TrustFed: A Framework for Fair and Trustworthy Cross-Device Federated Learning in IIoT

Muhammad Habib ur Rehman, *Senior Member, IEEE*, Ahmed Mukhtar Dirir, Khaled Salah, *Senior Member, IEEE*, Ernesto Damiani, *Senior Member, IEEE*, and Davor Svetinovic, *Senior Member, IEEE*

***Abstract*—Cross-device federated learning (CDFL) systems enable fully decentralized training networks whereby each participating device can act as a model-owner and a model-producer. CDFL systems need to ensure fairness, trustworthiness, and high-quality model availability across all the participants in the underlying training networks. This article presents a blockchain-based framework, TrustFed, for CDFL systems to detect the model poisoning attacks, enable fair training settings, and maintain the participating devices' reputation. TrustFed provides fairness by detecting and removing the attackers from the training distributions. It uses blockchain smart contracts to maintain participating devices' reputations to compel the participants in bringing active and honest model contributions. We implemented the TrustFed using a Python-simulated federated learning framework, blockchain smart contracts, and statistical outlier detection techniques. We tested it over the large-scale industrial Internet of things dataset and multiple attack models. We found that TrustFed produces better results regarding multiple aspects compared with the conventional baseline approaches.**

***Index Terms*—Blockchain, fairness, federated learning, industrial Internet of things (IIoT), reputation, security, trust.**

I. Introduction

FEDERATED learning (FL) represents a new class of distributed machine learning techniques whereby the training process is actuated without centralizing the data on cloud data centers [1]. A typical FL process is executed between centralized Internet-enabled servers and the distributed devices and systems connected to them via the Internet [2]. FL lowers the data communication cost by adopting the model-first approach. This approach enables the centralized servers to maintain a global model and push the model parameters to connected devices and systems instead of pulling the large datasets [1]. FL is a privacy-preserving distributed machine learning protocol wherein the devices bootstrap the training process using the global model received from the server and then perform the local model training over their local datasets [3]. The devices then apply the differential privacy preservation techniques (such as homomorphic encryption or multiparty computation) and upload their local model updates to centralized servers. Furthermore, the FL enables a secure model aggregation technique on the centralized servers. These servers aggregate global model updates by performing encrypted computations over reported local model updates without looking into the identity of devices or systems.

The FL model training networks are mainly configured using two approaches, first, cross-dataset FL (CDSFL) systems, whereby the datasets are vertically partitioned across multiple participants in the training network, and second, cross-device FL (CDFL) systems, whereby the datasets are horizontally partitioned across all the devices in the training networks [4]. The CDSFL system always depends on centralized servers (e.g., in cluster/cloud environments) to orchestrate the training services across the participants, aggregate the model updates, and maintain different versions of the centralized model. This type of FL settings always requires a stable communication network to ensure the high availability of FL participants. However, in the CDFL systems, the devices can join or leave the training networks. Also, the CDFL systems enable the personalization of learning models at the fine-grained level. The learning models are initially bootstrapped from a community-trained global model and then gradually personalized at the device level. CDFL delegates more user control over privacy configurations and enables devices to perform opportunistic model training in their desired communities. However, the decentralization property of CDFL not only increases the communication cost in the underlying mesh networks but also boosts up the probability of model poisoning and adversarial attacks [5].

The industrial Internet of things (IIoT) represents a network of intelligent and highly connected industrial devices configured to increase productivity, reduce cost, and enable operational efficiency [6]. IIoT communication networks enable machine-to-machine (M2M) and human-to-machine communication. The concept of CDFL highly correlates with the deployment of FL systems in the M2M communication. The devices and systems

Manuscript received November 1, 2020; revised March 2, 2021 and April 12, 2021; accepted April 19, 2021. Date of publication April 27, 2021; date of current version August 20, 2021. This work was supported by the Center for Cyber-Physical Systems, Khalifa University of Science and Technology, United Arab Emirates. Paper no. TII-20-5048. *(Corresponding author: Davor Svetinovic.)*

The authors are with the Center for Cyber-Physical Systems, Department of Electrical Engineering and Computer Science, Khalifa University of Science and Technology, Abu Dhabi 127788, United Arab Emirates (e-mail: habibcomsats@gmail.com; 100057669@ku.ac.ae; khaled.salah@ku.ac.ae; ernesto.damiani@ku.ac.ae; davor.svetinovic@ku.ac.ae).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TII.2021.3075706.

Digital Object Identifier 10.1109/TII.2021.3075706

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see http://creativecommons.org/licenses/by/4.0/

contribute to the CDFL by providing their data, computation, storage, and communication facilities.

The devices in the CDFL can act as an FL server for configuring model training and aggregating model updates. Similarly, the devices can contribute as model trainers who train the model on their native datasets and communicate the privacy-preserved model updates to the other devices (acting as servers) to update their versions of the trained models. However, the training networks' decoupling increases the communication cost over underlying peer-to-peer (P2P) networks. Similarly, the devices can collude and pollute the training models on some of the devices by initiating adversarial attacks, resulting in unfairly trained low-quality FL models.

Considering the decentralized nature of the datasets, the distribution of devices across the FL systems, and the requirements to train high-quality and population-wide representative models, the issues of fairness and trustworthiness require exceptional attention. The tendency to exhibit unintended, surprising, and adversarial behavior leads to unfairness in FL models. Therefore, FL systems need to ensure fairness at multiple levels [7]. For example, FL systems should meet the individual fairness criterion, where similar devices with similar data and same global model configurations should receive the same results. Similarly, FL models should comply with the criterion of demographic fairness, where the subsets of devices and systems should equally represent all the subsets of overall populations under observations. Likewise, FL systems should meet the criterion of counterfactual fairness, where all devices and systems should be equally provided with the same global model configurations and the same expected output. Also, the equal distribution of rewards among all the honest FL participants must meet these criteria.

