μDFL: A Secure Microchained Decentralized Federated Learning Fabric atop IoT Networks

Ronghua Xu, Yu Chen
Department of Electrical and Computer Engineering, Binghamton University, Binghamton, NY 13902, USA
{rxu22, ychen}@binghamton.edu

***Abstract*—Federated Learning (FL) has been recognized as a privacy-preserving machine learning (ML) technology that enables collaborative training and learning of a global ML model based on the aggregation of distributed local model updates. However, security and privacy guarantees could be compromised due to malicious participants and the centralized aggregation manner. Possessing attractive features like decentralization, immutability and auditability, Blockchain is promising to enable a tamper-proof and trust-free framework to enhance performance and security in IoT based FL systems. However, directly integrating blockchains into the large scale IoT-based FL scenarios still faces many limitations, such as high computation and storage demands, low transactions throughput, poor scalability and challenges in privacy preservation. This paper proposes μDFL, a novel hierarchical IoT network fabric for decentralized federated learning (DFL) atop of a lightweight blockchain called *microchain*. Following the hierarchical infrastructure of FL, participants in μDFL are fragmented into multiple small scale microchains. Each microchain network relies on a hybrid Proof of Credit (PoC) block generation and Voting-based Chain Finality (VCF) consensus protocol to ensure efficiency and privacy-preservation at the network of edge. Meanwhile, microchains are federated vie a high-level inter-chain network, which adopts an efficient Byzantine Fault Tolerance (BFT) consensus protocol to achieve scalability and security. A proof-of-concept prototype is implemented, and the experimental results verify the feasibility of the proposed μDFL solution in cross-devices FL settings with efficiency, security and privacy guarantees.**

***Index Terms*—Internet of Things (IoT), Federated Learning, Hierarchical Blockchain, Proof of Credit, Security, Privacy.**

I. Introduction

The proliferation of the Internet of Things (IoT) and advancements in machine learning (ML) promote the fusion of Big Data and artificial intelligence (AI), which make the concept of Smart Cities become realistic [38]. To build a safe and sustainable urban environment for its residents, IoT based smart applications highly depend on an efficient, secure, and low-cost real-time data sharing mechanism among device owners and third-party applications [26], [37]. However, sharing sensitive data among geographically scattered device owners and service providers also brings increasingly concerns on security and privacy. Motivated by privacy preservation among data owners, Federated Learning (FL) was recently proposed to build a distributed ML framework that enables training on a large corpus of decentralized data residing on devices like mobile phones [24]. Compared with conventional centralized ML approaches, only model updates that are computed on raw data of distributed clients are aggregated on data centers in FL, so that communication costs are reduced and network latency is decreased. Moreover, FL allows users to collaboratively train a shared model while keeping data on their devices, thus alleviating their privacy concerns [20]. FL provides a prospective solution to enable collaborative deep learning for smart applications that require scalable and privacy-preserving data and model updates sharing among massively distributed data owners and service providers.

While FL enables privacy preservation by sharing parameters of the trained model without directly exposing actual data on devices, it also brings new architecture and security concerns. The participants in conventional FL collaboratively train their local models and aggregate a global model under the orchestration of a central server. Such a centralized framework can be a performance bottleneck and susceptible to a single point of failure risk. In addition, dishonest participants can conclude with each other to compromise security and privacy of shared data and models, like poisoning and evasion attacks [13]. An ideal FL framework should be able to support decentralization, auditability and Byzantine Fault Tolerance (BFT) [17] compatibility, to secure model learning and inference process under a trustless and distributed IoT network environment.

Blockchain [25] has demonstrated great potential to revolutionize the fundamentals of information technology. It is a natural candidate to secure the decentralized architecture for FL, in which the data can be stored and verified distributively under a peer-to-peer (P2P) network without relying on a centralized authority. Such a decentralized architecture can improve system performance and mitigate single point failure issues existing in a centralized FL framework. Furthermore, leveraging a distributed ledger secured by consensus protocols, blockchain provides a verifiable, traceable, and append-only chained data structure of transactions. Whereas, integrating blockchain into a FL framework not only establishes secure connections among participants to protect confidentiality of data and privacy of model updates, it also guarantees both auditability and traceability to ensure data availability, correctness and provenance.

Existing research on decentralized federated learning (DFL) mostly focuses on combination of cryptographic schemes to ensure data confidentiality and privacy in distributed model training process [9], [21] or incentive mechanisms for fair rewards and verifiable FL models by the edge computing [15], [32]. However, the problem of designing a lightweight blockchain with FL-IoT scenarios remains open, like efficient miner selection, consensus algorithm and chain validation [12]. In this paper, we address aforementioned issues from blockchain design aspects and propose μDFL, a novel hierarchical microchained fabric for security and privacy of DFL atop IoT networks. Unlike existing work [15], [27], [32] that rely on a mono blockchain to provide provenance and integrity in FL tasks, μDFL adopts a federated networking framework [35] to balance trade-offs in terms of performance, security and scalability under the large-scale and hierarchical IoT-based cross-devices FL settings.

For each local training network, low-level aggregators and FL clients leverage a private microchain [34] to ensure efficiency and privacy-preservation of data sharing and local model aggregation at the edge network. Meanwhile, a high-level public inter-chain network inter-connects multiple fragmented private microchains to guarantee performance and security in global model propagation and aggregation. In summary, this paper makes the following contributions:

1) A complete μDFL architecture is presented along with details of key components and work flows;
2) A lightweight blockchain, *microchain* is proposed to achieve resource efficiency and privacy-preserving of executing consensus protocol at the edge network;
3) The core design of the microchain is illustrated in detail, which consists of Proof-of-Credit (PoC) block generation, Voting-based Chain Finality (VCF), and incentive mechanism; and
4) A proof-of-concept prototype is implemented and tested on a small-scale physical network. The experimental results show that μDFL only incurs less than 2 seconds latency in a FL cycle on a distributed MNIST dataset among 12 Rasppery Pis, and it needs 3.4 MB/s bandwidth as the system throughput is 1000 transactions per second. Our comparative evaluation demonstrates that microchain is lighter as running on IoT devices than Ethereum adopted by current DFL solutions.

The remainder of this paper is organized as follows: Section II discusses background knowledge of FLs and reviews existing consensus protocols and state-of-the-art research on blockchain-based distributed machine learning (DML). Section III introduces the rationale and architecture of μDFL, and microchain consensus protocol is described in Section IV. Section V provides details of the PoC algorithm, and Section VI-B explains the VCF mechanism. Incentive strategies are brief discussed in Section VII. Section VIII presents prototype implementation, numerical results and comparative analysis. Finally, Section IX concludes this paper with a brief discussion on our future directions.

II. Background and Related Work

A. Federated Learning

Artificial Intelligence (AI) has become an essential part of our lives today, following the recent successes and progression of Deep Learning (DL) in several domains, such as Computer Vision (CV) and Natural Language Processing (NLP) [20]. Conventional DL approaches adopt a centralized architecture in which raw data collection and model training are performed in a powerful server or data center. Thanks to the advancement in IoTs, the ubiquity of smart devices that are equipped with increasingly advanced sensing, computing and networking capabilities allow for migrating intelligence from the cloud to the edge. To enable a scalable collaborative training of complex models among distributed devices and guarantee privacy preservation of data owners, a decentralized approach of DL called Federated Learning (FL) is proposed. FL learns a shared model by aggregating locally-computing updates and leaves the training data distributed on the mobile devices [24]. FL is one instance of the more general approach of "*bringing the code to the data, instead of the data to the code*" and addresses the fundamental problems of privacy, ownership, and locality of data [5].

In general, the FL training process includes three steps: task initialization, local training and model update, and global model aggregation and update. In task initialization, the server specifies training configuration and initializes global model w_G^0, then broadcasts them to selected participants called *workers*. Given received w_G^t in current iteration t, each worker i utilizes its local data and computation resource to perform specified training tasks on parameters w_i^t. The goal of the training process is to find optimal parameters w_i^t that minimize the loss function $L(w_i^t)$ represented as:

$$w_i^{t^*} = \underset{w_i^t}{\operatorname{argmin}} L(w_i^t).$$

The local model will be updated as $w_i^{t+1} = w_i^{t^*}$ and sent to a server for aggregation. For model aggregation, there are three approaches: max pooling (MP), average pooling (AP) and Concatenation (CC) [31]. Since AP could reduce noisy inputs effect on end devices, it is adopted by *FederatedAveraging* (FedAvg) algorithm [24], which combines received local stochastic gradient (SGD) update w_i^{t+1} from each worker on the server side and performs model averaging:

$$w_G^{t+1} = \frac{1}{N}\sum_{1}^{N}(w_i^{t+1}).$$

Finally, the server updates global parameters w_G^{t+1} and sends them back to workers for next round of the FL task.

