FabricFL: Blockchain-in-the-Loop Federated Learning for Trusted Decentralized Systems

Viraaji Mothukuri, Reza M. Parizi, *Senior Member, IEEE*, Seyedamin Pouriyeh, *Associate Member, IEEE*, Ali Dehghantanha, *Senior Member, IEEE*, and Kim-Kwang Raymond Choo, *Senior Member, IEEE*

***Abstract*—Federated learning (FL) enables collaborative training of machine learning (ML) models while preserving user data privacy. Existing FL approaches can potentially facilitate collaborative ML, but ensuring secure trading/sharing of training data is challenging in practice, particularly in the presence of adversarial FL clients. The ongoing security concerns around FL and strict laws on personally identifiable information necessitate the design of a robust and trusted FL framework, for example, using blockchain. Existing blockchain-based solutions are generally not of industrial strength, where limitations include scalability and lack of engagement by participating clients. In this article, blockchain-in-the-loop FL is our proposed approach of intertwining classic FL and Hyperledger Fabric with a gamification component. Our proposed approach is a fusion of secure application integrated to seal and sign-off asynchronous and synchronous collaborative tasks of FL. The enterprise-level blockchain network provides an immutable ledger that can be leveraged at different FL layers to ensure auditable tracing and level-up security in industrial settings. We evaluate our proposed approach with three different datasets to demonstrate the security enhancements that improve the FL process, resulting in a more accurate global ML model to converge with the possible best performance.**

***Index Terms*—Blockchain, decentralized artificial intelligence, federated learning (FL), hyperledger fabric (HLF), privacy-preserving machine learning (ML), security.**

I. Introduction

IN FEDERATED learning (FL), a group of interested parties referred to as FL clients, such as Internet of Things (IoT) devices, collaborate with a common goal of training a machine learning (ML) model by keeping their training data private, i.e., on-premises. FL clients rely on either a consensus algorithm or a central server with an aggregation logic for averaging clients' local training results. In doing so, users can share and utilize the power of data without relying on/paying intermediate data resourcing companies. The associated benefits of using FL include privacy-preserving, cost-efficient, enhanced, and well-trained ML models. Orchestrating the FL environment and training a secure and robust global ML model are critical tasks in configuring FL environment, as poor implementation of these tasks may potentially result in a compromised FL environment breaching privacy legislation (e.g., data leakage).

Studies have also focused on analyzing zero-day attacks and calculated attacks that can impact the classic FL approaches. For example, researchers have pointed out that attackers can carry out attacks and successfully target an unguarded FL environment [1], [2]. Productionalizing FL in realistic scenarios can also be vulnerable to poisoning and evasion attacks, which can either be an attempt to impact the performance of the global ML model or to ex-filtrate private user data. Such attacks make FL the least reliable for trading in secure data, and consequently, FL clients may be reluctant to participate in the global ML model training.

Another disruptive innovation in recent years is blockchain, which has been shown to take the security of transactions to the next level with its immutable ledger and to provide strengthened security in decentralized artificial intelligence applications. Hence, to address the challenges associated with security, trust, and motivate the use of FL, we propose a new approach that integrates FL with Hyperledger Fabric (HLF) [4] enterprise-level blockchain suited for industrial and IoT applications. Our proposed approach (FabricFL) incorporates a permissioned blockchain, which results in add-on security by efficiently abstracting the visibility of FL clients from each other. In other words, our approach builds on the success of blockchain-based decentralized applications, and more specifically, the immutable ledger and secure authentication of transactions in the FL network.

Having access to global ML model parameters and sample training data makes FL an open platform for malicious clients to perform different attacks such as poisoning or backdoor attacks [1], [2]. To efficiently track and avoid unwanted updates to the global ML model, FabricFL provides such feasibility by tracking updates from clients and protect the global ML model from malicious updates. The blockchain part of the proposed approach equipped with a gamified component also enables reward and recognition for major contributors of FL, which keeps up the motivation resulting in active participants during training rounds of FL. FabricFL's user interface provides interactable features that enable straightforward workflow for enrollment and tracking stats in FL.

Manuscript received 10 January 2021; revised 6 April 2021 and 1 October 2021; accepted 28 October 2021. Date of publication 30 November 2021; date of current version 26 August 2022. *(Corresponding author: Kim-Kwang Raymond Choo.)*

Viraaji Mothukuri, Reza M. Parizi, and Seyedamin Pouriyeh are with the College of Computing and Software Engineering, Kennesaw State University, Kennesaw, GA 30144 USA (e-mail: vmothuku@students.kennesaw.edu; rparizi1@kennesaw.edu; spouriye@kennesaw.edu).

Ali Dehghantanha is with the Cyber Science Laboratory, University of Guelph, Guelph, ON N1G 2W1, Canada (e-mail: adehghan@uoguelph.ca).

Kim-Kwang Raymond Choo is with the Department of Information Systems and Cyber Security, University of Texas at San Antonio, San Antonio, TX 78249 USA (e-mail: raymond.choo@fulbrightmail.org).

Digital Object Identifier 10.1109/JSYST.2021.3124513

To demonstrate utility, we evaluated FabricFL using multiple datasets [5]–[7], as well as the performance and security levels of the classic FL. From the findings, one can observe that our proposed approach achieves promising results even in the presence of adversarial FL clients. Additional integration of blockchain also allows us to enhance the security and performance of FL with minimal impact on the total training time. Our integration logic during the orchestration of FL training rounds ensures that only the local ML model learning's results with positive weightage are added to the global ML model. This has led FabricFL to maintain the best possible and consistent accuracy throughout the training rounds, ensuring that the global ML model has the best performance all the time irrespective of unexpected variations during training rounds. Our evaluation results prove that, on average, FabricFL has insignificant inclusion of 0.1 s of additional training time, which is an acceptable tradeoff for gaining a minimum 10% increase in consistent accuracy rate compared to the classic FL. The main contributions of this article include the following:

1) the design of an end-to-end framework for integrating FL environment with an enterprise-level blockchain network suited for ML-based applications.
2) an efficient orchestration method to minimize the adversarial impact of malicious clients on the global ML model of FL, by proactively excluding local ML model updates that are trained on poisoned/fake data.
3) the design of a credibility-based strategy to build and maintain a trusted FL network with active gamified participation, driven by credibility scores, for efficient resource management and engaged contributions toward FL.

The rest of this article is structured as follows. Section II presents the related literature on FL and blockchain. Section III describes our proposed approach and its inner workings. In Section IV, we describe our implementation setup prior to presenting the evaluation results for both security and performance of the proposed approach in Section V. Finally, Section VI concludes this article.

II. Related Work

There are several proposals in the literature that advocate the application of blockchain in an FL environment. For example, Preuveneers *et al.* [8] leveraged multichain, a permissioned and private blockchain, to integrate FL with blockchain technology. Their approach provides two streams: one for local weight updates and the other for averaged model updates. This approach grants deep learning clients write-only to the first stream and read-only from the second. To skip the complicated task of the federated averaging task, Ramanan and Nakayama [9] proposed an aggregation free framework called BAFFLE, where local user devices collect FL-specific training data, collect global ML model from the blockchain network, and compute the average with the global ML model and local data to obtain a new local model.

In the TClearn framework outlined in [10], the local ML model is promoted as a global ML model based on the Federated Byzantine Agreement of clients in the FL network. Based on the trust factor among the clients, three variants of TClearn are proposed by the authors. The benefits of TClearn are feasibility to roll back the changes in case of issues, Off-chain storage to store the global ML model with the TLS network and private keys to enable access to authorized partners.

Research work in [11] focuses on optimizing the network usage issues in the FL network. Participant's local ML model accuracy and participation frequency are considered to select participants during training rounds. Evaluation results support the author's claim that the proposed algorithm works better than the classic FL federated averaging algorithm logic. Data security and latency in the FL network are considered in the research work of [12], where Somy *et al.* propose an additional cloud owner to host partial training data. Research work in [13] proposes to use EOS Blockchain/HLF, Interplanetary File System (IPFS) to record contributions and calculate rewards in the FL environment. Ethereum and PySyft technologies are used in the research work proposed in [14]; the model owner initiates the process by uploading the model and registering the model and training description to orchestrator smart contract. Ethereum, IPFS, Secure Aggregation are used to facilitates audit trials for detecting variation in updates model weights are encrypted by worker node before sharing.