This article aims to enable a fully decentralized CDFL system that uses IIoT devices as FL participants. Considering the decentralization of participants, training configurations, and the fairness and trust requirements in the CDFL systems, we aim to use blockchain as a decentralized trusted entity in the CDFL training networks. The main contributions in this article are as follows.

1) We propose a blockchain-enabled framework, TrustFed, for a fully decentralized CDFL system. The proposed framework uses Ethereum blockchain and smart contract technology to enable decentralization and maintain participants' reputation across the CDFL system.
2) We propose a novel protocol for CDFL, which detects the outliers in the training distributions and removes them before aggregating the model updates.
3) We implement and test the TrustFed and the proposed protocol using a real IIoT dataset. The source code of the dataset is available via GitHub [8].
4) We compare and evaluate the results with the existing state-of-the-art baseline approaches.

The rest of this article is organized as follows. Section II presents the related work. Section III presents the proposed framework. Section IV presents the details about the experimental setup. Section V describes the outcomes of the conducted experiments and the comparison with the related work. Finally, Section VI concludes this article.

II. Related Work

The integration of blockchain technologies with FL systems creates opportunities to distribute trust and ensure fairness across FL environments. A few early studies presented the relevant solutions. Researchers in [9] performed system-level integration of blockchain and FL algorithms to achieve improved performance in terms of privacy awareness and communication efficiency. The proposed framework enables smart vehicles to efficiently share FL model updates without centralizing the data using the proof-of-stake-based blockchain-consensus mechanism. Researchers performed a deep investigation of their proposed design considering various blockchain networks performance indicators, such as retransmission limit, block size, block arrival rate, frame size, and end-to-end delay. Their simulation results and mathematical evaluation suggest the favorability of blockchain-integrated FL platforms for mobility-aware FL applications. Their proposed framework will be fully adopted after the considerable improvements in future communication networks, such as 6G and smart vehicles [10]. However, considering the current technology stacks and in the absence of programmable proof-of-stake-based blockchain systems, it is still hard to replicate the experiments in real-world applications.

Researchers in [11] proposed a blockchain integrated framework for the Internet of battle things (IoBT) to enable trustworthy FL applications for sustainable societies. The proposed framework encompasses four layers: data-layer (to collect the IoBT data from external environments, edge-layer (to enable different edge devices to perform ephemeral model training, block mining, and off-chain model storage), fog-layer (to train FL models, aggregate model updates from multiple edge nodes, mine new blocks, and off-chain components to store transient models), and cloud-layer (to aggregate model updates from multiple fog-nodes. Although the proposed framework maintains model histories and learning model updates at multiple levels, trustworthiness and fairness still need to be addressed. A committee consensus-based blockchain-integrated FL framework was proposed to decentralize the model development process and to provide security against malicious attacks on the centralized parameters servers [12]. The proposed framework benefits in terms of maintaining model history and model updates using on-chain decentralized storage networks. It also benefits in terms of scalability and reduced computational cost compared to traditional proof-of-work (PoW)-based blockchain systems. However, it still requires general programmable features to support the wide range of application areas.

Several other researchers focused on blockchain-FL integration. For example, research was done to create the right balance between the amount of data and training cost (i.e., energy) on mobile devices [13]. The optimal block generation rate was also computed to minimize the cost of communication, lower the cost of energy, and reduce the incentive expenditures. Similarly, BlockFlow integrates differential privacy with Ethereum smart contracts, which incentivizes the training participants by offering rewards against their good behavior during the FL training process [14]. Similarly, BlockFL ensures the exchange and verification of shared model updates in the CDFL settings [15]. BlockFL eliminates a centralized

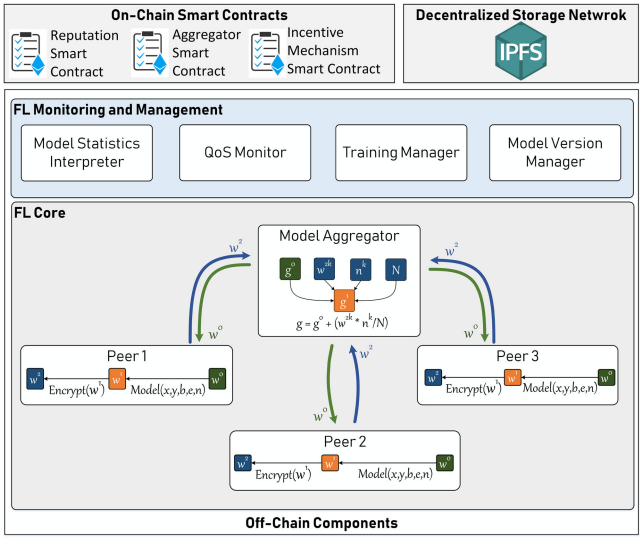


Fig. 1. Framework for trustworthy FL.

parameter server's requirement and uses blockchain technology to complement the centralized server requirements. Another study focused on blockchain-based model sharing between decentralized model training devices using Ethereum smart contracts and IPFS [16]. Also, the system ensures the fairness of training models by performing cross-validation between devices with high-quality models and devices with low-quality models. Researchers in [17] used blockchain-based reputation smart contracts in consortium settings to incentivize the honest and high-quality model producers and develop highly reliable global models. The literature review reveals that despite different proposals and variations, the research on blockchain-integrated trustworthy FL systems is still at its early stages [18].

III. Proposed Framework

This section elaborates on our proposed framework, as depicted in Figs. 1 and 2. The framework capacitates the interactions between on-chain and off-chain FL application components in order to enable FL environment; monitor the FL processes and collect relevant statistics and meet the promised QoS requirements; and deploy public blockchain technologies to perform on-chain FL model aggregation and ensure fairness and trustworthiness using reputation and incentive mechanism smart contracts.