B. Related Wrok

1) Consensus Protocols in Blockchain: The Nakamoto blockchain [25] uses a Proof-of-Work (PoW) consensus protocol to ensure good scalability and probabilistic finality at the cost of a low throughput and a high energy assumption. To reduce energy consumption in PoW, Ouroboros [14] was proposed in 2017, which is the first Proof-of-Stake (PoS) based blockchain architecture for cryptocurrency Cardano. PoS relies on the distribution of token ownership to simulate a verifiable random function to solve a puzzle problem. Such a process of efficient "virtual mining" manner makes PoS miners only consume limited computational resources for mining new blocks. Unlike PoW that requires high demand of computation and PoS that needs monetary deposit stakes, our PoC based block generation in microchain only depends on credits of validators.

Compared to PoW and PoS blockchains, Practical Byzantine Fault Tolerance (PBFT) [7] based blockchain networks offer excellent performance and a deterministic finality but demonstrate limited scalability. Therefore, adopting BFT style chain finality to PoS style consensus protocols provides a prospective solution to ensure data consistency and immediate finality. Tendermint [16] uses a round-robin fashion BFT consensus process to finalize a block. Decoupling stake value from the BFT voting process was a novel scheme in Tendermint to achieve the consensus goal given probabilistic PoS style block generation. However, Tendermint requires a pre-configured set of validators as the consensus committee, so that it introduces committee security concerns.

To address the fixed set of validators in conventional BFT based blockchain networks, Algorand [10] utilizes verifiable random functions (VRFs) to randomly elect a committee in a private and non-interactive way. However, randomness of each VRF vocation can be biased by an adversary [39]. To improve the scalability of random committee election in a decentralized network, DFINITY [11] uses a random beacon scheme to enable regularly active committee changes. The decentralized random beacon acts as a VRF to put random stream values for the leader selection and ranking. Our microchain adopts a VRF-based sorting algorithm to achieve random committee selection. However, randomness of VRFs is guaranteed by a Publicly Verifiable Secret Sharing (PVSS) based randomness mechanism [28], [30].

Similar to DFINITY, Casper [6] introduces a lightweight chain finality layer on top of a block proposal mechanism, like PoW and PoS. Given an ever-growing *block tree* generated by executing a block proposal protocol among a fixed set of validators, Casper introduces an efficient voting-based process to commit a direct ancestor block of the finalized parent block as a *checkpoint*. As a result, only a unique checkpoint block path is selected from checkpoint tree as the finalized chain. Unlike Casper which is a PoS-based finality system overlaying an existing PoW blockchain, our microchain adopts a voting-based chain finality to resolve the forks by probabilistic block generation.

2) Blockchain-based DML: DeepChain [32] relies on blockchain-based incentive mechanism and cryptographic primitives to ensure privacy-preserving distributed deep learning at the edge networks. Similarly, BlockFL [15] uses a decentralized PoW blockchain network to enable verifiable and rewardable exchanging the mobile devices' local learning model updates in FL networks. VFchain [27] provides a verifiable and auditable federated learning framework based on the permissioned blockchain Hyperledger. However, compute-intensive PoW algorithms are not affordable on IoT devices, while PBFT protocol adopted by Hyperledger allows for limited scalability with reduced security due to the predefined set of validators.

FLchain [22] leverages the concept of private channels for enhancing security of FL. The global model training in FLchain is divided into separate private channels, and only channel-associated peers are allowed to submit transactions and execute consensus for maintaining a channel-specific ledger. Each peer in a channel update its own Global Model State Trie, which records local models of current round in the form of Merkle Patricia tree. However, it incurs storage overhead on edge devices and communication cost for block propagation. Unlike FLchain, our microchain only records hashed reference of models on distributed ledger to reduce storage and communication cost by blockchain.

A hierarchical blockchain-enabled FL framework [8] is proposed to guarantee the security and privacy of knowledge sharing in the large scale vehicular networks. The hierarchical blockchain framework consists of one Top Chain (TC) for high-level further learning and multiple Ground Chains (GCs) for low-level primary learning. The consensus process uses a Proof-of-Knowledge (PoK) algorithm, such that a node with the most accurate learning model can propose the candidate block. However, using local data of each node instead of a uniform test data for learning result evaluation cannot ensure agreement in consensus. Compared to PoK, our microchain decouples PoC block generation from FL process to improve efficiency. Moreover, the high-level inter-chain in our μDFL leverages an efficient BFT consensus protocol to guarantee a low latency and a high throughput.

A decentralized, autonomous blockchain-based decentralized FL framework (BFLC) with committee consensus [19] is proposed to guarantee performance and security in model exchange. BFLC leverages a committee consensus mechanism to improve consensus efficiency, and it designs the storage pattern on the chain to reduce storage consumption on the blockchain. Compared to BFLC, our microchain uses bias-resistant randomness generation and VRF based cryptographic sortition to ensure unpredictable committee election. In addition, we implement the prototype to verify the proposed μDFL in a physical cross-devices FL test-bed.

Integrating blockchain into privacy-preserving multi-party ML process, Biscotti [29] is proposed for private and secure privacy-preserving P2P ML system. Biscotti uses a consistent hashing protocol based on the current stakes distribution of network to choose the consensus committee. However, detail consensus algorithm is not explained, and blockchain fork issues is not considered. Unlike Biscotti, our microchain uses a voting based chain finality to solve fork issue in probabilistic block generation. Moreover, μDFL adopts a federated ledger framework [33] that inter-connects multiple fragmented microchains to guarantee scalability in large scale cross-devices FL settings.

III. μDFL: Rationale and System Design

Aiming at a secure-by-design, self-adaptive and partial decentralized network architecture, μDFL enables an efficient, privacy preserving and secure cooperative training under distributed cross-device FL settings [13] in heterogeneous Mobile Edge Computing (MEC) environments. Figure 1

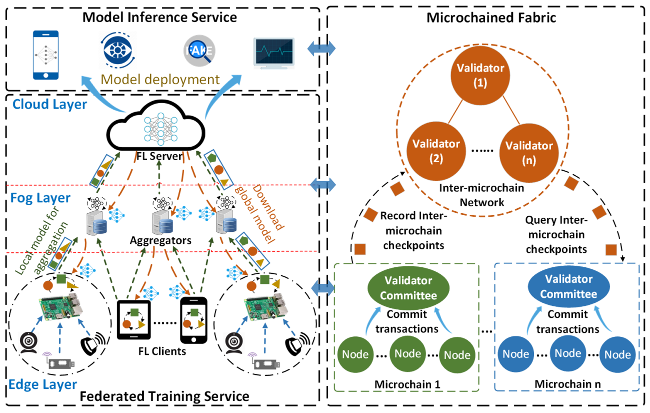

Fig. 1. The μDFL system architecture.

shows the μDFL architecture that consists of *a FL framework* that provides cooperative training on distributed data and global model inference for smart applications, and *a Microchained fabric* that provides decentralized security and privacy-preserving properties for the FL system.

In μDFL, the interaction between FL and microchain can be envisioned analogous to the TCP/IP protocols in the Internet. Microchained fabric acts as the network infrastructure to provide decentralization and security features for FL, like IP functions as a connection-less, best-effort service network layer protocol for the TCP. On the other hand, FL utilizes its capability of model learning and inference to enable high-level intelligent applications as well as the optimization for the microchained network, like TCP provides a connection-oriented, reliable, end-to-end transport layer service to tackle the open issues that IP does not address and meet high quality of service requirements for upper lever applications.

A. Hierarchical Federated Learning Framework

The left part of Fig. 1 demonstrates the hierarchy of FL framework in μDFL, which consists of the cloud, fog and edge computing layers. FL brings an enabling technology for DML model training at MEC network with advantages of highly efficient use of network bandwidth, low latency and local user privacy preservation [20]. Given the life-cycle of an FL-trained model in a FL system, the FL framework is divided into two service layers: federated training service layer and model inference service layer.