In [15], Weng *et al.* propose the DeepChain framework, which combines Blockchain with cryptographic techniques and proposes deepcoin for reward/payment toward the contribution of FL training rounds. Deepchain components consist of corda deepchain, corapp, and threshold Paillier algorithm for encryption. Another framework called FLChain proposed by Majeed and Hong [16] uses HLF channels for every new global ML model training process and saves the model on the ledger. Each channel saves a global ML model in the form of a Merkle Patricia tree. The POSTER framework proposed in [17] contains the following components: a server to initiate the tasks and share initial model parameters to clients, a group of clients to update commitment toward the task and run local updates on the shared initial model, an aggregator to work on local model updates, and a miner to compute and verify the global ML model. Research work in [18] uses Ethereum, where each user in FL uploads the local ML model to the associated miner, who performs synchronous proof of work on the local ML model updates, and the verified global ML model is uploaded on the blockchain ledger. ML model data poisoning and FL client drop-off during training rounds are addressed in [19].

In summary, it has to be said that existing research proposals on FL and blockchain unnecessarily complicate the use of blockchain and do not fully meet the requirements of an efficient, scalable, and secure approach suited for large-scale industrial applications. Usage of blockchain with FL needs to be done sensibly without overturning the best of blockchain into a cost-imposing component. Much of the current research suggests securing the global ML model on the ledger, which is the least preferred approach as the global ML model gets updated with each training round. That alone can potentially harm the overall performance of the ML algorithm due to the huge size of data being saved and updated on the blockchain. Our approach, FabricFL, demonstrates an efficient and cost-effective way of securing FL with optimal usage of blockchain features.

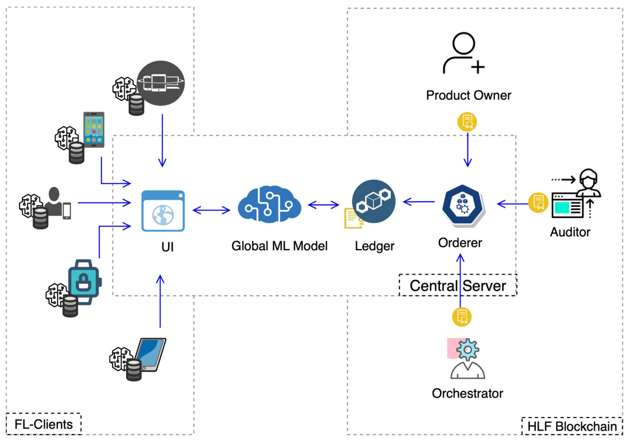


Fig. 1. FabricFL framework: High-level architecture.

III. Proposed Approach

FabricFL provides an end-to-end solution for industrial use cases of decentralized ML applications. Fig. 1 illustrates an overview of FabricFL's architecture. As shown in Fig. 1, FabricFL consists of an FL network with n FL clients each with a training dataset. A global ML model is shared with FL clients. The blockchain components include Orderer, Orchestrator, Ledger, Auditor, and Product Owner (PO), who manage smart contracts and the updates from the FL network. FabricFL UI has two views for hosting launched FL channel details and providing options for FL clients to submit enrollment requests and for authorized POs to submit enrollment requests for new FL channels. Finally, the central server will host the components of blockchain, FabricFL UI, and the global ML model. In the following, we will describe each component in the proposed architecture.

A. *Product Owners*

As the name suggests, POs are the ones who own the FL solutions (FL channels; explained in Section III-B) hosted on the FabricFL framework. They own and manage the lifecycle of the FabricFL framework's FL channels from the development phase to the deployment phase. The owners leverage all the admin privileges to collaboratively construct and maintain the FabricFL framework's FL network. ML solutions hosted on the FabricFL framework are usually either a pretrained model for inference or collaborative FL training of an ML model. POs of an organization can employ an internal team or self-manage the roles of an Auditor and Orchestrator (explained in Sections III-D and III-E, respectively). Fig. 3 illustrates FL channel creation by POs, which is discussed further in Section IV.

B. *FL Channels*

FL channels are the available FL solutions on FabricFL that take in contribution from FL clients and further enhance and share the global ML model. FL solutions of the FabricFL framework hosted by POs are named FL channels. POs pick up a use case of an ML and architect the FL process, which includes choosing or building an ML model network skeleton. POs complete the required development and testing process of the FL channel and deploy it. The deployed FL channel is then launched on FabricFL's UI with a brief description to help FL clients and enroll for preferred FL channels. Examples of FL channels are the text classification ML solution and image data classification with a specific pattern in the dataset.

C. *FL Clients*

In the FabricFL framework, we refer to end-users as FL clients, which are edge devices and can be categorized into two variants: 1) *FL clients contributing toward global ML training* and 2) *FL clients leveraging the hosted ML model through inference*. All the incoming requests from FL clients are handled with add-on security. POs advertise and make FL channels Section III-B available for FL clients to enroll and contribute toward FL training rounds or use the hosted ML model for prediction tasks. FL clients enroll using request templates to opt-in for existing FL channels. FL is well known for its non-independent and identically distributed (non-IID) feature, making it crucial to manage and maintain participants/FL clients data in the FabricFL framework in a secure way. Algorithm 1 summarizes the enrollment of FL-clients into the FabricFL framework. In practical scenarios, it is assumed that enrollment is initiated either individually or in bulk by app/edge device owners who wish to add FL capabilities.

D. *Auditor*

In a multiorganization setup, POs from different organizations collaborate through an auditor role to propose and vote for new FL solutions and further deploy it on FabricFL. The FabricFL framework promotes a decentralized concept with minimal human interaction, so the auditing tasks consider all of the expected loopholes in the process and automate the scripts to exclude them. Algorithm 1 summarizes the role of the Auditor. Following are the auditor's tasks in the FabricFL framework, which can be personalized according to the FL solution being implemented.

1) Manually excludes an FL client from future training rounds if marked in the exclusion list.
2) Review and approve FL solutions/FL channel requests from POs.
3) Review Enrollment requests from FL clients.

In Algorithm 1, requests to launch a new FL channel (R_n^L) are created by an organization in FabricFL and submitted for approval auditors from all the organizations in FabricFL. Once the approved $FL_{Channel}$ is launched on FabricFL, the FL clients interested in joining and contributing toward FL channel submit an enrollmengt request $R_{fl_i}^{FL_{channel}}$, which is reviewed and approved by auditors.

E. *Orchestrator*

Orchestration of training rounds with enrolled FL clients can be either automated or manually triggered by FabricFL UI. During FL training rounds, the Orchestrator asynchronously updates the ledger with the information from the local model weights. A

Algorithm 1: FabricFL Auditing Process.

```
Input: $enroll_{Requests}$
    // $enroll_{Requests}$: List of Enrollment Requests
    submitted for Auditor review
Output: Create/Launch new FL channel or Enrollment
        into existing FL channel of FabricFL
1  $R_n^{FL_{channel}}$  // Enrollment Request to Launch a
   new FL channel from an Organization $n$
2  $R_{fl_i}^{FL_{channel}}$  // Enrollment Request to join FL
   channel from $i^{th}$ FL client
3  while $r_i$ in $enroll_{Requests}$ do
       // Review Requests each request $r_i$ from the
       list $R$
4      if $r_i$ is $R_n^{FL_{channel}}$ then
5          $review(r_i)$
6          $updateLedger(FL_{channel})$
7          $launch(FL_{channel})$
8      if $r_i$ is $R_{fl_i}^{FL_{channel}}$ then
9          $review(r_i)$
10         $FL_{channel}.append(fl_i)$
11         $updateLedger(FL_{channel}(fl_i))$
```

credibility score is evaluated based on updates from FL clients. A credibility score is later used to shortlist the eligibility to push the local model updates into the global ML model. The credibility score is the key deciding factor for the orchestrator to include/exclude the FL client's local ML model updates. During and after FL training rounds, an entry is logged into the ledger with contribution details of FL client's. The contribution factor is calculated using the ML model's positive evaluation metrics, which adds a significant value to the global ML model. Logic that is used to calculate the credibility score is explained as follows:

$$credibilityScore_{fl_n}^{t_i} = mean(eval(m_{fl_n}^{t_i})) + \sum_{t=1}^{i-1} credibilityScore_{fl_n}^{t}.$$

In the equation, fl_n is assumed as the nth FL client, t_i is the current training round, $m_{fl_n}^{t_i}$ is the local ML model of client fl_n for current t_i, and $credibilityScore_{fl_n}^{t_i}$ is credibility score of clientfl_n for current t_i.