A. Reputation System for FL Systems

1) Centralized FL Systems: The server in centralized FL is responsible for choosing the set of devices for the following FL round. The server selects these devices based on the performance evaluation of all devices. The evaluation is created for each new device and updated after each round. The existence of evaluation records inside the server removes the need for a shared reputation system in centralized FL systems. Although the server has sizeable computational power and is less likely to disconnect, a centralized entity controls all devices' reputations and becomes less desirable and a severe trust issue. It is not a fully transparent process, and no one can stop the server from malicious acts.

2) Decentralized FL Systems: For decentralized FL systems, there is a need for an openly shared record that shows each device's reputation. Blockchain offers a decentralized public ledger that can be used to store the reputation of all devices. New devices can register themselves, check and update the reputation of all other devices. It is a community-driven decentralized reputation system whereby the reputation scores are stored on the public blockchain, and no one can tamper with the reputation scores. Our proposed solution's primary motivation for incorporating blockchain is to enable a fair and trustworthy CDFL system.

3) Relationship Between FL and Blockchain: Blockchain can be integrated with FL applications in many ways (see Section II). However, we aim to deliver a loosely coupled and blockchain-agnostic CDFL system. This approach helps to integrate the proposed framework with any permissioned or permissionless programmable blockchain system. Also, the framework ensures the separation of concerns between CDFL application components and their underlying blockchain network. This design strategy benefits by minimizing the dependence between the FL participants (i.e., the training workers) and the blockchain operators (i.e., mining nodes). Hence, the loose coupling allows the deployment of the CDFL applications across a wide range of public and private blockchain systems.

B. Off-Chain FL Components

Considering many miners in the public blockchain networks (e.g., Ethereum mainnet consists of 8959 nodes as of August 18, 2020 [19]), the replication and synchronization of distributed ledger become the bottlenecks in ensuring the high throughput and scalability for FL applications. Alternately, FL models are required to be continuously trained and evaluated on the new data. Therefore, the proposed framework enables computation-intensive and data-intensive FL components off-chain, which benefit in handling a large number of participating devices and systems in the FL environments. Also, the off-chain FL components facilitate iterative model training on each batch of data and interact only in the case of significant changes in the local model parameters.

The conventional FL process is executed between a set of centralized parameter servers (S_i) and a set of workers W_i (i.e., devices or systems) whereby each participant (P_i) can act as a parameter server to aggregate the model updates from other P_i and as a worker to train the model on their local datasets and, finally, report the model updates to the requesting server. Mathematically, it is denoted as $(P_i \subset S_i) \,||\, (P_i \subset W_i) || (P_i \subset S_i \;\&\&\; P_i \subset W_i)$ where P_i could be an arbitrary subset of either S_i or W_i.

For each training iteration among S_i and W_i, Bonawitz *et al.* [2] elaborated on the execution of the FL process in three stages: selection, configuration, and reporting. At the selection stage, the W_i report or show interest in the model training process and S_i, in turn, selects a subset of reporting W_i;

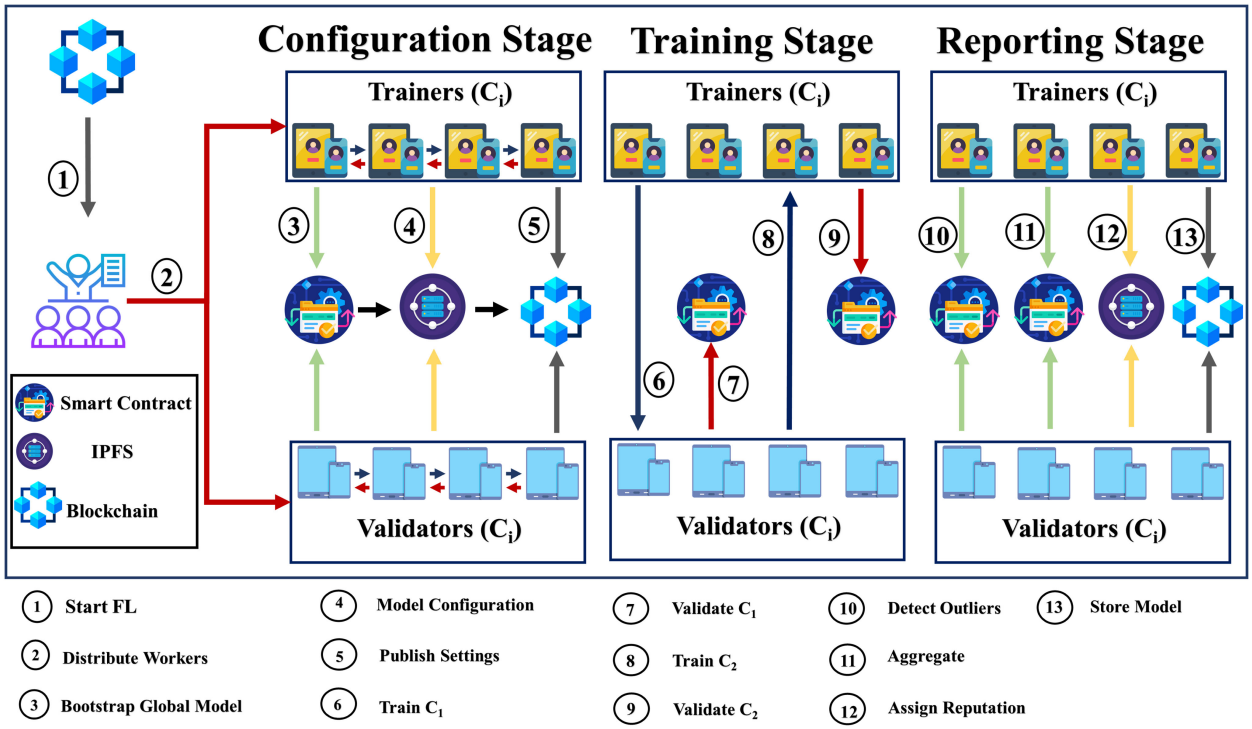


Fig. 2. Cross-device model development using TrustFed framework.

however, the selection criteria may vary based upon the application requirements and contextual information about W_i. S_i communicates the reason for not selecting the arbitrary W_i and asks them to return in the subsequent training rounds if they can meet the specified criteria for selection. At the configuration stage, S_i checks and reads the global model check-points (e.g., initial model weights $[W^0]$) in the persistent cloud storage) and then communicate it to selected W_i. S_i also sends the model configurations, such as learning rates, the number of epochs, and the momentum to selected W_i. At the reporting stage, W_i runs the model configurations on their local datasets DS_i and produces the new model updates (e.g., new model weights W^1). At the reporting stage, the privacy information (e.g., gradient encryption of W^1) is added, and the encrypted model updates are communicated back to S_i, which in turn performs the secure aggregation and writes the global model updates on the persistent centralized storage systems.