1) Federated Training Service: FL clients, the aggregator and the FL server are key players in the model training cycle, including global model propagation (down-stream) and local model aggregation (up-stream). For a FL task, a FL server firstly advertises task specifications including the global model data and training program, among aggregators. The aggregators are deployed on the fog layer as intermediates to transfer task specifications to edge computing devices. All FL clients are allocated at the edge layer to locally compute a model update by executing the training program.

To mitigate model positioning attack in global model propagation, the FL server also generates an authenticator of training program and current global model and records it on the microchain. Both aggregators and FL clients can verify received global model and training program by querying authenticator from the microchain. In model aggregation process, each aggregator firstly collects local model updates from its associated FL clients, then executes the predefined secure MPC aggregation algorithm to generate aggregated model updates. Finally, all aggregated model updates are sent to the FL server where the global model is aggregated.

2) Model Inference Service: Given an analysis on the trained global model, good candidates are selected and deployed on smart applications, such as key board action prediction and anomalous behavior detection based on multiple cameras. However, an adversary can introduce a backdoor by modifying the global model updates in the model deployment process. To defend against model poisoning attacks in model inference time, smart applications can audit the deployed model by verifying its authenticator on microchain, then perform detection tasks if the deployed model is valid.

Hierarchically distributed computing architecture not only provides system scalability for large-scale deep learning tasks based on geographically distributed IoT devices, it also supports flexible management and coordinated central and local decisions among heterogeneous networks and application domains. In addition, those FL players rely on permissioned networks, hence, basic security primitives are provided, such as identity authentication and access control, etc. Furthermore, by interconnecting the FL server, aggregators and FL clients, decentralized and trust-free microchained fabric can protect data and model updates and supports secure MPC functions in FL services.

B. Hybrid Microchained Network Fabric

Following the hierarchical structure of FL, the microchained fabric integrates microchain [34] and a BFT compatible consensus protocol to build a federated blockchain architecture [33]. The right part of Fig. 1 shows the hybrid microchained network fabric, in which an inter-microchain network acts as a hub to interconnect multiple independent microchain consensus networks. The FL server and aggregators could be participants of the inter-microchain network, while FL clients and low-level aggregators are divided into microchain networks, that are associated with training task groups at the network of edge.

The system administrator selects a subset of the nodes from the inter-microchain network as a validator committee. The inter-microchain committee executes an efficient BFT consensus protocol to maintain a public distributed ledger for recording checkpoints of microchains and inter-microchain transactions. The BFT consensus protocol can guarantee liveness and safety by requiring that at most of $f = \lfloor \frac{n-1}{3} \rfloor$ out of total of n participants are dishonest ones [17]. Thus, the ultimate goal of the agreement is achieved if a committee includes $n \geq 3f + 1$ total validators.

Enforcing a BFT consensus on a small-scale inter-microchain committee can reduce messages propagation delay and communication cost, such that scalability and high throughput are achieved during the high-level aggregation over a large-scale FL network.

During the process of a local model training, an aggregator swarms several FL clients to form a microchain consensus network. Given a random committee election mechanism, only a small subset of the nodes are eligible to work as validators in the microchain. In the synchronous network environment, the validator committee executes a novel PoC algorithm to generate new blocks, and uses a VCF strategy to solve fork issue and solidify the private ledger history. Thanks to a lightweight consensus protocol and a small group of validators, microchain aims to guarantee computation and communication efficiency for resource-constrained IoT devices at the edge layer. Meanwhile, a random committee rotation strategy ensures that the robustness and security are not sacrificed because of the small consensus network.

IV. Microchain Consensus Protocol

A. System and Network Model

Table I describes relevant notations used in microchain consensus protocol. Microchain assumes a synchronous network environment, in which operations of processes are coordinated in rounds with bounded delay constraints. Thus, we define *Epoch* $sl_E = \{sl_1, sl2, ..., sl_t..., sl_R\}$ to model a set of sequential time slot sl_t in consensus rounds, where R value is epoch size. To ensure liveness of consensus, the length of time window for a sl_t should be sufficient to guarantee that message transmitted by a sender will be received by its intended recipient within that time window (accounting for local time discrepancies and network delays). The upper bounded delay of system is defined as $T_\triangle$ such that any $sl_t \geq T_\triangle$.

Microchain relies on a permissioned network management, and assumes that the system administrator is a trustworthy oracle to maintain global identity profiles. We adopt a standard asymmetrical algorithm like Rivest–Shamir–Adleman (RSA) for key generation ($RSA.gen$) and digital signature scheme ($RSA.sign$, $RSA.verify$). A pre-defined collision-resistant hash function denotes $\mathcal{H}(\cdot)$ which generates a λ length of hash string $h \in \{0,1\}^\lambda$. During the registration process, a signing-verification keys pair $(sk_i, pk_i) \leftarrow RSA.gen(i)$ along with its account address $a_i = \mathcal{H}(pk_i)$ is generated by a trust Public Key Infrastructure (PKI) and assigned to the authorized user u_i. As security is guaranteed by the permissioned system management, microchain is a partially decentralized blockchain.

TABLE I
Relevant Basic Notation.

Symbol	Descriptions
$\mathcal{N}$	A local model training network including all FL clients
$\mathcal{M}$	An adversary network including malicious FL clients
sl_E	Epoch including sequential time slot sl_t
$T_\triangle$	Upper bounded delay for message propagation
D	Dynasty represents current consensus committee
tx	A transaction broadcasted by the node of network
B	A block proposed by the validator in current Dynasty
$\mathcal{C}$	Distributed ledger maintained by the consensus network

We consider a small-scale local model training network including $u_i \in \mathcal{N}$ FL clients (workers) at an edge network. All malicious workers are denoted by $m_i \in \mathcal{M}$ and their fraction is $f = |\mathcal{M}|/|\mathcal{N}|$. Microchain uses workers' credit stake $0 \leq c_i \leq C_{max}$ for PoC consensus, where C_{max} is the maximum value of credit stake defined by system. A worker's credit stake is calculated periodically according to its contribution to training model and consensus protocol. Each node has an initial credit stake C_{init}=1 when it firstly join network $\mathcal{N}$. All registered workers can be represented as $u_i = (a_i, pk_i, c_i)$, where $0 \leq i \leq n$ and $n = |\mathcal{N}|$.

Each u_i can record byte strings $data \in \{0,1\}^*$ on the distributed ledger by broadcasting time stamped transaction $tx = \{tx\_hash, a_i, T_{stamp}, data, \sigma_i\}$, where $tx\_hash = \mathcal{H}(a_i, T_{stamp}, data)$ and σ_i is a digital signature $RSA.sign_{sk_i}(tx\_hash, pk_i, T_{stamp}, data)$. Microchain elects a subset of $\mathcal{N}$ as *validator* $v_i \in D \subseteq \mathcal{N}$ to work as a consensus committee called *Dynasty* D, and committee size $K = |D|$. In current consensus round sl_t, a validators v_j is allowed to generate a time stamped $Block$ with digital signature, which is represented by $B = (pre\_hash, h, mt\_root, tx\_list, T_{stamp}, a_j, \sigma_j)$, where mt_root is a root hash of Merkle tree of valid transactions tx_list and h is the height of current block in *Distributed Ledger (Blockchain)* $\mathcal{C}$. At the end of consensus round, valid B is append on distributed ledger $\mathcal{C} = B_0 \rightarrow B_1 \rightarrow ... \rightarrow B_{n-1} \rightarrow B_n$, which is a partial order of blocks indexed by strictly sequential increasing h.

B. Consensus Protocol Overview

The distributed ledger structure in microchain is illustrated as the upper part of Fig. 2. The blue nodes represent confirmed blocks while red ones indicate finalized blocks. The genesis block is the root node of blockchain, and each block uses its pre_hash to point to the parent block and extend the chain. The chain height follows a strictly increasing sequence of the finalized blocks (path through red nodes). The head of blockchain (distributed ledger) is a last confirmed block whose parent is finalized and has largest height.