Algorithm 2 is used to orchestrate the training rounds with FL clients of a chosen $FL_{Channel}$. Global model parameters M_p are shared/copied to all $fl_{clients}$ from a chosen $FL_{Channel}$. Training rounds are repeated for each FL client fl_n. Federated averaging (presented in Algorithm 3) is called after a preset number (for instance, if preset number k is 10, 3 is called whereafter ($t_i\%10$) is equal to zero. There would be multiples of 10 10, 20, 30, ... training round.) of training rounds for aggregating the local ML model. Algorithm 3 details the logic of averaging process in FabricFL, where historic information of the FL client in ledger along with current training round's credibility score is used to determine the eligibility to consider local ML model updates for the averaging process.

Algorithm 2: Secure Orchestration Process.

```
Input: $fl_{clients}$ $M_p$      // $M_p$:Global ML model
       parameters
Output: Global ML model $Global_{MLmodel}$,
        $credibilityScore_{fl_n}$
1  $fl_{clients} = fl_1, fl_2....fl_n$            // FL clients
2  $fl_{rounds}$               // Total FL training rounds
3  $m_{fl_{client}} = m_{fl_1}, m_{fl_2} ... m_{fl_n}$     // FL client's
   Local ML Models
4  $avg_{flrounds}$    // Average after defined training
   rounds
5  $credibilityScore_{fl_{clients}}$             // Represents
   credibilityscore of Fl clients , initial value
   is assigned as '0'.
6  while $fl_n$ in $fl_{clients}$ do
7      $m_{fl_n}$ = copy($M_p$)
8  while $t_i$ in $fl_{rounds}$ do
9      while $fl_n$ in $fl_{clients}$ do
10         $train(m_{fl_n})$
11     $credibilityScore_{fl_n} = credibilityScore_{fl_n} +$
        $mean(positiveEvaluationMetrics(m_{fl_n}))$
        $updateLedger(fl_n, credibilityScore)$
12     if $t_i$ % $avg_{flrounds}$ == 0 then
13         $Global_{MLmodel} = fabricflAveraging(m_{fl_n},$
            $credibilityScore_{fl_{clients}})$        // Call
            Algorithm 3
14         while $fl_n$ in $fl_{clients}$ do
15             $m_{fl_n} = Global_{MLmodel}$   // send a copy
                of global ML model to each FL client
                in $fl_{clients}$ and make it as current
                local model
                         // loop training rounds.
```

F. Orderer

The Orderer component provided by HLF routes the request from peers to the Ledger. As the name suggests, the Orderer ensures that the order of the transaction is maintained while updating the Ledger and that all the peers are in sync. The Orderer manages the details such as the list of organizations, channels, and approving authorities. As HLF is a permissioned blockchain, it can leverage few consensus protocols that do not require cryptocurrency as an incentive for mining or smart contract/chaincode execution, enabling cost-efficient solutions. We adapt the leader–follower mode of orderer service, which follows etcd[1] based Raft's crash fault-tolerant (CFT) [20] consensus algorithm. CFT requires $\frac{N}{2} + 1$ nodes to reach consensus, enabling scalability and availability of the blockchain network.

G. Smart Contracts

A smart contract provides a gateway to liaise with the blockchain's immutable ledger. In FabricFL, initially, smart contracts for each FL channel are agreed and approved by

[1]https://etcd.io

Algorithm 3: FabricFL Averaging Process.

```
Input: $m_{fl_n}$, $credibilityScore_{fl_n}$
Output: $GlobalMLModel$ $Global_{MLmodel}$
1  $m_{fl_{clients}} = m_{fl_1}, m_{fl_2} ... m_{fl_n}$    // FL client's Local ML Models., where 'n' represents $n_{th}$ FL client
2  $fl_{clients} = fl_1, fl_2 .... fl_n$    // FL clients
3  $Global_{MLmodel}$    // global ML model
4  $credibilityScore_{fl_{clients}} = credibilityScore_{fl_1}, credibilityScore_{fl_2} .... credibilityScore_{fl_n}$
     // credibilityscore of all $fl_{clients}$
5  $weights$ = 1,2...w    // ML model weights
6  $credibilityscore_{minimum} = getPercintile(credibilityScore_{fl_{clients}})$
7  while $m_{fl_n}$ in $m_{fl_{clients}}$ do
8      $credibilityScore_{fl_n} = getcredibilityscoreHistory(fl_n)$    // Get historic information from Ledger of FL client $fl_n$
9      if $credibilityScore_{fl_n} > credibilityscore_{minimum}$ then
10         $modelList.append(m_{fl_n})$
11 while $m_{fl_n}$ in $modelList$ do
12     $Global_{MLmodel}.append(\sum_{weights=1}^{w} (m_{fl_n}))$
13 return $Global_{MLmodel}$
```

auditors. We propose a set of smart contracts as a default setting for each FL channel to leverage an immutable ledger for different transactions of FabricFL. In a distributed environment with multiple end-users, a smart contract serves as the binding factor, which allows the updates of the Ledger/blocks in the blockchain to adhere to the agreed terms and conditions. The privilege to access and invoke the smart contracts is granted to the admin group of FabricFL, and the participants will be unaware of the background details of the smart contract and the ledger.

1) FL Client Smart Contract (FLCSC): It is used to secure the details of FL clients' connection parameters. After the approval of FL client's enrollment, FL clients' details are logged on the Ledger by an auditor. The FLCSC, by default, ensures that the duplicate entries are not logged for the chosen FL channel. During FL training rounds, the Orchestrator retrieves the information of FL clients from the Ledger using the FLCSC. The Auditor and the Orchestrator of the FabricFL network are authorized to invoke the FLCSC to update, retrieve, or delete the records from the Ledger.

2) FL Evaluation Metrics Smart Contract (FLESC): It is used to log evaluation metrics of FL clients' local ML model updates. The Orchestrator calculates the credibility score from each update of the FL client and using historic data logged on the Ledger using the FLESC. Only the Orchestrator has the authority to invoke this smart contract for updating or retrieving.

3) FL Channel Proposal Smart Contract (FLPSC): It is used to secure FL channel information. The auditor updates the approved FL channel information on the Ledger using the FLPSC. The same smart contract periodically gets query and displays the list of available FL channels for enrollment on FabricFL UI. The Orchestrator retrieves the FL channel information from the Ledger for initiating FL training using this smart contract.

4) FL Credibility Score Smart Contract (FLCSSC): It is used to secure the credibility score information of FL clients on the Ledger. The FLCSSC is used to record credibility scores (as explained in Section III-E) during FL training rounds and makes it available before the federated averaging process. Based on the historic information of the credibility score, a decision is made to either include/exclude FL client's updates in the FL averaging process.

H. Ledger

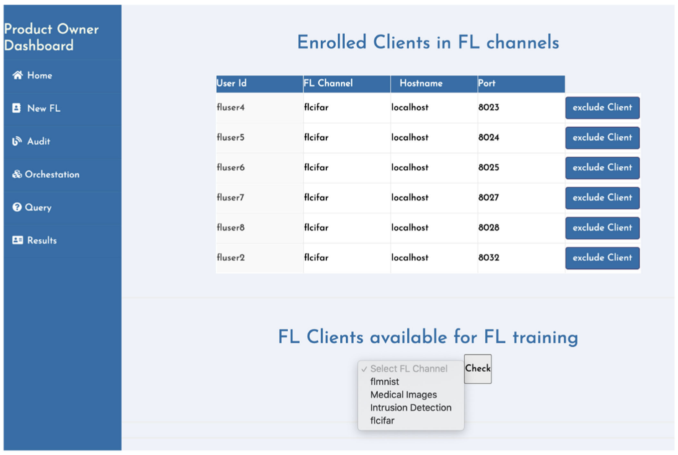
In HLF version of the blockchain, a Ledger is segregated into world state and blockchain. A world state holds the current state or the latest values; each node/peer of an organization holds a separate individual copy of the world state. Any changes to the world state are updated across each peer node. The blockchain part of the Ledger holds the transaction logs, which write each and every change made to the world state. The Ledger can be accessed only through authorized smart contracts.