Our proposed framework extends the conventional centralized FL framework proposed by Bonawitz *et al.* by enabling a fully decentralized cross-device FL model training in P2P networks. Hence, each P_i can arbitrarily act as S_i, W_i, or both. However, each P_i can independently actuate, participate, or monitor the model training process. The proposed framework enables two types of off-chain components to monitor and manage the FL processes and provide a separate run-time for FL applications.

1) QoS Manager: A plethora of heterogeneous devices and systems in decentralized systems raises quality concerns regarding FL models. For example, late reporting by some devices may result in the loss of some essential model updates or non-independent and identically distributed (IID) datasets on some devices that may cause the overfitted models. Similarly, the asynchronous settings may cause delayed gradient optimization, which results in poor model convergence. Alternately, devices' ability to ensure security and privacy-preserving FL environments can severely affect the quality of produced models. Considering those mentioned earlier and many other application-specific quality issues, the QoS manager publishes the quality requirements of each P_i along with the incentive promises, and it selects the most promising devices considering their reputation on the blockchain network.

2) Training Manager: Despite the selection of reputed devices, the centralization attacks pose serious threats to fair and trustworthy FL systems. The repetitive selection of the same group of reputed devices increases the centralization. It results in a low model convergence rate due to the gradual absence of new data points on the same devices. For example, the infrequent changes in user behaviors result in non-IID datasets, which result in an overfitted trained model. The training manager randomly selects a cohort (i.e., a subset) of reputed devices $C = \{p : p \in P_i, \text{Rep}(p) > \lambda\}$ where λ represents the average reputation score of P_i. Considering the decentralized settings, each device maintains its hyperparameters distribution $\rho(H|\psi)$ over the space of hyperparameters H. Then, at the beginning of each training round t_i, the training manager ensures fairness by performing training in two subrounds. In the first subround, the training manager delegates FL tasks to each C by sending global model hyperparameters h_t sampled from $\rho(H|\psi)$ and initial global model weights W^0. C performs on-device model training and generates new weights W^1 and append the privacy information $Enc(W^1)$ and sends it back to the training manager. The training manager aggregates the model updates and examines the significance of the converged model. In the second round, the training

manager partitions C into two subgroups considering the model updates. C_i with low and average convergence contributions are categorized in the first group C^1 while the remaining C_i are categorized into the second group C^2. Then, the training manager performs the cross-validation of model updates by performing cross-group weight assignment, i.e., ${W_{C^1}}^1 \rightarrow C^2$ and ${W_{C^2}}^1 \rightarrow C^1$ and sends the same model hyperparameters h_t initially sampled from $\rho(H|\psi)$. C_i once again trains the models and produces a new set of model weights W^2. The training manager then retrains the global model by aggregating the model updates considering W^2, and it generates model performance statistics (e.g., training loss, validation loss, training accuracy, validation accuracy, etc.) at the end of each training round.

3) Model Statistics Interpreter: The model statistics interpreter maintains a distribution of reported model statistics to examine the cohortwise model performance, and based on variations in each training subround, it assigns the new reputation scores to each C_i.

4) Model Version Manager: The model version manager cross-examines the model performance and compares it with the initial model performance. Also, it validates the model performance considering the QoS requirements. In the case of QoS-compliant training, it stores the model on the decentralized storage systems, such as IPFS in this research. Since the IPFS enables content-addressed data management instead of IP-addressed data access, therefore it distributes the large-size learning models into even-size chunks of 256 KB. The IPFS enables content addressing by implementing a multihashing mechanism whereby different SHA256 hashing algorithms are used to uniquely identify the model chunks on the underlying P2P storage networks. Also, IPFS provides Merkle-DAG data structures to maintain distributed hash tables, which maintain the information freshness of model updates on the P2P network. However, the absence or low level of data replication on peer nodes causes unavailability and minimal data synchronization. The model version manager publishes the model content addresses to all FL participants via blockchain smart contracts to address these issues and inform all FL participants about model updates.

5) FL Core: The FL-core enables a separate run-time for each FL application whereby it enables two types of components: model aggregator and peers. Since the framework complements fully decentralized cross-device FL training requirements, the traditional FL model training does not need to be controlled from a centralized cloud server. The training manager at each P_i manages the training process. The model aggregator aggregates the resulting W^2, stores the updated model weights on the IPFS, and initiates new blockchain transactions via model aggregator smart contract to update all the FL participants on the network. The peer components communicate with P_i on the P2P network and collect W^i for the model aggregator.

C. On-Chain FL Components

1) Reputation Smart Contract: The main objective of this study is to ensure bidirectional trust among all devices on the training network. Therefore, servers and trainers both should be able to maintain their reputations. This requirement arises because training devices need reputation scores to find more trustworthy servers to get a better quality initial learning model for bootstrapping. Alternately, servers are interested in devices' reputations to find the potential model contributors and maintain their retention rate by offering them incentives to participate in the subsequent training rounds.