In a microchain network, all nodes use Kademlia DHT (Distributed Hash Table) protocol [23] to synchronize peers in a P2P manner. Thus, a node can connect other nodes by its peering neighbours based on UDP communication. At the initialization stage, a special dynasty, which includes a group of validators specified by the system administrator, acts as an initial committee D_{init} to initialize a blockchain. Each validator creates a genesis block B_0 and sets the local blockchain $\mathcal{C} = B_0$ and $head = B_0$. The initial committee will work as the first dynasty of the system until the election of the next dynasty. The lower part of Fig. 2 demonstrates consensus protocol executed by a validator committee. The key components and work flows are described as follows:

1. Committee Selection: At the beginning of each dynasty's lifetime, an epoch randomness string is used as the seed for committee selection process. The committee formation protocol exploits a Verifiable Random Function (VRF) based cryptographic sortition scheme [10] to randomly choose a

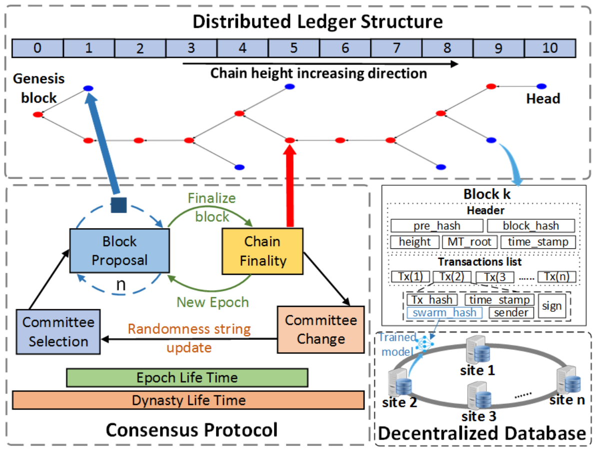

Fig. 2. Illustration of microchain consensus protocol.

subset of validators as a new committee according to their credit weights. The selected committee members D will be added to the current block, which is marked as a beginning block of a new dynasty epoch. The lifetime of a dynasty epoch begins from committee selection and ends after the dynasty is changed by committee change process. The new committee members can reach out each others by their peers to construct a new committee, which executes consensus protocol and randomness string change on a small-scale fully connected P2P network based on TCP communication.

2. *Block Proposal*: The block proposal mechanism leverages a computationally efficient PoC to generate new blocks in each block proposal run. Only validators in the current dynasty are allowed to propose a new block. The probability of proposing a candidate block is associated with the validator's credit distribution in current dynasty. If validator v_j could solve a PoC puzzle difficulty problem defined in Section V, it generates a new block $B_{i+1} = (\mathcal{H}(B_i), h+1, tx\_data, T_{stamp}, a_j, \sigma_j)$ and broadcasts it to current committee members. Each committee member accepts all valid blocks in the current slot, and verifies if blocks meet confirmation requirements. The verified block will be added to local chain $\mathcal{C}$ with $head = B_{i+1}$.

3. *Chain Finality*: At the end of an epoch, the $head$ with epoch height becomes a checkpoint to resolve forks and finalize chain history. A voting-based algorithm commits checkpoint blocks and finalizes those already committed blocks on the main chain. The chain finality ensures that only one path including finalized blocks becomes a main chain, as Fig. 2 highlights. Therefore, the blocks generated in new epoch are only extended on such unique main chain. The chain fork problem is prevent by resolving conflicting checkpoints and finalizing the history of blockchain.

4. *Committee Change*: At the end of the lifetime of a dynasty, all members in the current committee rely on a RandShare mechanism to cooperatively generate a new global randomness string. RandShare is a randomness protocol that is based on Publicly Verifiable Secret Sharing (PVSS) [30], and it ensures unbiasability, unpredictability, and availability in public randomness sharing. The proposed unbiasable and unpredictable public randomness string will be used as seed for the new committee selection process. Finally, the new global randomness string is propagated to other nodes by peers of validators in current committee.

C. Hybrid On-chain and off-chain Storage

Existing blockchain based FL solutions [15], [22], [32] directly record training models or test data on blockchain network, it will greatly increase cost associated with block propagation and chain data storage by edge devices. To mitigate extra overheads incurred by directly saving large raw data into distributed ledger, microchain utilizes a lightweight hybrid on-chain and off-chain storage solution [36]. As lower right part of Fig. 2 shows, block is the basic unit of on-chain storage, which includes header information and the orderly transactions list. Each transaction only saves references in data field $swarm\_hash$ that point to raw data, while original data themselves are stored on the decentralized database (DDB). Because a reference is simply a hash value with fixed length like 32 or 64 bytes, all transactions have almost the same data size even if linked raw data are large training models or test data required different data formats.

Off-chain storage leverages Swarm [4] to build a DDB system, and a site refers to a fog or edge server. The basic unit of storage and retrieval in the Swarm network is *chunks*, which can be accessed at a unique address which is calculated by its hashed content. Swarm implements a Distributed Preimage Archive (DPA) to manage chunks across distributed sites. All Swarm sites have their own base addresses with the same size as the chunk hash, and the site(s) closest to the address of a chunk do not only serve information about the content but actually host the data [4]. All sites in the Swarm network use Kademlia DHT protocol [23], which synchronizes chunks in a P2P manner, to ensure data persistence and redundancy.

V. PoC-based Block Proposal Mechanism

Given a certain period of sliding window for local model training, each validator in committee collects transactions sent by workers. Then, all valid transactions are buffered into a local transactions pool TX. Essentially, the PoC-based block proposal mechanism follows principles of chain based PoS and simulates virtual mining by pseudorandomly assigning block proposal rights to validators. The block generation relies on a slot leader selection process which is associated with credit distribution $\mathcal{D}$ of the current dynasty. The credit distribution is defined as following:

***Definition** 1:* Credit Distribution - is represented as $\mathcal{D} = \{p_1, p_2, ..., p_K\}$, where $p_i = \frac{c_i}{\sum_{j=1}^{K} c_j}$.

For each consensus round $sl_t \in e$, a random slot leader selection procedure determines if a validator v_i is allowed to propose a new block given the probability p_i which is proportional to its weight of credit c_i in current dynasty.

A. Block Generation Algorithm

To become a slot leader who is qualified to generate a new block, a validator must show its proof by solving a puzzle problem. Unlike PoW that utilizes a brute-force manner to find a nonce to meet the uniform target difficulty, each validator v_i simply computes proof based on the chain head block and its credit c_i. The target difficulty that is associated with its credit weight p_i determines if proof is valid or not. The PoC puzzle problem can be formally defined as follows:

Definition 2: Proof-of-Credit - Given an adjustable difficulty condition parameter ξ, the process of PoC puzzle solution aims to verify a solution string $hc = \mathcal{H}(pre\_hash||mt\_root||TX||a_j||c_j)$ where $pre\_hash = \mathcal{H}(head(\mathcal{C}))$, such that the value calculated by taking ξ length lower bits of the hc is smaller than a target value generated by the difficulty condition $d_{cond}(\xi, p_j)$:

$$\mathcal{TB}(hc, \xi) \leq d_{cond}(\xi, p_j) \tag{1}$$

where $\mathcal{TB}(hc, \xi)$ function outputs lower ξ bits of the hash-code hc; and the difficulty condition function $d_{cond}(\cdot, \cdot)$ is denoted as:

$$d_{cond}(\xi, p_j) = (2^{\xi} - 1) \cdot p_j \tag{2}$$

where $d_{cond}(\xi, p_j) \in \{0, 1\}^{\xi}$.

Given the above definitions, the PoC-enabled block generation procedures are presented in Algorithm 1. During the current round slot sl_t, each validator v_i executes $generate\_block()$ function to probably get a candidate block based on its credit stake. If the proof work of new_block is smaller than target value d_{cond}, then a new block is generated and broadcasted to the network. The higher credit weight $v_i.c$, the higher probability that v_i can propose a new block. In block verification process, each validator calls $generate\_block()$ function to determine if the received new_block is valid in the current round slot sl_t with correct proof work. If all conditions are validated, new_block is accepted as confirmed status, and validator updates head of local chain as $head(\mathcal{C}) = B_{i+1}$ accordingly.