I. FL Integration With Blockchain

FabricFL comes together as a consortium during the initial setup of a blockchain network usually initiated by an admin group referred to as POs in the FabricFL framework. Digital certificates are created by certification authorities (CAs) for components/actors of organizations to create a unique identity, which defines the access list over the resources of blockchain and ensures privacy and authorized access of the data saved on the blockchain. The membership service provider is a trusted authority that defines and verifies the identity of components/actors from different organizations. Auditors/POs/Orchestrator's are the actors/components of blockchain, which use digitally certified identities provided by CAs. FL client details are managed securely on the Ledger, which are retrieved by the authorized Orchestrator during FL training rounds. The endorsement policy defined in each smart contract enables control over authorization of smart contracts being executed to update the ledger. In HLF, a transaction initiated through a smart contract is segregated into execute–order–validate, connected peers in the organization execute the transaction and endorse it, order the transaction via consensus algorithm, and validate the transaction against endorsement policy before committing the transaction to the Ledger.

J. FabricFL UI

Fig. 2 gives the snapshots of the user interface for the proposed approach, which provides two dashboards: one for the PO and the other for FL clients. For demonstration purposes, we have included Orchestrator and Auditor subroles of PO dashboard page in Fig. 2(a), which contains a list of requests to be approved/rejected by the Auditor, lists FL client details with an option to exclude FL clients from training rounds, initiates a training round for the specific FL channel, monitors logs



Product Owner Dashboard

Home
New FL
Audit
Orchestation
Query
Results

Enrolled Clients in FL channels

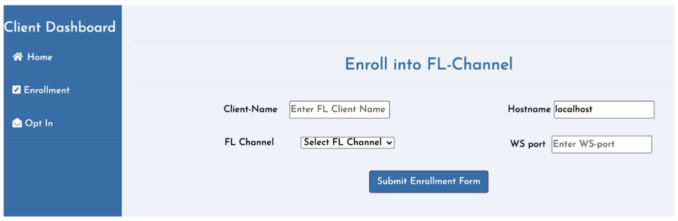
User Id	FL Channel	Hostname	Port	
fluser4	flcifar	localhost	8023	exclude Client
fluser5	flcifar	localhost	8024	exclude Client
fluser6	flcifar	localhost	8025	exclude Client
fluser7	flcifar	localhost	8027	exclude Client
fluser8	flcifar	localhost	8028	exclude Client
fluser2	flcifar	localhost	8032	exclude Client

FL Clients available for FL training

✓ Select FL Channel
flmnist
Medical Images
Intrusion Detection
flcifar

Check

(a)



Client Dashboard

Home
Enrollment
Opt In

Enroll into FL-Channel

Client-Name: Enter FL Client Name
Hostname: localhost
FL Channel: Select FL Channel
WS port: Enter WS-port

Submit Enrollment Form

(b)

Fig. 2. Proposed solution FabricFL framework UI. (a) Product owner dashboard. (b) FL client dashboard.

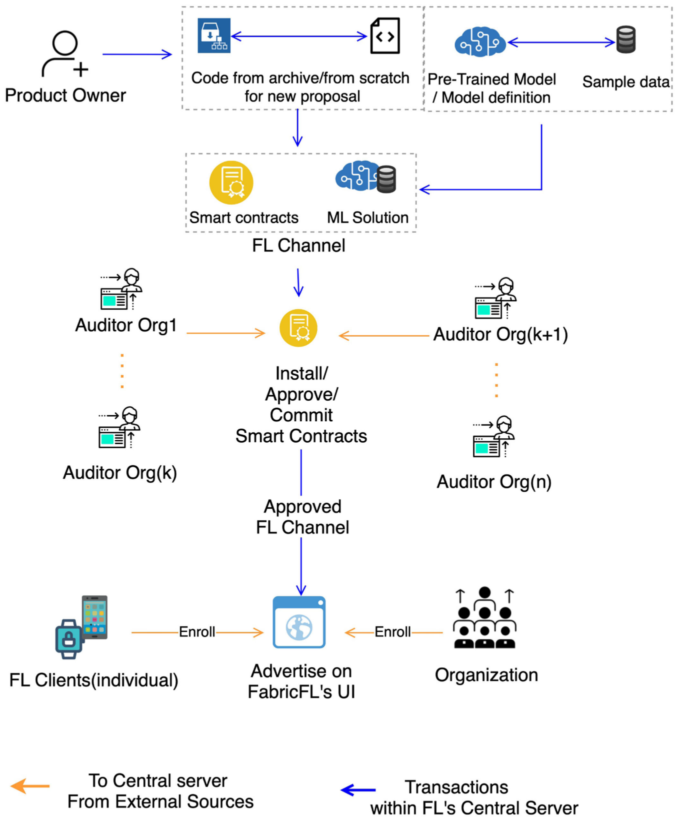

Product Owner
Code from archive/from scratch for new proposal
Pre-Trained Model / Model definition
Sample data
Smart contracts
ML Solution
FL Channel
Auditor Org1
Auditor Org(k+1)
Auditor Org(k)
Auditor Org(n)
Install/ Approve/ Commit Smart Contracts
Approved FL Channel
FL Clients(individual)
Enroll
Advertise on FabricFL's UI
Enroll
Organization
To Central server From External Sources
Transactions within FL's Central Server

Fig. 3. PO: FL channel creation.

of ongoing FL training, and views archived stats of previous training rounds. FL clients' dashboard lists the available FL channels for enrollment. Fig. 2(b) shows the sample enrollment page of FL clients.

K. Central Server

FL's central server, where all the components of FabricFL framework blockchain, Orchestrator, Auditor, POs, and FabricFL UI are deployed, facilitates decentralized capabilities of FL, such as training rounds, management of smart contract's, enrollment, and authorization of POs.

IV. Implementation

The implementation of FabricFL's prototype has been based on a realistic design principle that a group of interested parties from different organizations with expertise in the ML domain work toward either migrating their current centralized ML applications to FL or building new FL solutions. We implement the benefits of blockchain through HLF to provide personalized solutions at multiple stages, starting from authenticated entry to the FL environment to securing information of the FL environment in a secure ledger. HLF is further utilized to store the statistics to establish credibility for the contribution toward the global ML model. The credibility score provides a benchmark to influence and motivate clients of FL to contribute and work toward the common goal of enhancing the global ML model performance. During the orchestration of training rounds in FL, FL clients' credibility score can also help filter out the client's updates. Secure integration with blockchain at different stages of FL provides additional security and assure trust factor among the FL clients. The credibility score tracking enables credibility for the contribution and provides a tracking mechanism for malicious updates. Management of FL clients in secure blockchain ensures restricted/permissioned access to the FabricFL framework. We discuss further details by segregating the framework implementation into the following subtasks and explore each of them in detail.

A. FL Channel Creation

POs own the responsibility of hosting FabricFL's FL channels on a central server and liaise with organizations on the set of predefined rules through smart contracts on the blockchain network. FL clients are the stakeholders who own the key data required to train the ML algorithm. All the prework done by POs is irrelevant and, therefore, not transparent to the FL clients. They act as the end-user/devices that either leverage the global ML model or contribute to the FL training process. Fig. 3 illustrates the implementation of FL channel creation steps listed as follows.

1) *Algorithms and sample data preprocess logic:* For the chosen new FL solution/FL channel, POs develop the required algorithms and the logic to preprocess the sample data.
2) *Choose or create layers of the global ML neural network:* Based on the complexity of the FL channel, POs can either

choose a pretrained ML model or manually build/develop neural network layers, which can efficiently train on the dataset and deliver optimal results. Fake data generators such GANs [21], faker,[2] or a personalized algorithm can be used to recreate extra samples of training data to kick start global ML model training at the central server.

3) *Develop smart contracts for proposed FL channel:* Section III-G lists a minimal set of smart contracts required. POs develop smart contracts specific to each and every FL channel.
4) *Submit FL channel for approval from Auditor's of all the organizations:* The new FL channel proposal requests, which consist of smart contracts, ML solutions, and a description of the new sample dataset, are submitted for review and approval of auditors from all the organizations of FabricFL.
5) *On-board the FL channel on FabricFL's UI:* Once the approval process is completed, the FL channel is on-boarded to FabricFL's UI enabling FL clients to enroll and contribute.
6) *Resource management:* Integration of HLF blockchain and FL environment is initiated with POs advertising hosted FL channel to FL clients enrollment to contribute toward the training process. Algorithm 1 summarizes the on-boarding or launching steps of new FL solutions in FabricFL. POs allocate an auditor to manage the incoming requests for enrollments into the FL channel. Algorithm 1 gives insight into the enrollment of new requests from FL clients. Once the details are logged into the Ledger, the Orchestrator accesses them after secure login and utilizes them during FL training rounds.
7) *Authentication and authorization:* Authentication and authorization of POs are secured using blockchain CAs as the transaction performed through POs is crucial for ensuring the security of the FL environment. The Orchestrator is configured to read the connection details during FL training rounds only after secure authentication and authorization.