1) Trustor: A trustor (T_u) possibly plays a dual role in the system as a trusted user and trusted source. The trust values, such as reputation scores of a trustee (T_p), are publicly made available to all participants in the training network. However, only participating devices acts as trust sources to ensure the legitimacy and quality of collected trust values. For example, a server or a training device may act as a T_u while requesting the reputation scores and provide new reputation ratings after each training round.
2) Trustee: A T_p interacts with the system as a trusted provider and a trusted destination. The platform ensures public availability of aggregated trust values of T_p to all T_u on the network. However, only participating devices can update the destination trust values of each T_p. Any server or training device may act as T_p if they want to maintain their training network's reputations.
3) Off-chain Operations: The accumulation of trust values from several T_u to execute computationally complex iterative functions is done off-chain. The off-chain computations involve normalizing and processing trust values using different statistical methods, AI models, prediction schemes, context acquisition and processing methods, ontology trees, and graph algorithms. For example, we used statistical methods to detect the outliers off-chain and then assign a positive reputation score to all honest participants while decreasing malicious training devices' reputation.
4) On-Chain Reputation Aggregation: The consensus-based aggregation of trust scores for each T_p and storing it on decentralized P2P storage infrastructures is performed using reputation smart contracts, as presented in Algorithm 1. The smart contract is owned by T_p and calculates the reputation of training devices after executing the fair FL protocol presented in Algorithm 2. All devices execute the reputation smart contract in the server mode. The reputation score is incremented or decremented by 100 points based upon the device performance during the fair FL training process.

2) Incentive Mechanism Smart Contract: Considering the decentralized FL nature, all the devices are perceived to be voluntarily participating in the training process. The incentive mechanism smart contract enables the participating devices to publish their QoS requirements: minimum acceptable accuracy, their training budget (i.e., the offered incentives), number of minimum devices required for training, the required training time, and many other application-specific parameters; and pay the rewards using the native cryptocurrency tokens (e.g., ERC-20 on the Ethereum blockchain).

3) FL Aggregator Smart Contract: The aggregator smart contract performs on-chain aggregation of model weights to

Algorithm 1: Reputation Smart Contract for FL Sys-Tems.

Modifier: trustProviders
Input: $\text{EA}_{reputationSC}, EA_{devices}, modelHashes$
Output: $\text{EA}_{repDevices}, RepScore_{devices}$
State: model,
$\text{hash}_{modelAgreement}, numOfMatches, counter$
model = model.download(server)
$\text{hash}_{modelAgreement} = counter$
numOfMatches = $\text{EA}_{devices}.Length/2$
for *every model in modelHashes* **do**
  if *modelAgreement[model]* $\neq 0$ **then**
    modelAgreement[model] $\leftarrow modelAgreement[model] + +$**else**
      modelAgreement[model] $\leftarrow 1$
for *every counter in* $\text{hash}_{modelAgreement}$ **do**
  if $\text{hash}_{modelAgreement}[counter] > numOfMatches$ **then**
    $EA_{repDevices} \leftarrow model[counter]$
    $RepScore_{devices} \leftarrow 100$
    break loop
  Return $EA_{repDevices}, RepScore_{devices}$

create a new version of the updated global model, accessible to all participating devices on the training network. Considering the decentralized FL nature, each device maintains its version of the global model; however, periodic updates of the global model are necessary to maintain a coherent and community-wide trained global model. Hence, any device on the training network can report a new global model update after finding a significant improvement. An aggregating server device generates a unified hash of the updated weight matrix and their corresponding model file, and then it reports the unified hash for on-chain aggregation. The aggregator smart contract updates the weight chain, accessible to all participating devices on the training network.

D. Fair FL Protocol

Considering the trust, fairness, and decentralization requirements, the fair FL training protocol (as presented in Algorithm 2) executes the following steps.

1) Each S_i device publishes QoS parameters (i.e., minimum acceptable accuracy, training budget (i.e., the offered incentives), number of minimum devices required for training, the required training time, and the minimum reputation threshold) on the network.
2) Subscribing W_i on the network show their interest.
3) S_i selects W_i based on their reputation and current promises.
4) Each S_i and W_i manage their native deep neural network architectures.
5) S_i randomly generates two worker subpopulations and form cohorts, i.e., $(C_1, C_2) \in W_i$.
6) S_i sends model configurations (i.e., training parameters) to all selected C_1 in W_i.
7) C_1 performs training and reports back with updated model parameters and accuracy.
8) S_i sends model configurations (i.e., validation parameters) to all selected C_2 in W_i.
9) C_2 performs validation and reports back with updated model parameters and accuracy.
10) C_1 now will play the role of C_2 and C_2 will play the role of C_1, and the training-validation process described before will be repeated.
11) S_i prepares two different accuracy distributions for C_1 and C_2.
12) S_i runs statistical methods (i.e., mean and standard deviation) over both accuracy distributions.
13) S_i detects the outliers by considering the honest W_i, which produces the training and validation accuracy within the control limits [mean $(\mu_i) \pm$ standard deviation (σ_i)]. It removes the remaining W_i.
14) S_i calculates the reputation scores and aggregates top-k highly reputed contributing devices.
15) S_i retrains the global model and generates a new average accuracy.
16) S_i stores a new version of the updated model using IPFS.
17) S_i generates a unified hash by combining input as average accuracy and model hash.
18) S_i stores the unified hash using a smart contract for community-wide on-chain model aggregation.

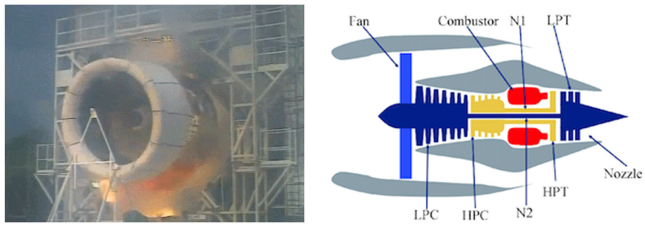


Fig. 3. Simplified diagram of TurboFan engine and its failure mode [20].