Algorithm 1 The PoC-based block generation procedures.

```
1:  procedure: generate_block($v_i$)
2:      $hc \leftarrow \mathcal{H}(head(\mathcal{C}))$
3:      $height \leftarrow head(\mathcal{C}).height + 1$
4:      $mt\_root \leftarrow MTree(v_i.TX)$
5:      $new\_block \leftarrow (hc||mt\_root||v_i.TX||v_i.a||v_i.c||height)$
6:      $C \leftarrow \sum_{i=1}^{K} v_i.c$
7:      $d_{cond} \leftarrow (2^{\xi} - 1)\frac{v_i.c}{C}$
8:      $new\_proof \leftarrow \mathcal{TB}(\mathcal{H}(new\_block), \xi)$
9:      if $new\_proof < d_{cond}$ then
10:         $\sigma \leftarrow$ Sign($new\_block, v_i.a$)
11:         return $(new\_block, \sigma)$
12:     end if
13: procedure: verify_block($new\_block$, $\sigma$)
14:     if Verify_Sign($new\_block, \sigma$) $\neq True$ OR
15:         Verify_TX($new\_block$) $\neq True$ then
16:         return $False$
17:     end if
18:     $hc \leftarrow \mathcal{H}(head(\mathcal{C}))$
19:     if $new\_block.height \neq head(\mathcal{C}).height + 1$ OR
20:         $new\_block.hc \neq hc$ then
21:         return $False$
22:     end if
23:     $C \leftarrow \sum_{i=1}^{K} v_i.c$
24:     $d_{cond} \leftarrow (2^{\xi} - 1)\frac{new\_block.v.c}{C}$
25:     $new\_proof \leftarrow \mathcal{TB}(\mathcal{H}(new\_block), \xi)$
26:     if $new\_proof \geq d_{cond}$ then
27:         return $False$
28:     end if
29:     return $True$
```

B. Chain Extension Rules

The blockchain generation process allows that the probability of block generated by validator v_j is proportional to its credit weight p_j, however, it's hard to achieve conditions that only one block is proposed in the current slot round. Thus, the candidate block number is denoted as $b \in [0, K]$, and the chain extension rules are described as follows:

i) $b = 1$: if there is only one proposed candidate block B_{i+1}, then the block is accepted as confirmed status and updates the chain head as $head(\mathcal{C}) = B_{i+1}$.

ii) $b > 1$: if more than one candidate blocks are proposed, then all blocks are accepted as confirmed status. The $head(\mathcal{C})$ update follows two sub rules:

a) chain head points to a block whose sender has the highest credit c than other blocks' senders; and
b) for the candidate blocks generated by miners who have the same highest credit, the block which has the smallest PoC condition value $\mathcal{TB}(hc, \xi)$ becomes the chain head.

iii) $b = 0$: if no block is proposed at the end of current slot round, block generation follows a spin manner. As validators of current committee can be sorted by account address, we can calculate $ind = height \pmod K$. Thus, validator at rank ind can also propose a candidate block in current round. The chain head update process follows the rule i) $b = 1$.

The rule i) covers a basic scenario to ensure that all blocks are extended on chain head. The rule ii) could handle conflicting chain head update scenario when multiple validators propose their valid blocks during current slot round. The rule iii) ensures the liveness, so that there is at least one block is generated for chain extension even if none of validators is able to propose a new block given PoC algorithm.

VI. Voting-based Chain Finality Mechanism

Owing to the network latency or deliberate attacks by byzantine nodes, it inevitably produces fork issue caused by multiple conflicting blocks with the same ancestor block in a block generation round. Therefore, a chain finality protocol is developed such that committee members can vote for a unique chain path on the block tree, as Fig. 3 shows. We introduce key definitions before explaining workflows and rules in the proposed VCF protocol.

During the R th round of block generation which is also the last round of an epoch, all validators make agreement on a *Checkpoint*, which is the current head block with $(height \mod R) = 0$. Thus, we define *Epoch Height* $H_e(B_i) = \lfloor \frac{B_i.height}{R} \rfloor$ for a block B_i, so that all blocks generated by an epoch have the same H_e. A

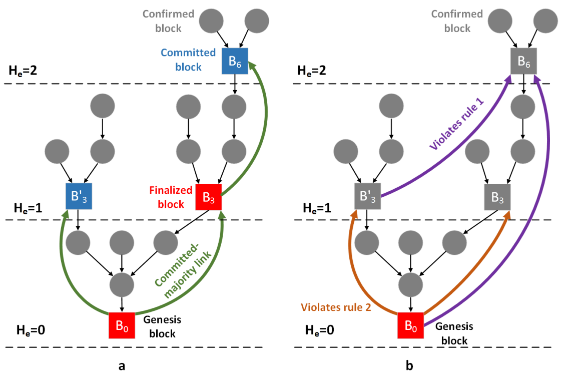

Fig. 3. a) Chain finality protocol based on checkpoint tree; b) Illustration of chain fork resolving rules.

Vote sent by validator v_j at time T_{stamp} is represented as $Vote_j = (vote\_hash, h_{source}, h_{target}, H_e(source), H_e(target), T_{stamp}, a_j, \sigma_j)$, where $source$ is a committed block and $target$ is a checkpoint. Given the voting result, the checkpoint block has following proprieties:

- *Committed-majority Link*: Given an ordered pair of checkpoint blocks (B_s, B_t) denoted as $B_s \rightarrow B_t$, if more than $2/3$ of validators propose vote for it, then relationship $B_s \rightarrow B_t$ is called a committed-majority link. Figure 3a shows that both $B_0 \rightarrow B_3$ and $B_3 \rightarrow B_6$ are committed-majority link.
- *Committed Block*: A checkpoint block B_t is called as a committed block if 1) B_t is a genesis block, or 2) there exists a committed-majority link $B_s \rightarrow B_t$ where B_s is a committed block.
- *Finalized Block*: A checkpoint block B_s is called as a finalized block if 1) B_s is a genesis block, or 2) B_s is a committed block, and there exists a committed-majority link $B_s \rightarrow B_t$, where B_s is parent of B_t in checkpoint tree ($H_e(B_t) = H_e(B_s) + 1$).

A. *Checkpoint Finality Protocol*

Figure 3a illustrates a checkpoint tree with a finalized chain along commit-majority links $B_0 \rightarrow B_3 \rightarrow B_6$. The gray nodes represent all confirmed blocks generated by the block proposal mechanism, while all checkpoint blocks are presented as squire boxes. Assuming that epoch size $R = 3$, all blocks with $(height \mod 3) = 0$ are considered as checkpoint blocks, such as B_0, B_3, $B_3^{'}$ and B_6. The B_0 and B_3 are finalized blocks. While $B_3^{'}$ and B_6 are committed blocks. The chain finality workflow is described as follows:

1. Sending Votes: During current time slot sl_t for chain finality process, a validator v_j checks if $H_e(head(\mathcal{C})) = H_e(B_c) + 1$, where B_c is the last committed block of local chain $\mathcal{C}$. If yes, it prepares $Vote_j$ which votes for a new committed-majority link $B_c \rightarrow head(\mathcal{C})$, and broadcasts signed vote message $Vote_j$ to committee members.

2. Counting Votes: After receiving a vote $Vote_j$, a validator v_i checks if $Vote_j$ is sent by a valid validator v_j ($j \in K$) and σ_j is correctly signed by sender's pk_j. Then, v_i will verify whether received vote violates the chain fork resolving rules, which is defined in section VI-B. If no violation behavior is founded, v_i updates its votes count $vote\_count[h_{source}, h_{target}]$ value by adding 1. Each validator maintains its local votes counting table until current time slot sl_t is expired.

3. Finalizing Checkpoint: A committed-majority link $B_s \rightarrow B_t)$ is only accepted when $vote\_count$ is more than a certain threshold $\mathcal{T} \cdot K$, where $\mathcal{T}$ is a fraction of the committee size K. The chain finality process requires that no less than $2f + 1$ validators are honest given there is no more than f dishonest nodes among N current committee members. Thus, we get $\mathcal{T} \geq \frac{2f+1}{N}$, where $N = 3f + 1$. Given $\mathcal{T} = \frac{2}{3}$, if $vote\_count > \frac{2K}{3}$, the checkpoint block B_t is accepted as a committed block, whereas the committed block B_s is changed to a finalized block.

Given synchronous assumptions that time slot $sl_t \geq \Delta$, all validators receive broadcasted votes from each other by the end of bounded delay. If no more than $\frac{1}{3}$ of the validators violate the chain fork resolving rule, only a committed block will be finalized even if there are conflicting checkpoint blocks with the same epoch height.