B. FL Training

The FL part of FabricFL with "n" clients and a central server is created using PySyft,[3] which is a Python library for secure and private deep learning. FL clients enrolled in FabricFL are connected using PySyft worker API. FL clients willing to participate in the training rounds need to run a script to open a WebSocket port, which includes a tag pointing to the dataset located on the local server. ML model updates are pushed to and from the FL clients using a WebSocket port, which is provided during enrollment of FL clients. FL clients' local updates and the global model are saved on FL's central server. After the averaging process, the global model is shared across the FL clients to keep the network up to date with the latest knowledge and train on top of it. The blockchain part of FabricFL is implemented with HLF [4] v2.2 LTS and the chain code logic (also known as smart contract) is written in Go programming language using Apache CouchDB to store information on the Ledger.

[2]https://faker.readthedocs.io/en/master/
[3]https://github.com/OpenMined/PySyft

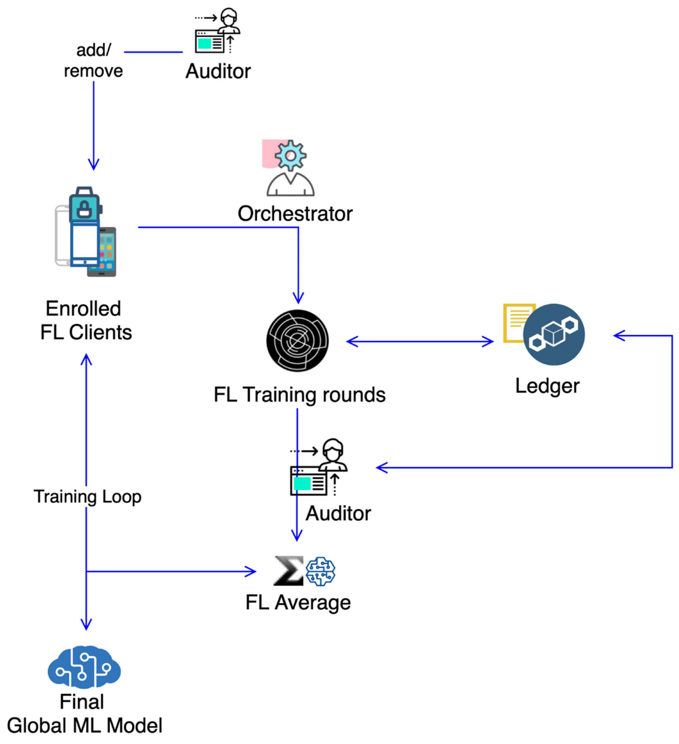


Fig. 4. Orchestrator: FL training.

As described in Section III-E, the Orchestrator oversees the FL training rounds in FabricFL. Based on the FL solutions, training rounds can be preconfigured to engage synchronous/asynchronous interaction with a configured number of clients from the available clients. Once the training rounds are completed, an updated smart contract is triggered to log the model evaluation metrics of each client from the current FL training round. FL training round collects ML model weights from each participating FL client and aggregates them to obtain a global ML model. The generated global ML model holds the knowledge of all the individual FL clients leanings. As illustrated in Fig. 4, the orchestration initiates the FL training rounds with the approved FL clients. Each update from the FL client goes through an auditing process to ensure that only the updates are useful enough and do not contain malicious or harmful information that can impact the performance of the global ML model.

V. Evaluation and Results

In this section, we present the evaluation results of the proposed approach and compare them with the classic FL (hereafter referred to as BasicFL), which is an implementation of FL without blockchain. To demonstrate the broad impact of the work, we have used different deep learning algorithms and datasets from two industrial domains, including IoT medical images and IoT network datasets. We evaluate FabricFL against the basic version of FL (BasicFL) in terms of security and performance.

A. Evaluation Metrics

We use the following measurement metrics to compare our proposed approach with BasicFL in terms of security and performance.

1) *Accuracy, F1-score, and loss:* We calculate the impact of including malicious users in FL training rounds. Security impact is calculated using accuracy, F1-score, and loss evaluation metrics of the global ML model. For calculating the loss function, we use the cross-entropy[4] loss function, which represents the number of instances the ML model predicted wrong labels for the given test dataset. The formulas of each evaluation metrics are

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

$$\text{F1}-\text{score} = \frac{\text{TP}}{\text{TP} + \frac{1}{2} * (\text{FP} + \text{FN})}$$

where TP and TN are predictions that match with the actual labels and FP and FN are predictions that differ from actual labels

$$Loss = CrossEntropyLoss(Predicted_{labels}, Actual_{labels}).$$

2) *Total training time:* It indicates the time taken for each epoch; for BasicFL, it is the total time taken for training rounds and FL averaging. In our proposed model, the total training time includes an additional amount of time taken for the updates to the blockchain ledger during the training rounds. The following equations represent formulas with which we have calculated the training times for BasicFL and FabricFL:

$$\begin{aligned} BasicFL_{Time_{TT}} &= Time[t_r(fl_1) + t_r(fl_2) \ldots t_r(fl_n)] \\ &\quad + Time[c(fl_1) + c(fl_2) \ldots c(fl_n)] \\ &\quad + Time[fl_{average}] \\ FabricFL_{Time_{TT}} &= Time[t_r(fl_1) + t_r(fl_2) \ldots t_r(fl_n)] \\ &\quad + Time[c(fl_1) + c(fl_2) \ldots c(fl_n)] \\ &\quad + Time[fl_{average}] \\ &\quad + Time[c(Ledger)] \end{aligned}$$

where $Time_{TT}$ is the total training time, $t_r(fl_1 \ldots fl_n)$ is the training time for n FL clients, $Time[c(fl_1) \ldots c(fl_n)]$ is the communication time to each FL client, and $Time[c(Ledger)]$ is the time taken to update the Ledger.

B. Datasets and Deep Learning Techniques Used

For image data, we use two datasets. One is a medical dataset [5] of 5863 chest X-rays containing images of normal and viral pneumonia infection. The other dataset is the CIFAR-10 (Canadian Institute for Advanced Research) dataset [7], which is a collection of 6000 (32×32) low-resolution images containing ten different sets of image types. For the classification of images, we use a deep residual network called Resnet [22], which is proven to work well with image classification in ML space. Resnet addresses the vanishing gradient problem of neural networks by implementing building blocks and identity mappings called short-cut connections that carry-on weights from layer n to $n + k$ layers, where k is the layer where the neural network learns well from previous layers.

The third dataset is IoT network traffic data [6] with 115 features. The dataset contains ten categories of attacks and one benign category (each 15 000) extracted from nine IoT devices' network traffic. Different categories of IoT devices are efficiently enhancing the quality of daily mundane tasks, but on the downside, digitalizing sensitive personally identifiable information data makes it crucial to keep track and proactively identify possibilities of intrusion in IoT network hubs. For attack classification, we use a deep neural network with four layers, two hidden layers, and one fully connected layer, and one input layer taking the input perceptron's equal to the number of input features (i.e., 115) in the dataset.

C. Experimental Environment and Setup

To evaluate our proposed work, we simulate the FL network using PySyft, and the central server is configured on the Ubuntu operating system version of 18.04.5 hosted on a lambda GPU server. FabricFL UI is developed with Python Flask. FL clients are built on Virtual/WebSocket instances of PySyft API, and communication is established through WebSocket port. Blockchain functionality is built using HLF LTS version 2.0. We have tested FabricFL on both Docker and Kubernetes, enabling scalability and availability features for updates from POs of different organizations. The feature collision poisoning technique [23] and label flipping poisoning attacks are used to poison a subset of training data. We perform the following set of tasks to simulate the FabricFL framework with the aforementioned environment setup.

1) Setup blockchain network, enroll POs, and generate certificates to enable authentication. Activate smart contracts for the FL channel, create a specific HLF channel for FabricFL channel, and deploy the chain code/smart contracts.
2) Create n FL clients using PySyft's worker instance API. In practical scenarios, FL clients have their own subset of the dataset; to replicate the scenario, we split and distributed the dataset among FL clients to ensure data availability during training rounds.
3) We use the framework's UI to configure FL client details with FL channels.
4) Initiate training rounds using the script, which uses Algorithm 2.