IV. Experimental Setup

We adopted the predictive maintenance use-case to test our proposed framework and the fair FL protocol. Conventional predictive maintenance systems collect the data streams from various distributed data sources and process them in centralized computing infrastructures. However, this centralization poses severe threats to various security, privacy, data consistency, data completeness, and data synchronization-related requirements. The *in situ* privacy-preserving secure data analytics components in FL systems enable mitigation of the challenges mentioned earlier. However, the threats of adversarial attacks and malicious model trainers still exist; hence, we mapped the proposed framework to a decentralized predictive maintenance system.

We implemented the proposed framework using the Python programming language. We performed all experiments on the Turbofan Engine Degradation simulation dataset released by NASA [20], which is the baseline dataset used to estimate the remaining useful life of a Turbofan engine (see Fig. 3). The dataset was generated using commercial modular aero propulsion system simulation, which simulates the real flight

Algorithm 2: Fair FL Protocol.

Input: $\text{QoS}_{Parameters}$
Output: gradientMatrix, averageAccuracy
model = model.download(latestModelHash)
initModel = model
trainedModels = []
devices = findDevices($\text{QoS}_{parameters}$)
trainers = selectRandomTrainers(devices.length/2)
validators = getRemainingDevices(devices,trainers)
$t_{acc} = [], v_{acc} = []$
shift = trainers.length **for** *i = 0 to trainers.length* **do**
  $t_{acc}[i], model_i = model.train(trainer_i)$
  trainedModels.push(model_i)
for *j = 0 to validators.length* **do**
  trainedModels.send(validator_j)
  $v_{acc}[j + shift] = validateModels(validator_j)$
trainedModels = []
for *j = 0 to validators.length* **do**
  $t_{acc}[j+shift], model_j = model.train(validator_j)$
  trainedModels.push(model_j)
for *i = 0 to trainers.length* **do**
  trainedModels.send(trainer_i)
  $v_{acc}[i] = validateModels(trainer_i)$
$\text{fair}_{devices}[] = calculate_{fairness}(t_{acc}, v_{acc})$
averageAccuracy = model.aggregate($\text{fair}_{devices}$)
gradientMatrix = model.weights
if *averageAccuracy* $\leq QoS_{Parameters}[reqAccuracy]$ **then**
  Print("%averageAccuracy% is less than $\%QoS_{Parameters}[reqAccuracy]\%$")
  Discard(model)
  model = initModel
  model.store()
else
  initModel = model
  Return *gradientMatrix, averageAccuracy*

Fig. 4. Aggregated model loss with and without outliers in fair data distribution across devices in the FL training network.

Fig. 5. Aggregated model loss with and without outliers in random data distribution across devices in the FL training network.

conditions and collects realistic sensory measurements from the turbofan engine (to measure the temperature, fan and core speed, pressure, fuel flow ratio, bypass ratio, burner fuel-air ratio, and bleed enthalpy). The dataset was prepared with four different flight and sensory configurations for several engines and every cycle of their lifetime. To run the FL experiment, we split the dataset into training and validation subsets for 100 engines to replicate the node behavior in the FL training network.

The number of available IIoT datasets is very small and the simulation results might look simple; however, it is enough to show how fairness can be achieved in a decentralized FL system via the detection of outliers. This dataset is an IID dataset. We used a set of dense layers in the FL model, and integrated the PyTorch framework to simulate the FL environments. The FL model's layered architecture was configured as follows. The input layer takes $input\_size \times 24$ input neurons for each window. Two middle dense layers represent 24×24 neurons, and the output dense layer is mapped on 24×1 neurons. Also, we used ReLU as an activation function at the input, middle, and output layers. Each experiment was run with 10 epochs and the stochastic gradient descent (SGD) optimizer was used for weight adjustments. We tested TrustFed with different worker populations, as presented in Section V. To actuate the fairness strategy, each experiment was performed on two worker subgroups where the workers train and validate the models to simulate the cross-device FL environments. For each iteration, we added random noise to a worker by adding perturbations to input images, and each experiment was performed in two rounds to calculate fairness across worker populations. We also used the solidity programming language and Remix IDE to integrate the blockchain smart contracts in CDFL system. The proposed smart contract was tested and validated using Securify and Remix Native compiler.

V. Results and Discussion

We tested the TrustFed in terms of trained model loss and the ability to detect adversaries with random and fair data distributions across the FL participants in the FL training network. Figs. 4 and 5 present the comparison of TrustFed, with the absence of the outliers, with the baseline FL systems with the present outliers. For example, in our experiment, we used 20 000 sensors' readings data and distributed these data across different workers, starting from a worker population of 6 and ending

Fig. 6. Model loss when each worker model is tested in cross-device settings with fair data distribution.

Fig. 7. Model loss when each worker model is tested in cross-device settings with random data distribution.

with a worker population of 22. The performance comparison of TrustFed with the baseline shows that our proposed scheme always yields better results in terms of the lower loss irrespective of the population size. The better results are due to the TrustFed's ability to identify and remove the outliers.

The model loss, as depicted in Figs. 6 and 7, shows the loss for each worker model when tested in cross-device settings. For

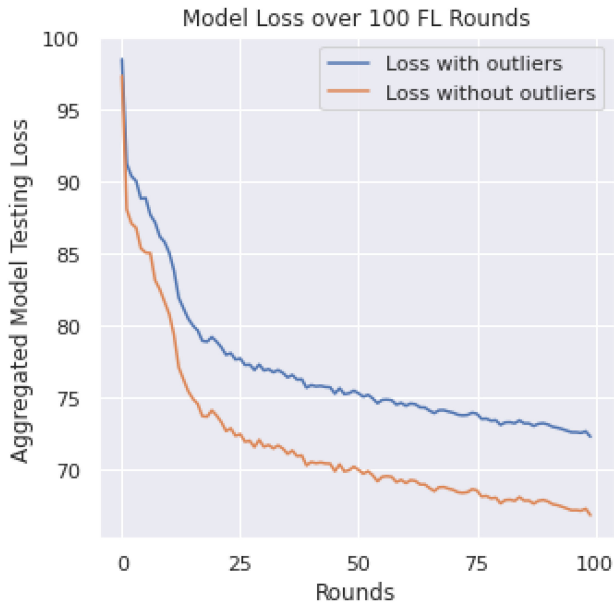


Fig. 8. Aggregated model loss with and without outliers in random data distribution across devices in the FL training network (100 workers).