B. *Chain Fork Resolving Rule*

If two checkpoint blocks have the same H_e and h_{source}, but they are neither ancestors nor descendants of each other. Then, these sibling blocks are *conflicting* checkpoint. As Fig. 3a shows, B_3 and $B_3^{'}$ are conflicting checkpoint blocks as $H_e = 1$. Figure 3b illustrates vote violation scenarios, and a set of chain fork resolving rules are defined as follows:

***Rule** 1:* Given a vote $Vote_j = (vote\_hash, h_s, h_t, H_e(s), H_e(t), T_{stamp}, pk_j, \sigma_j)$, if $H_e(t) \neq H_e(head(\mathcal{C})$ or $H_e(t) \neq H_e(s) + 1$, the voter j violates rule and $Vote_j$ will be rejected. This rule ensures that vote for committed-majority $s \rightarrow t$ is accepted only if t is the same epoch height as local chain head and s is parent of t in checkpoint tree.

***Rule** 2:* Given two votes $(vote\_hash, h_{s1}, h_{t1}, H_e(s1), H_e(t1), T_{stamp}, pk_j, \sigma_j)$ and $(vote\_hash, h_{s2}, h_{t2}, H_e(s2), H_e(t2), T_{stamp}, pk_j, \sigma_j)$ sent by the same voter j, if $t1 \neq t2$ and $H_e(t1) = H_e(t2)$, the voter j violates rule and all votes from voter will be rejected. This rule prevents one voter from voting for distinct checkpoint blocks with the same epoch height.

***Rule** 3:* Given scenario that vote split at the checkpoint owing to chain finality failure, like no checkpoint can gain more than 2/3 votes, we introduce a trust arbitrator, like the system administrator, who finally decides a checkpoint and ensures liveness of protocol.

VII. Incentives and Punishment Strategies

This section briefly discusses incentives design while leaving further analysis for future work. A honest worker can either receive rewards from local model training process or gain transaction fees from microchain consensus. At the end of a local model training round, aggregator also records a unique test data on the distributed ledger. By querying trained models stored on microchain, each worker u_i has a FL score s_i by using its trained model accuracy based on the test data of current round. Let $S = \{s_1, s_2, ..., s_n\}$ denote FL scores, rewards obtained by a worker u_i denotes as $\gamma_i = \frac{s_i}{\sum_i^n s_i}\mathcal{R}$, where $\mathcal{R}$ is total rewarding fees during current round. At the end of a Dynasty in microchain, transactions fees committed by blocks construct a rewarding fees pool that can be fairly shared with all validators in current consensus committee.

In addition to financial benefits, a honest worker v_i can also deposit one point into its credit stake c_i as the reputation reward. The higher credit stake c, the higher probability that a worker is selected as a microchain committee member. In microchain, credits are neither directly associated with any type of currency nor transferable in any format of transactions. Therefore, all workers are encouraged to behave honestly to gain more benefits by increasing their reputation credits. Moreover, credit stake c of a worker cannot excel a upper-bounded limitation C_{max}. Therefore, an adversary cannot simply accumulate its credit stake to achieve mining centralization in PoS.

We develop punishment strategies to discourage misbehavior, like poisoning models or violating consensus protocol. To increase financial cost for attackers who create Sybil nodes or uses compromised workers to disturb consensus process, each $v_i \in D$ is required to deposit a fix amount of fees to its *security stake* sc_i after committee selection finished. The balance of sc_i will be slashed as punishment if v_i has been found any misbehaving actions in consensus. In addition, a dishonest worker or validator will also loss credit points. As reducing credit stake will decrease the probability of being committee members, it is promising to constraint an adversary's capability.

VIII. Prototype Implementation and Evaluation

A proof-of-concept prototype of μDFL is implemented and tested in a physical network environment. The federated training services are developed using PySyft [41], which is a Python library to develop a secure and private deep learning framework. Microchain is implemented in Python with Flask [2] as web-service framework. We use RSA for asymmetry cryptography and SHA-256 for hash function, which are developed using standard python lib: cryptography [3]. To compare μDFL with PoW based solution [15], we use Ethereum [1] to setup a private PoW blockchain network.

A. Experimental Setup

The prototype of FL part simulates a digital image recognition task by training a CNN model based on MNIST dataset [18]. The 60,000 training samples are randomly split into four equi-sized subsets that are assigned to 12 workers, and 10,000 test samples are held by the server. Table II describes devices used for the experimental study. All workers are deployed on edge devices (Raspberry Pi 4 Model B) and the aggregator is deployed on a fog server (Redbarn HPC). For a private Ethereum network setup, six miners are deployed on six separate desktops that each has Intel(R) Core(TM) 2 Duo CPU E8400 @ 3 GHz and 4 GB of RAM. A test swarm network includes 6 desktops as service sites. Microchain network includes 20 nodes, and each node is hosted on a Raspberry Pi (RPi). All desktops and RPis are connected through a local area network (LAN).

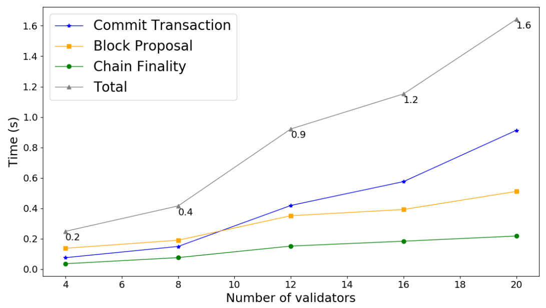

Fig. 4. Latency for an epoch cycle of microchain consensus protocol.

TABLE II
Configuration of Experimental Devices.

Device	Redbarn HPC	Raspberry Pi 4 Model B
CPU	3.4GHz, Core (TM) i7-2600K (8 cores)	1.5GHz, Quad core Cortex-A72 (ARM v8)
Memory	16GB DDR3	4GB SDRAM
Storage	500GB HHD	64GB (microSD card)
OS	Ubuntu 18.04	Raspbian GNU/Linux (Jessie)

In FL simulation test, an aggregator swarms remote workers to train a model on MNIST dataset. The target model is a small neural network with two 5×5 convolutional layers (the first with 20 channels, the second with 50 channels, each followed with a 2×2 max pooling) and two fully connected layers (the first with 500 units, the second with ten units), ReLU activations, and a final softmax output layer (431,080 total parameters). Each worker regularly executes a training epoch with batch size of 64, and the aggregator performs FedAvg model aggregation every 50 training epochs. Therefore, a trained model in our test scenarios is about 1.7 MB.

B. Performance Evaluation

During model aggregation stage, each worker stores its trained model into DDB then launches a transaction to record hash of trained model on microchain. Given multiple complete epoch cycles of consensus protocol within a dynasty, we evaluate performance of running microchain on our FL simulation setting, like consensus latency, data throughput and comparative results. We conducted 100 Monte Carlo test runs and used the average of results for evaluation.

1) Processing Latency: We let system transaction throughput Th_S=100 Transaction Per Second (TPS). Figure 4 presents the network latency for microchain to complete an entire round of consensus protocol given the number of validators varying from 4 to 20. The latency includes the round trip time (RTT) and service processing time on the remote host. As committing a transaction needs $\mathcal{O}(K)$ communication complexity for broadcasting and verification, the latency of commit transaction $\mathcal{T}_{ct}$ is linear scale to committee size K, and it varies from 76 ms to 913 ms. Like commit transaction, the latency of chain finality $\mathcal{T}_{cf}$, requires verification on received K votes from final committee members with complexity $\mathcal{O}(K)$. Thus, $\mathcal{T}_{cf}$ is almost

TABLE III
LATENCY SUMMARY FOR μDFL IN A TRAINING CYCLE.

Operation Stage	**1**	**2**	**3**	**4**	**5**
Processing Time (sec)	30	1.5	1.6	1.4	0.05

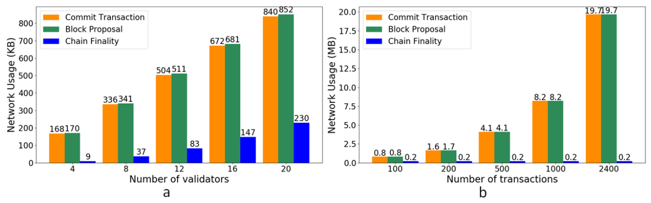

Fig. 5. Comparison of network usage as scaling: a) validators; b) TPS.

scale to K, and it varies from 36 ms to 218 ms. The PoC based block generation algorithm is proportion to validator's credit distribution $\mathcal{D}$ with expectation $E(\mathcal{D})$, therefore, $\mathcal{T}_{bp}$ is scale to complexity $\mathcal{O}(\frac{K^2}{E(\mathcal{D})})$. Given uniform distribution $\mathcal{D}$ in our test with $E(\mathcal{D}) = K$, the $\mathcal{T}_{bp}$ is almost linear scale to the $\mathcal{O}(K)$ with vary from 137 ms to 511 ms.