D. Security Analysis

FabricFL, by default, provides defense against the major security concerns observed in BasicFL, such as the free riding, risk of a compromised central server, which may disrupt the training

[4]https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html

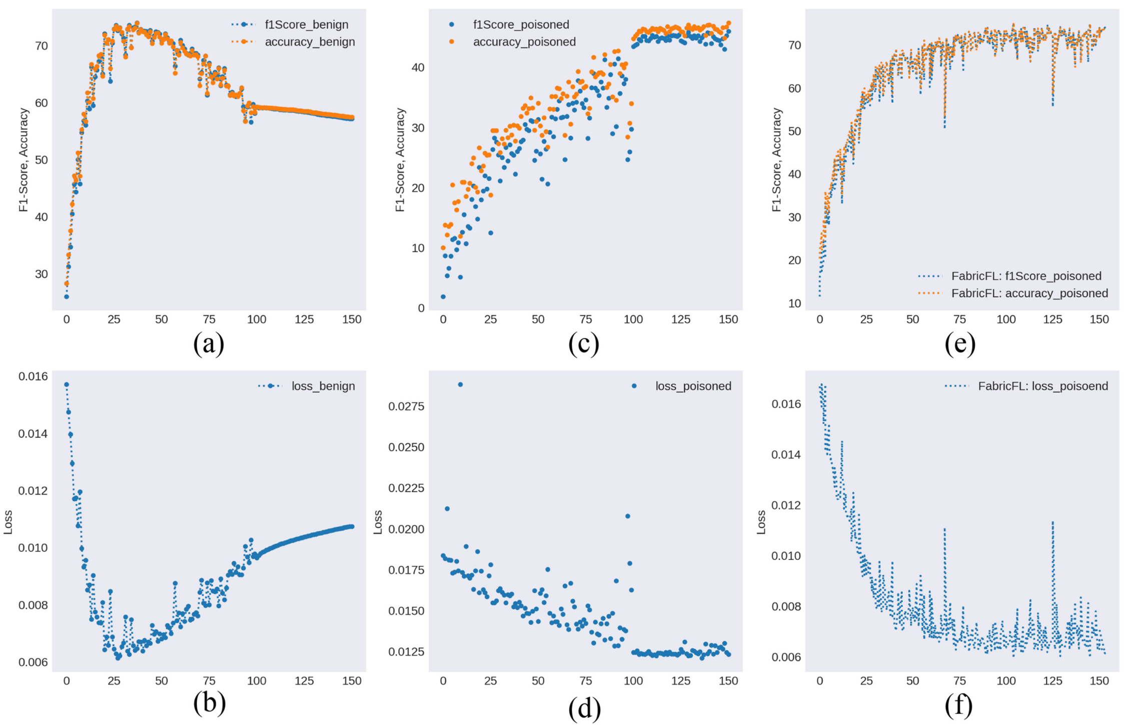


Fig. 5. Security results. Image dataset with malicious clients. (a)–(d) BasicFL. (e) and (f) FabricFL.

process of FL, the risk of global ML model poisoning with fake (adversarial) updates, and unauthorized access to FL clients' connection parameters [24]. The central server of FL plays a key role in orchestrating the training rounds of FL and holds sensitive information of FL clients. In FabricFL, we address the security concern of user information leakage by leveraging secure blockchain. Only the orchestrator of FabricFL has the authority to access FL client connection details. Orchestrator needs a secure authentication key from centralized CAs to gain access to the information stored on the ledger. This secure configuration enables restricted access to information of FL training rounds and offers a clear vision of all the transactions in FL authorized POs.

To evaluate and measure the security enhancements provided by FabricFL, we perform ML training rounds on the chosen datasets and ML algorithms. To replicate a poisoned/fake data attack in FL, we tamper with a set of training data samples and send/tag it to an FL client, which we further refer to as an adversarial FL client. Fig. 5(a) and (b) represents evaluation results of BasicFL with complete real training data from benign FL clients, Fig. 5(c) and (d) shows the results of BasicFL with a combination of real and fake training data from benign and adversarial FL clients, and Fig. 5(e) and (f) demonstrates results of FabricFL with benign and adversarial FL clients.

The evaluation results of BasicFL, as illustrated in Fig. 5(a) and (b), support the fact that BasicFL works well on clean training data, yet it fails to maintain optimal performance values and fluctuates [illustrated in Fig. 5(c) and (d)] with the addition of adversarial FL clients with tampered training data. Positive evaluation metrics, such as accuracy and F1-score, degrade in the basic version of FL with the addition of adversarial FL clients. Fig. 5(c) represents the graph staying at an accuracy rate of 40.45. In Fig. 5(e), positive evaluation results stay consistent and the graph of accuracy and F1-score continue to stay over 70 after the initial phase of training rounds, even with the addition of adversarial client in training rounds, proving that orchestration logic succeeds in eliminating the updates from adversarial FL clients. As discussed in Algorithm 3, the credibility score calculated from the updates of FL clients is used to shortlist the eligible updates for federated averaging of local ML updates. This step secures the global ML model from updates of adversarial FL clients, which eventually helps maintain optimal accuracy values during training rounds. Fig. 6 represents evaluation results with our IoT dataset, where we see a similar pattern of FabricFL resistance toward poisoned updates. Fig. 6(d) represents BasicFL with poisoned data; updates from adversarial FL clients diminish positive evaluation stats of the neural network.

E. Performance Analysis

In performance analysis, we evaluate the performance of our proposed model based on the time taken to complete a specified number of training rounds in comparison to the BasicFL approach. Fig. 7 demonstrates the overall performance of FabricFL with three datasets in terms of accuracy, F1-score, loss, and training time. To further measure the maximum throughput, we evaluate FabricFL and BasicFL with varying number of clients in training rounds, as the transaction to blockchain increases with the number of clients in FL. HLF provides

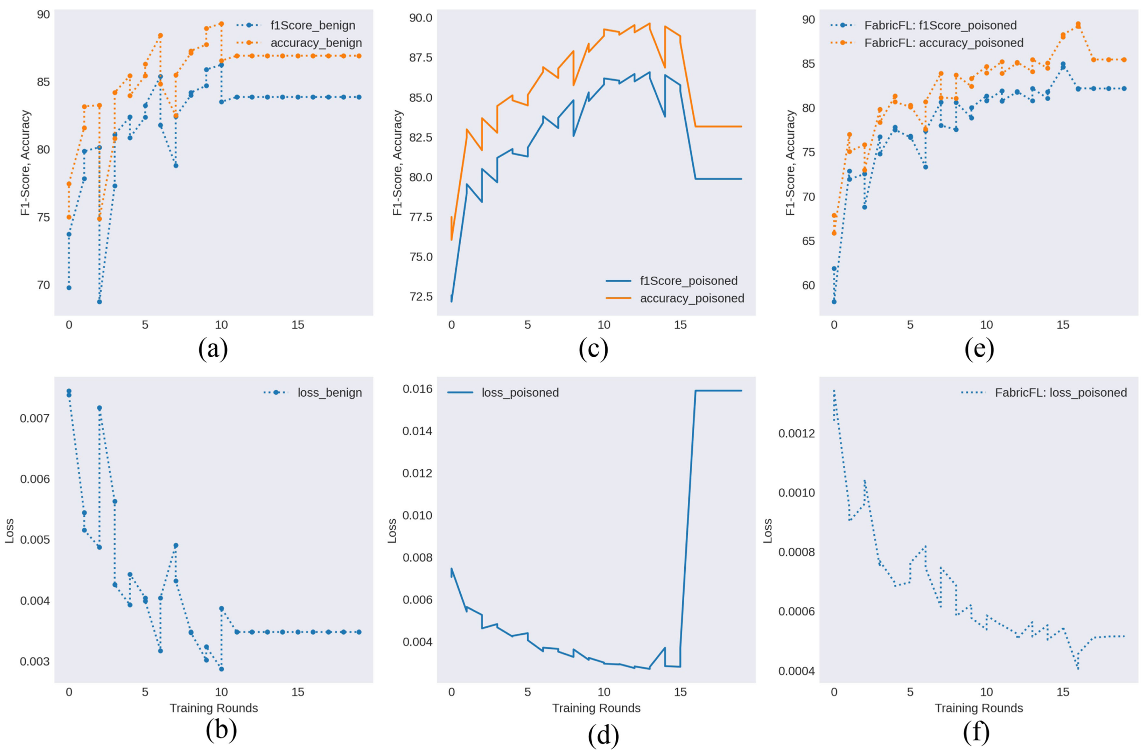


Fig. 6. Security results. Text dataset with malicious clients. (a)–(d) BasicFL. (e) and (f) FabricFL.