instance, Worker 6 is reporting a high loss value when tested on the other devices. The other workers are reporting high loss values when tested on the Worker 6 device. This loss is a clear indicator that Worker 6 is not performing well, including the fact that it might be acting maliciously. Thus, Worker 6 should be eliminated from subsequent training rounds. If most workers are honest, then malicious workers can be detected using this graph. The guideline is that if a particular worker trained a model on its device, and this model is not performing well when sent for testing on the other workers' devices, this is a strong indicator that this worker is not adding value to the global model. If the other workers' models are not performing well on this worker's device, this worker should be removed from the subsequent training rounds.

To further test the effect on performance, scalability, and accuracy of the system, we increased the number of workers up to 100. Requesting each worker to test models generated by the other workers might add computation and communication overhead to the proposed system. Fig. 8 shows the aggregated model loss with and without the outliers. As expected, the loss is lower when outliers are removed compared with the situation when all models, including models generated by the outliers, are considered in model aggregation. The number of workers in Fig. 8 is 100, and the number of outliers is 10. If we ask each worker to test all the other workers, we will have 9900 messages passed between workers, as these messages carry model weights; hence, it will increase the communication load between the peer nodes. However, the communication overhead is still far below when compared with transmitting raw data in a P2P network.

For more scalability, we divided the workers into groups whereby the workers in each group validate models for the other workers within the same group only. For instance, if we have two groups of 50 workers each, we will reduce the number of messages passed between the workers to 5000. Therefore,

TABLE I
TOTAL TIME TAKEN TO VALIDATE OTHER WORKERS WITH DIFFERENT GROUP'S SIZES

Num of Groups	Workers Per Group	Num of Messages	Time (sec)
10	10	900	0.785
5	20	1900	1.560
4	25	2400	1.953
2	50	4900	3.880
1	100	9900	7.856

dividing workers into groups where each group is having a relatively smaller set of workers boosts the performance and the scalability of the system, as shown in Table I, while maintaining fairness through the outlier detection algorithm. The number of workers in each group should not be too small compared to the expected number of outliers in the system. The outliers might overtake a particular group by excluding honest workers. Our reputation-enabled worker selection strategy maximizes the chances of detecting the colluding workers when the workers are shuffled and reassigned during each training round.

We used $A_i \in \{\mu_i \pm \sigma_i\}$ as the initial statistical control limit for the outlier detection because of the small population sizes, and this control limit provides coverage to 95% of the population. Therefore, we were only able to identify significant malicious adversaries. Since the commercial-grade FL networks recruit FL participants from a large-scale population (from few millions to 1000 s of millions), TrustFed could help detect the hidden adversaries as well because training on the large-scale network will minimize the gap between average model loss and it will gradually fit well within the control limits. Therefore, a deep correlation analysis of average loss level and control limits will help determine the optimal control limits for the different sizes of the worker populations.

We performed the security analysis of smart contracts using Securify (the common smart contract security analysis tool) to ensure that our proposed CDFL system withstands all the known adversarial attacks on the blockchain network. Also, the proposed smart contracts return a false flag using low-level call methods coupled with external calls to detect malicious activities or exceptions in the code. We also ensured the safe execution of the smart contracts by using pull-based external calls. We explicitly labeled all state variables and functions to ensure easy accessibility on the Ethereum blockchain. Since all of the CDFL system participants try to build their reputation on the blockchain network, we ensure that all contracts are being executed autonomously without being attacked by the other smart contracts on the chain and without transferring reputation values to the other contracts. Using a shared state variable could lead to cross-function race conditions to attract the attackers in replicating the attack from the other functions with the shared state variables. Therefore, our proposed smart contract does not cater to any cross-function race condition, and the function calls are only made after a complete lifetime of the shared state variables.

VI. CONCLUSION

The presence of adversaries and malicious participants can jeopardize model performance in the FL environments. This article proposed TrustFed, a framework that enables public blockchain-empowered cross-device and cross-population fairness over decentralized FL training networks. The TrustFed ensured fairness by distributing the training network participants into multiple subgroups, detecting the outliers within the FL network, and performing the training and validation across the devices. TrustFed used the Ethereum blockchain smart contracts to incentivize the training network participants and maintain their on-chain reputation for active and honest contributions during the model training.

REFERENCES

[1] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," 2016, *arXiv:1610.05492*.

[2] K. Bonawitz *et al.*, "Towards federated learning at scale: System design," 2019, *arXiv:1902.01046*.

[3] C. Dwork *et al.*, "The algorithmic foundations of differential privacy," *Found. Trends Theor. Comput. Sci.*, vol. 9, no. 3/4, pp. 211–407, 2014.

[4] P. Kairouz *et al.*, "Advances and open problems in federated learning," 2019, *arXiv:1912.04977*.

[5] P. Bellavista, L. Foschini, and A. Mora, "Decentralised learning in federated deployment environments: A system-level survey," *ACM Comput. Surv.*, vol. 54, no. 1, pp. 1–38, 2021.

[6] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial internet of things: Challenges, opportunities, and directions," *IEEE Trans. Ind. Informat.*, vol. 14, no. 11, pp. 4724–4734, Nov. 2018.

[7] C. Dwork, C. Ilvento, G. N. Rothblum, and P. Sur, "Abstracting fairness: Oracles, metrics, and interpretability," in *Proc. 1st Symp. Found. Responsible Comput.*, 2020, pp. 8:1–8:16.