Table III provides a summary on latency incurred by μDFL when committee size $K = 20$ and Th_S=100. Stage 1 refers to a model training cycle in conventional FL scenario which includes: i) a worker completes 50 local training epochs; ii) network latency by sending local model to the aggregator; and iii) FedAvg model aggregation. In stage 2, the worker encrypts trained model (1.3 s) and then saves encrypted data into DDB (0.16 s). In stage 3, a worker commits swarm hash of a trained model on microchain. It takes 0.08 s to download model data from DDB and 1.3 s to decrypt data in stage 4. For verification in stage 5, a user queries swarm_hash from microchain and compares with the trained model. It only introduces about 15% extra latency by stage 2-5 as integrating μDFL into a conventional FL scenario.

2) Data Throughput: We evaluated the communication cost of running microchain in μDFL in terms of network usage by message propagation and data throughput. Given $Th_S = 100$ (TPS), Figure 5-a demonstrates data transmission for individual stages of a consensus round as scaling up committee size. Each microchain transaction has fixed size d_{tx}=430 Bytes, and a vote message has fixed size d_{vt}=589 Bytes. Total data transmission of commit transaction denotes as $\mathcal{D}_{ct}$=$d_{tx} \times Th_S \times K$, and it linearly scales to K with fixed Th_S. While a chain finality round requires broadcasting vote messages with total data transmission $\mathcal{D}_{cf}$=$d_{vt} \times K^2$, which scales to K^2. As a block has the fixed header d_{head}=613 Bytes, and we can calculate block size d_B=d_{head}+$d_{tx} \times Th_S$. Assuming an ideal case that only one valid block is generated during an epoch cycle, data transmission of block proposal can be calculated as $\mathcal{D}_{bp}$=$d_B \times K$=$d_{head} \times K$+$d_{tx} \times Th_S \times K$, which is almost scale to K when Th_S is fixed.

Figure 5-b shows network usage by microchain consensus with different system throughput Th_S given a fixed committee size $K = 20$. Scaling up Th_S has no effect on chain finality as $\mathcal{D}_{cf}$ only depends on K. However, both $\mathcal{D}_{ct}$ and $\mathcal{D}_{bp}$ are linearly scale to Th_S. By setting $K = 20$, Table IV provides block size and data throughput with variant Th_S. Given observed results microchain process

TABLE IV
DATA THROUGHPUT VS. TRANSACTIONS.

Th_S **(TPS)**	100	200	500	1000	2400
d_B **(KB)**	42.6	84.6	210.6	420.5	1008.4
$\mathcal{T}_M$ **(s)**	1.4	1.8	2.5	4.9	14.7
Th_D **(MB/s)**	1.2	1.9	3.3	3.4	2.7

TABLE V
COMPARATIVE EVALUATION OF RECORDING HASHED MODEL ON DIFFERENT BLOCKCHAIN NETWORKS.

	Ethereum	Microchain
tx **committed time (s)**	4.6	4.4
tx **rate (tx/s)**	124	227
CPU usage (%)	103	6.9
Memory usage (MB)	1,232	28

latency $\mathcal{T}_M$=$\mathcal{T}_{ct}$+$\mathcal{T}_{bp}$+$\mathcal{T}_{cf}$, we can calculate data throughput $Th_D = \frac{\mathcal{D}_{ct}+\mathcal{D}_{bp}+\mathcal{D}_{cf}}{\mathcal{T}_M}$ (MB/s). Compared to existing FLchain solutions [15], [32] that directly record trained models on blockchain, μDFL only encapsulates a small length hash value of raw data into a transaction. Therefore, it's markedly reduce the network bandwidth requirements of transaction and block propagation in blockchain. Given the maximum block size of 1MB in microchain, increasing Th_S can improve efficiency of consensus protocol, and it implies a theoretical maximum data throughput of 3.4 MB/s as Th_S=1000 (TPS).

C. Comparative Evaluation

We let Th_S=1 (TPS) and evaluate key performance matrices in terms of transaction tx committed time, tx rate and resource usage on validator(miner). Table V provides a comprehensive performance of committing 1 TPS on benchmark blockchain networks. As tx committed time indicates how long a transaction can be finalized in a block, and it can be used to evaluate block confirmation time of a blockchain network. Microchain and Ethereum have almost the same tx committed latency. To evaluate throughput of a blockchain network, we calculate tx rate which indicates how many tx can be recorded per second on distributed ledger given tx committed time. Ethereum block size is confined by gas limitation per block such that each block can store about 571 transactions [36]. tx rate in our private Ethereum network can achieve (571.4/4.6)≈124 tx/s. Given observed maximum data throughput in Table IV, a microchain block can store a maximum of 1000 transactions, thus, tx rate is about (1000/4.4)≈227 tx/s. Unlike Ehtereum that relies on a computation intensive PoW algorithm, our microchain uses a lightweight consensus protocol to achieve efficiency in CPU and memory usage, and it's affordable for training workers deployed on edge platforms.

Table VI presents the comparison between our μDFL and previous works. Unlike existing solutions that lack details on blockchain design or investigations on the impact of integrating blockchain into FL, we illustrate μDFL system architecture along with a lightweight microchain design, and evaluate computation and communication cost, blockchain latency

TABLE VI
COMPARISON AMONG EXISTING SOLUTIONS.

	Consensus	Computation	Communication	Storage	Latency	Incentives	Scalability	Privacy
BlockFL [15]	PoW	High	×	×	×	×	×	×
VFChain [27]	PBFT	LoW	×	×	×	√	×	×
Vehicular BC-FL [8]	Proof-of-Knowledge	×	×	×	×	×	√	√
FLchain [22]	×	×	×	×	×	×	×	√
BFLC [19]	×	×	×	×	×	×	×	×
μDFL	PoC+VCF	Low	√	√	√	√	√	√

and throughput in training process. In μDFL system, each private microchain aims to achieve efficiency and privacy preserving for local training at the edge network, while the inter-microchain guarantees global security and scalability for global model aggregation. Such a hierarchical framework is promising to handle trilemma in mono-blockchain solutions that decentralization, security and scalability cannot perfectly co-exist [40]. Furthermore, hybrid on-chain & off-chain design can reduce communication and storage cost by avoiding directly saving models into transactions, and it also protects sensitive information by only exposing model's references on transparent distributed ledgers.

D. Security Analysis

1) Consensus Security: Assume an adversary is subject to the usual cryptographic hardness assumptions, he/she is aware of neither the private keys of the honest nodes nor the input string x to the VRF function. Therefore, the unpredictability property of the VRF-based randomness string generation allows members of committee be completely random. In PoC process, the probability of a leader being selected is proportional to its credit stake. Hence, the probability of an adversary controlling a block generation round can be represented by:

$$P_{\mathcal{M}} = \sum_{i=\frac{K}{2}+1}^{K} \left(\begin{array}{c} K \\ i \end{array} \right) s^i (1-s)^{K-i}$$

where s is the fraction of credit stake controlled by the adversary. The chain finality requires $n \geq 3f+1$ to make agreement on checkpoints, and we let upper bound $P_{\mathcal{M}} = 0.25$. Thus, a committee size of 20 can protect against an adversary controlling 40% of credit stakes in the committee. In this case, we can let C_{max}=1.5 to ensure that the adversary can not control consensus committee. For chain finality stage, the adversary has at most $m = 1/4$ chance per round to control the checkpoint voting process. As a result, the probability of the adversary controlling n consecutive checkpoint is upper-bounded by $P[X \geq n] = \frac{1}{4^n} < 10^{-\lambda}$. For threshold $\lambda = 6$, the adversary will control at most 10 consecutive chain finality runs.

2) Possible Attacks: In training cycle, workers can record encrypted local models into the DDB and commit references of models as correctness proofs on the distributed ledger. As microchain guarantees auditability, immutability and integrity of data on such a hybrid on-chain & off-chain storage, the aggregators can verify trained models before aggregating them into a global model. It is promising to prevent against potential model poisoning attacks. In reference time, FL servers can save references of global models into the distributed ledger, and then any users can check if the deployed models are altered to prevent against evasion attacks.