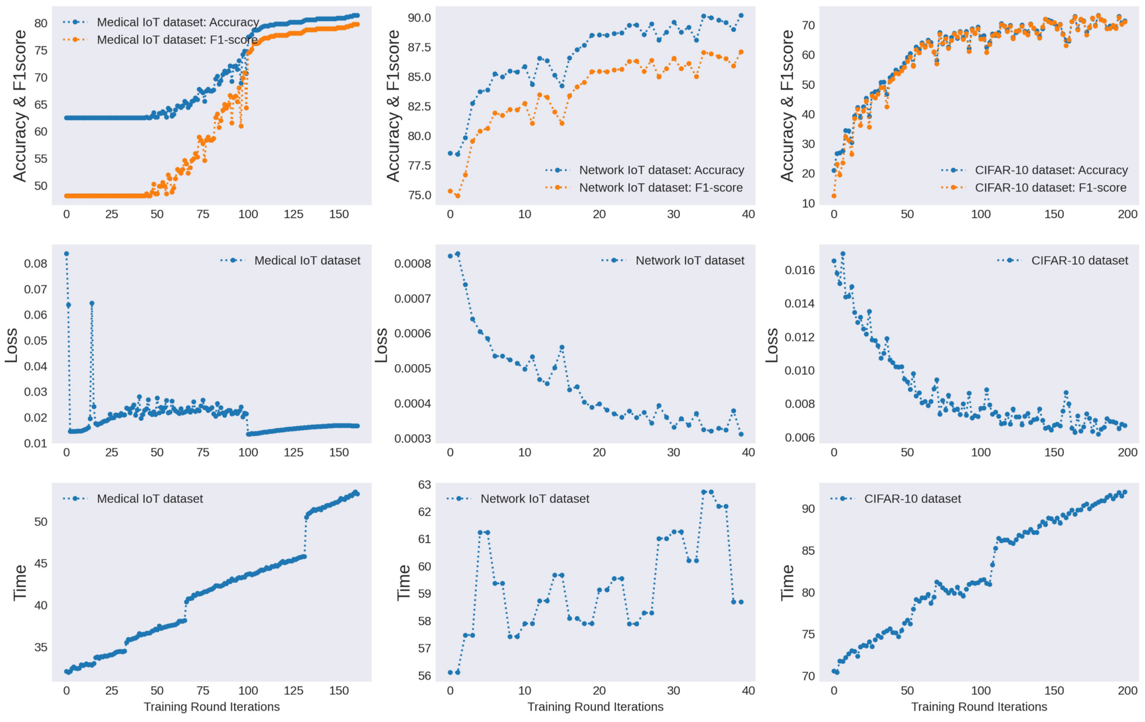


Fig. 7. Overall performance results.

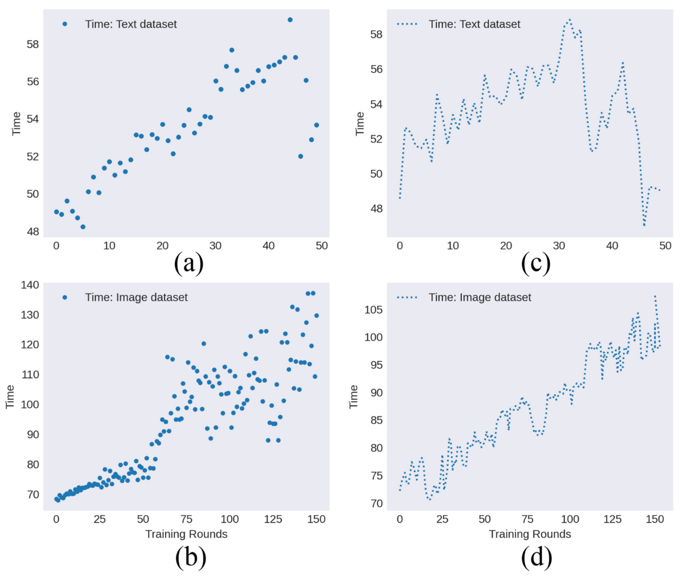


Fig. 8. Total training time. (a) and (b) BasicFL. (c) and (d) FabricFL.

high throughput of 20 000 transactions. We have analyzed the impact on the total training time due to the increased number of transactions to the blockchain. Moreover, Fig. 8 demonstrates the results representing the time taken for each training round in BasicFL versus FabricFL. Overall, the average difference per training round is minimal, which is an acceptable tradeoff with the benefits achieved with the inclusion of blockchain.

In BasicFL, the batch size that determines the number of records processed by the ML algorithm in each training round is one of the hyperparameters of ML, which denotes that the number of records is loaded into each training round. The batch size, along with the number of FL clients in training rounds, determines the total time taken for completion of training rounds. In FabricFL, we have additional calculations and a call to the blockchain network. We capture the total training time included in training rounds, including the additional work done in FabricFL. Comparing the total training time in BasicFL [see Fig. 8(a)] and FabricFL [see Fig. 8(c)], the average difference between FabricFL and BasicFL is 0.17 s, where the minimum and maximum timestamps of a training round of FabricFL 46.93 and 58.82, respectively, whereas BasicFL records minimum and maximum values of 48.22 and 59.28, respectively.

F. Implications

There are a number of implications associated with our proposed approach, designed to upgrade classic FL-based solutions or migrate non-FL-based solutions to blockchain-based FL networks. First, practical possibilities and security benefits are discussed with supporting evaluation results, which serve as a good starting point for the migration of FL/ML solutions to blockchain networks. Second, FabricFL's end-to-end implementation of the FL network also provides a proven strategy to implement an FL-based solution and sets the groundwork for designing the architecture for productionalizable FL solutions.

FabricFL leverages a permissioned and private blockchain that is highly scalable, making its FL network implementable with other public blockchains. The private blockchain network (HLF) of FabricFL can be replaced with a public blockchain with additional changes to implementation strategies, which include logic to enable traceability to track each FL client participating in FL training rounds, extending the accessibility of FabricFL UI to a much larger number of FL clients, and considering offline updates from the FL clients.

However, the approach of FabricFL with a public blockchain may not be efficient in maintaining unique trace and historic information of credibility scores. In this article, we have used HLF to evaluate our proposed approach. In practical scenarios, HLF is a cost-efficient choice as it skips the computationally expensive mining for adding a block to the ledger. FL promotes decentralized learning with a large number of FL clients, and having a limitation to mine a transaction update may impact the overall timings of the FL process.

VI. Conclusion

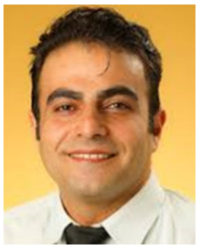
Our research demonstrated the potential of integrating two decentralized technologies and the practical benefits with experimental results. Our proposed approach enhances trust between the FL clients and promotes motivation to actively participate in training rounds. The user interface of the FabricFL framework simplifies the configuration of the FL environment. Within FabricFL, the blockchain part of FabricFL saves the information under a secure impenetrable hood, which makes the information intransparent to FL clients and keeps the non-IID feature of FL intact.

