Conference on Local Computer Networks*, 2014, pp. 366–369.

[15] Y. Ren, X. Liu, Q. Wu, L. Wang, and W. Zhang, “Cryptographic Accumulator and Its Application: A Survey,” *Security and Communication Networks*, vol. 2022, p. 5429195, Mar 2022.

[16] C. Jiang, C. Xu, and Y. Zhang, “PFLM: Privacy-preserving federated learning with membership proof,” *Information Sciences*, vol. 576, pp. 288–311, 2021.

[17] N. Romandini, A. Mora, C. Mazzocca, R. Montanari, and P. Bellavista, “Federated Unlearning: A Survey on Methods, Design Guidelines, and Evaluation Metrics,” *arXiv preprint arXiv:2401.05146*, 2024.

[18] Z. Liu, Y. Jiang, J. Shen, M. Peng, K.-Y. Lam, and X. Yuan, “A survey on federated unlearning: Challenges, methods, and future directions,” *arXiv preprint arXiv:2310.20448*, 2023.

[19] Z. Liu, H. Ye, C. Chen, and K.-Y. Lam, “Threats, attacks, and defenses in machine unlearning: A survey,” *arXiv preprint arXiv:2403.13682*, 2024.

[20] MMw-Unibo, “Mpfl: Membership proof in federated learning via cryptographic accumulators,” https://github.com/MMw-Unibo/MPFL, 2024, accessed: 2024-07-14.

[21] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “Mobilenetv2: Inverted residuals and linear bottlenecks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[22] P. Camacho and A. Hevia, “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*. Springer, 2010, pp. 178–188.

[23] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečný, S. Mazzocchi, B. McMahan, T. Van Overveldt, D. Petrou, D. Ramage, and J. Roselander, “Towards Federated Learning at Scale: System Design,” in *Proceedings of Machine Learning and Systems*, A. Talwalkar, V. Smith, and M. Zaharia, Eds., vol. 1, 2019, pp. 374–388. [Online]. Available: https://proceedings.mlsys.org/paper_files/paper/2019/file/7b770da633baf74895be22a8807f1a8f-Paper.pdf