As a permissioned network, microchain relies on access control and encryption technology to provide a certain degree of privacy protection for data stored on DDB network. It can prevent against dishonest workers violating privileges to inspect sensitive information in model aggregation. However, our solution cannot guarantee protect against data breach by honest but curious users, like dishonest aggregators or server. Using differential privacy is promising to guarantee data privacy and we leave it for future work.

Regarding blockchain system, the adversary may launch double spending attacks to revert committed transactions on distributed ledger. The chain finality ensures total order and persistence of data, therefore, all honest nodes will work on finalized chain and disregard the double spending transactions. In addition, the adversary can perform selfish-mining attack by withholding blocks until release them strategically to gain extra benefits. Microchain requires that all honest validators only accept blocks generated in current round. As delaying block proposal is also considered as a type of misbehavior such that selfish-mining is unprofitable.

IX. CONCLUSIONS

This paper presents μDFL, a hierarchical and secure-by-design microchained fabric for cross-devices DFL at the edge network. Each fragmented microchain allows for efficient, auditable, and privacy-preserving data sharing in local model training time. A high-level inter-chain network federates microchains to ensure scalability and global security. The experimental results based on a prototype implementation are encouraging. However, several open issues need to be solved before bring μDFL into practical FL scenarios.

Although federated microchains is promising improve performance and scalability, more investigation and test are necessary to evaluate impacts by inter-chain transactions. Thus, our on-going efforts includes validating the proposed μDFL in real-world FL applications and evaluating overall performance and security based on various attack scenarios. Moreover, there are unanswered questions on incentives mechanism that motivates devices to devote their resources (e.g., computation and storage) to model training and consensus running for extra profits. Our future work will use game theory to model incentive design and evaluate its effectiveness and robustness.

Acknowledgement

This work is supported by the U.S. National Science Foundation (NSF) via grant CNS-2141468 and the U.S. Air Force Office of Scientific Research (AFOSR) Dynamic Data and Information Processing Program (DDIP) via grant FA9550-21-1-0229. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the U. S. Air Force.

References

[1] "Ethereum Homestead Documentation," http://www.ethdocs.org/en/latest/index.html.

[2] "Flask: A Pyhon Microframework," http://flask.pocoo.org/.

[3] "pyca/cryptography documentation," https://cryptography.io/.

[4] "Swarm," https://swarm-guide.readthedocs.io/en/latest/index.html.

[5] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečnỳ, S. Mazzocchi, B. McMahan *et al.*, "Towards federated learning at scale: System design," *Proceedings of Machine Learning and Systems*, vol. 1, pp. 374–388, 2019.

[6] V. Buterin and V. Griffith, "Casper the friendly finality gadget," *arXiv preprint arXiv:1710.09437*, 2017.

[7] M. Castro, B. Liskov *et al.*, "Practical byzantine fault tolerance," in *OSDI*, vol. 99, 1999, pp. 173–186.

[8] H. Chai, S. Leng, Y. Chen, and K. Zhang, "A hierarchical blockchain-enabled federated learning algorithm for knowledge sharing in internet of vehicles," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[9] D. Froelicher, J. R. Troncoso-Pastoriza, J. S. Sousa, and J.-P. Hubaux, "Drynx: Decentralized, secure, verifiable system for statistical queries and machine learning on distributed datasets," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3035–3050, 2020.

[10] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies," in *Proceedings of the 26th Symposium on Operating Systems Principles*. ACM, 2017, pp. 51–68.

[11] T. Hanke, M. Movahedi, and D. Williams, "Dfinity technology overview series, consensus system," *arXiv preprint arXiv:1805.04548*, 2018.

[12] A. Imteaj, U. Thakker, S. Wang, J. Li, and M. H. Amini, "A survey on federated learning for resource-constrained iot devices," *IEEE Internet of Things Journal*, 2021.

[13] 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.

[14] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: A provably secure proof-of-stake blockchain protocol," in *Annual International Cryptology Conference*. Springer, 2017, pp. 357–388.

[15] 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.

[16] J. Kwon, "Tendermint: Consensus without mining," *Draft v. 0.6, fall*, vol. 1, p. 11, 2014.

[17] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 4, no. 3, pp. 382–401, 1982.

[18] Y. LeCun, C. Cortes, and C. J. Burges, "The mnist database of handwritten digits, 1998," *URL http://yann. lecun. com/exdb/mnist*, vol. 10, p. 34, 1998.

[19] Y. Li, C. Chen, N. Liu, H. Huang, Z. Zheng, and Q. Yan, "A blockchain-based decentralized federated learning framework with committee consensus," *IEEE Network*, 2020.

[20] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, 2020.

[21] L. Lyu, J. Yu, K. Nandakumar, Y. Li, X. Ma, J. Jin, H. Yu, and K. S. Ng, "Towards fair and privacy-preserving federated deep models," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 11, pp. 2524–2541, 2020.

[22] U. Majeed and C. S. Hong, "Flchain: Federated learning via mec-enabled blockchain network," in *2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS)*. IEEE, 2019, pp. 1–4.

[23] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the xor metric," in *International Workshop on Peer-to-Peer Systems*. Springer, 2002, pp. 53–65.

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

[25] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Manubot, Tech. Rep., 2019.

[26] S. Y. Nikouei, R. Xu, Y. Chen, A. Aved, and E. Blasch, "Decentralized smart surveillance through microservices platform," in *Sensors and Systems for Space Applications XII*, vol. 11017. International Society for Optics and Photonics, 2019, p. 110170K.

[27] 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, 2021.

[28] B. Schoenmakers, "A simple publicly verifiable secret sharing scheme and its application to electronic voting," in *Annual International Cryptology Conference*. Springer, 1999, pp. 148–164.

[29] M. Shayan, C. Fung, C. J. Yoon, and I. Beschastnikh, "Biscotti: A ledger for private and secure peer-to-peer machine learning," *arXiv preprint arXiv:1811.09904*, 2018.

[30] M. Stadler, "Publicly verifiable secret sharing," in *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 1996, pp. 190–199.

[31] S. Teerapittayanon, B. McDanel, and H.-T. Kung, "Distributed deep neural networks over the cloud, the edge and end devices," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2017, pp. 328–339.

[32] 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 Transactions on Dependable and Secure Computing*, 2019.

[33] R. Xu and Y. Chen, "Fed-ddm: A federated ledgers based framework for hierarchical decentralized data marketplaces," in *2021 International Conference on Computer Communications and Networks (ICCCN)*. IEEE, 2021, pp. 1–8.

[34] R. Xu, Y. Chen, and E. Blasch, "Microchain: a light hierarchical consensus protocol for iot system," *Blockchain Applications in IoT: Principles and Practices*, 2021.

[35] R. Xu., Y. Chen, E. Blasch, and G. Chen, "A federated capability-based access control mechanism for internet of things (iots)," in *Sensors and Systems for Space Applications XI*, vol. 10641. International Society for Optics and Photonics, 2018, p. 106410U.

[36] R. Xu, D. Nagothu, and Y. Chen, "Econledger: A proof-of-enf consensus based lightweight distributed ledger for iovt networks," *Future Internet*, vol. 13, no. 10, p. 248, 2021.

[37] R. Xu, S. Y. Nikouei, Y. Chen, E. Blasch, and A. Aved, "Blendmas: A blockchain-enabled decentralized microservices architecture for smart public safety," in *The 2019 IEEE International Conference on Blockchain (Blockchain-2019)*. IEEE, 2019, pp. 1–8.

[38] R. Xu, S. Y. Nikouei, D. Nagothu, A. Fitwi, and Y. Chen, "Blendsps: A blockchain-enabled decentralized smart public safety system," *Smart Cities*, vol. 3, no. 3, pp. 928–951, 2020.

[39] M. Zamani, M. Movahedi, and M. Raykova, "Rapidchain: Scaling blockchain via full sharding," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2018, pp. 931–948.

[40] Q. Zhou, H. Huang, Z. Zheng, and J. Bian, "Solutions to scalability of blockchain: A survey," *IEEE Access*, vol. 8, pp. 16 440–16 455, 2020.

[41] A. Ziller, A. Trask, A. Lopardo, B. Szymkow, B. Wagner, E. Bluemke, J.-M. Nounahon, J. Passerat-Palmbach, K. Prakash, N. Rose *et al.*, "Pysyft: A library for easy federated learning," in *Federated Learning Systems*. Springer, 2021, pp. 111–139.