References

[1] Z. Sun, P. Kairouz, A. T. Suresh, and H. B. McMahan, "Can you really backdoor federated learning?," 2019, *arXiv:1911.07963*.
[2] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," in *Proc. Int. Conf. Mach. Learn.* 2019, pp. 634–643.
[3] *Hyperledger Fabric*, 2019. [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/release-2.1/
[4] D. Kermany, K. Zhang, and M. Goldbaum, "Labeled optical coherence tomography (OCT) and chest X-ray images for classification," Mendeley Data, v2," 2018. [Online]. Available: http://dx.doi.org/10.17632/rscbjbr9sj.2
[5] Y. Meidan *et al.*, "N-baIoT-Network-based detection of IoT botnet attacks using deep autoencoders," *IEEE Pervasive Comput.*, vol. 17, no. 3, pp. 12–22, Jul.–Sep. 2018.
[6] *CIFAR-10 Dataset*, 2019. [Online]. Available: https://www.cs.toronto.edu/ kriz/cifar.html
[7] D. Preuveneers, V. Rimmer, I. Tsingenopoulos, J. Spooren, W. Joosen, and E. Ilie-Zudor, "Chained anomaly detection models for federated learning: An intrusion detection case study," *Appl. Sci.*, vol. 8, Dec. 2018, Art. no. 2663.
[8] P. Ramanan and K. Nakayama, "BAFFLE: Blockchain based aggregator free federated learning," in *Proc. IEEE Int. Conf. Blockchain*, 2020, pp. 72–81.
[9] S. Lugan, P. Desbordes, E. Brion, L. X. Ramos Tormo, A. Legay, and B. Macq, "Secure architectures implementing trusted coalitions for blockchained distributed learning (TCLearn)," *IEEE Access*, vol. 7, pp. 181 789–181 799, 2019.
[10] Y. J. Kim and C. S. Hong, "Blockchain-based node-aware dynamic weighting methods for improving federated learning performance," in *Proc. 20th Asia-Pacific Netw. Oper. Manage. Symp.*, 2019, pp. 1–4.
[11] N. B. Somy *et al.*, "Ownership preserving AI market places using blockchain," in *Proc. IEEE Int. Conf. Blockchain*, 2019, pp. 156–165.

[12] I. Martinez, S. Francis, and A. S. Hafid, "Record and reward federated learning contributions with blockchain," in *Proc. Int. Conf. Cyber-Enabled Distrib. Comput. Knowl. Discov.*, 2019, pp. 50–57.
[13] J. Passerat-Palmbach, T. Farnan, R. Miller, M. S. Gross, H. L. Flannery, and B. Gleim, "A blockchain-orchestrated federated learning architecture for healthcare consortia," 2019, *arXiv:1910.12603*.
[14] J. Weng, J. Weng, J. Zhang, M. Li, Y. Zhang, and W. Luo, "DeepChain: Auditable and privacy-preserving deep learning with blockchain-based incentive," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 5, pp. 2438–2455, Sep./Oct. 2021.
[15] U. Majeed and C. S. Hong, "FLchain: Federated learning via MEC-enabled blockchain network," in *Proc. 20th Asia-Pacific Netw. Oper. Manage. Symp.*, 2019, pp. 1–4.
[16] S. Awan, F. Li, B. Luo, and M. Liu, "Poster: A reliable and accountable privacy-preserving federated learning framework using the blockchain," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2019, pp. 2561–2563.
[17] H. Kim, J. Park, M. Bennis, and S.-L. Kim, "Blockchained on-device federated learning," *IEEE Commun. Lett.*, vol. 24, no. 6, pp. 1279–1283, Jun. 2020.
[18] X. Bao, C. Su, Y. Xiong, W. Huang, and Y. Hu, "FLChain: A blockchain for auditable federated learning with trust and incentive," in *Proc. 5th Int. Conf. Big Data Comput. Commun.*, 2019, pp. 151–159.
[19] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *Proc. USENIX Annu. Tech. Conf.*, 2014, pp. 305–319.
[20] I. Goodfellow *et al.*, "Generative Adversarial Nets," in *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, Eds., vol. 27. Red Hook, NY, USA: Curran Associates, 2014, pp. 2672–2680.
[21] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
[22] A. Shafahi *et al.*, "Poison frogs! Targeted clean-label poisoning attacks on neural networks," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 6103–6113.
[23] V. Mothukuri, R. M. Parizi, S. Pouriyeh, Y. Huang, A. Dehghantanha, and G. Srivastava, "A survey on security and privacy of federated learning," *Future Gener. Comput. Syst.*, vol. 115, pp. 619–640, Feb. 2021.

Viraaji Mothukuri is currently a Research Assistant with the College of Computing and Software Engineering, Kennesaw State University, Kennesaw, GA, USA. She has several years of experience in Java and middleware technologies working with WIPRO and JP Morgan companies. She has a professional certification in machine learning to her credit. Her research interests include machine learning, hyperledger fabric, blockchain systems, and decentralized applications.



Reza M. Parizi (Senior Member, IEEE) received the B.Sc. and M.Sc. degrees in computer science in 2005 and 2008, respectively, and the Ph.D. degree in software engineering in 2012.

He is currently the Director of the Decentralized Science Lab, Kennesaw State University (KSU), Kennesaw, GA, USA. He is a consummate artificial intelligence (AI) technologist and cybersecurity researcher with an entrepreneurial spirit. Prior to joining KSU, he was a faculty with the New York Institute of Technology, New York, NY, USA. His research interests include R&D in decentralized AI, cybersecurity, blockchain systems, smart contracts, and emerging issues in the practice of secure software-run world applications.

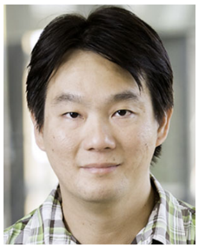
Dr. Parizi is a Senior Member of the IEEE Blockchain Community and the Association for Computing Machinery.

Seyedamin Pouriyeh (Associate Member, IEEE) received the M.Sc. degree in information technology engineering from Shiraz University, Shiraz, Iran, in 2009, and the Ph.D. degree in computer science from the University of Georgia, Athens, GA, USA, in 2018.

He is currently an Assistant Professor of Information Technology with Kennesaw State University, Kennesaw, GA, USA. His research interests include Federated machine learning, blockchain, and cyber security.

Ali Dehghantanha (Senior Member, IEEE) received the Ph.D. degree in security computing.

He has a number of professional certifications, including being a Certified Information Systems Security Professional and Certified Information Security Manager. He is currently the Director of the Cyber Science Lab, University of Guelph, Guelph, ON, Canada. His laboratory is focused on building artificial-intelligence-powered solutions to support cyber threat attribution, cyber threat hunting, and digital forensics tasks in the Internet of Things (IoT), industrial IoT, and Internet of Military of Things environments. He has served for more than a decade in a variety of industrial and academic positions with leading players in cyber-security and artificial intelligence. Prior to joining the University of Guelph, he was a Senior Lecturer with the University of Sheffield, Sheffield, U.K., and an EU Marie-Curie International Incoming Fellow with the University of Salford, Salford, U.K.



Kim-Kwang Raymond Choo (Senior Member, IEEE) received the Ph.D. degree in information security from the Queensland University of Technology, Brisbane, QLD, Australia, in 2006.

He currently holds the Cloud Technology Endowed Professorship with the University of Texas at San Antonio (UTSA), San Antonio, TX, USA.

Dr. Choo is the Founding Co-Editor-in-Chief of *ACM Distributed Ledger Technologies: Research and Practice*, and the Founding Chair of the Technical Committee on Blockchain and Distributed Ledger Technologies of the IEEE Technology and Engineering Management Society. He is an ACM Distinguished Speaker and IEEE Computer Society Distinguished Visitor (for 2021–2023). He was included in Web of Science's Highly Cited Researcher in the field of cross-field in 2020, and in the field of computer science in 2021. He and his team won the Digital Forensics Research Challenge organized by the University of Erlangen-Nuremberg, Erlangen, Germany, in 2015. He is the Recipient of the 2019 IEEE Technical Committee on Scalable Computing Award for Excellence in Scalable Computing (Middle Career Researcher), the 2018 UTSA College of Business Col. Jean Piccione and Lt. Col. Philip Piccione Endowed Research Award for Tenured Faculty, the British Computer Society's 2019 Wilkes Award runner-up, the 2014 Highly Commended Award by the Australia New Zealand Policing Advisory Agency, the Fulbright Scholarship in 2009, the 2008 Australia Day Achievement Medallion, and the British Computer Society's Wilkes Award in 2008. He has also received best paper awards from IEEE SYSTEMS JOURNAL in 2021, IEEE Conference on Dependable and Secure Computing in 2021, *IEEE Consumer Electronics Magazine* for 2020, *Journal of Network and Computer Applications* for 2020, *EURASIP Journal on Wireless Communications and Networking* in 2019, IEEE International Conference on Trust, Security and Privacy in Computing and Communications in 2018, and European Symposium on Research in Computer Security in 2015. He was also the recipient of the Outstanding Paper Award at IEEE International Conference on Blockchain in 2019 and best student paper awards at 2019 International Conference on Information Security and Cryptology and 2005 Australasian Conference on Information Security and Privacy.