[8] *TrustFed Code*. Accessed: Apr. 12, 2021. [Online]. Available: https://github.com/habibcomsats/TrustFed.git

[9] S. R. Pokhrel and J. Choi, "Federated learning with blockchain for autonomous vehicles: Analysis and design challenges," *IEEE Trans. Commun.*, vol. 68, no. 8, pp. 4734–4746, Aug. 2020.

[10] S. R. Pokhrel, "Federated learning meets blockchain at 6G edge: A drone-assisted networking for disaster response," in *Proc. 2nd ACM MobiCom Workshop Drone Assist. Wireless Commun. 5G Beyond*, 2020, pp. 49–54. [Online]. Available: https://doi-org.libconnect.ku.ac.ae/10.1145/3414045.3415949

[11] P. K. Sharma, J. H. Park, and K. Cho, "Blockchain and federated learning-based distributed computing defence framework for sustainable society," *Sustain. Cities Soc.*, vol. 59, 2020, Art. no. 102220.

[12] Y. Li, C. Chen, N. Liu, H. Huang, Z. Zheng, and Q. Yan, "A blockchain-based decentralized federated learning framework with committee consensus," 2020, *arXiv:2004.00773*.

[13] N. Q. Hieu, T. T. Anh, N. C. Luong, D. Niyato, D. I. Kim, and E. Elmroth, "Resource management for blockchain-enabled federated learning: A deep reinforcement learning approach," 2020, *arXiv:2004.04104*.

[14] V. Mugunthan, R. Rahman, and L. Kagal, "BlockFlow: An accountable and privacy-preserving solution for federated learning," 2020, *arXiv:2007.03856*.

[15] 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.

[16] X. Hao, W. Ren, R. Xiong, X. Zheng, T. Zhu, and N. N. Xiong, "Fair and autonomous sharing of federate learning models in mobile Internet of Things," 2020, *arXiv:2007.10650*.

[17] J. Kang, Z. Xiong, D. Niyato, S. Xie, and J. Zhang, "Incentive mechanism for reliable federated learning: A joint optimization approach to combining reputation and contract theory," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 10700–10714, Dec. 2019.

[18] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Blockchain and federated learning for privacy-preserved data sharing in industrial IoT," *IEEE Trans. Ind. Informat.*, vol. 16, no. 6, pp. 4177–4186, Jun. 2020.

[19] *Ethereum Mainnet Statistics*. Accessed: Apr. 12, 2021. [Online]. Available: https://www.ethernodes.org/

[20] A. Saxena, K. Goebel, D. Simon, and N. Eklund, "Damage propagation modeling for aircraft engine run-to-failure simulation," in *Proc. Int. Conf. Prognostics Health Manage.*, 2008, pp. 1–9.

Muhammad Habib ur Rehman (Senior Member, IEEE) received the Ph.D. degree in mobile distributed analytics systems from the Faculty of Computer Science and Information Technology, University of Malaya, Kuala Lumpur, Malaysia, in 2017.

He is currently with the Division of Biomedical Engineering and Imaging Sciences, King's College London, London, U.K. Prior to this, he was with the Center for Cyber-Physical Systems, Khalifa University, Abu Dhabi, United Arab Emirates, as a Research Scientist. He is also an alumnae of DAAD's Postdoctoral Network since September 2019. His main research activities include research and development of trust models for decentralized and trustworthy artificial intelligence applications for cyber-physical systems.

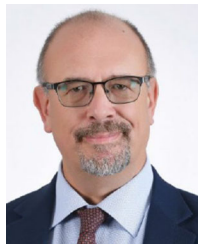
Dr. Rehman is a Bright Spark Fellow.

Ahmed Mukhtar Dirir received the B.S. degree in electrical engineering from the United Arab Emirates University, Al Ain, United Arab Emirates, in 2018. He is currently working toward the M.S. degree in computer science with Khalifa University, Abu Dhabi, United Arab Emirates.

His research interests include federated learning, blockchain technology, and Internet of Things.

Khaled Salah (Senior Member, IEEE) received the B.Sc. degree in computer engineering with a minor in computer science from Iowa State University, Ames, IA, USA, in 1990, the M.Sc. degree in computer systems engineering and the Ph.D. degree in computer science from the Illinois Institute of Technology, Chicago, IL, USA, in 1994 and 2000, respectively.

He is currently a Full Professor with the Department of Electrical Engineering and Computer Science, Khalifa University, Abu Dhabi, United Arab Emirates. He joined Khalifa University in August 2010, and is currently teaching graduate and undergraduate courses in the areas of cloud computing, computer and network security, computer networks, operating systems, and performance modeling and analysis.



Ernesto Damiani (Senior Member, IEEE) received the Ph.D. degree in computer science from Università degli Studi di Milano, Milan, Italy, in 1994.

He is currently the Senior Director of the Robotics and Intelligent Systems Institute, Khalifa University, Abu Dhabi, United Arab Emirates. He is also a Professor with the Department of Electrical and Computer Engineering and the Director of the Khalifa University Center for Cyber Physical Systems. He is the Chair of the Information Security Program and a Research Professor with EBTIC, Abu Dhabi. He is on extended leave from the Department of Computer Science, Università degli Studi di Milano, where he leads the SESAR research lab. He is also the President of the Italian Consortium of Computer Science Universities. His research interests include secure service-oriented architectures, privacy-preserving big data analytics, and cyber-physical systems security.

Davor Svetinovic (Senior Member, IEEE) received the Ph.D. degree in computer science from the University of Waterloo, Waterloo, ON, Canada, in 2006.

He is currently an Associate Professor of computer science with Masdar Institute, Khalifa University of Science and Technology, Abu Dhabi, United Arab Emirates, and a Research Affiliate with the Massachusetts Institute of Technology (MIT), Cambridge, MA, USA. He was a Visiting Professor with MIT, and was a Postdoctoral Researcher with Lero, the Irish Software Engineering Center, Limerick, Ireland, and Vienna University of Technology, Vienna, Austria. He leads the Strategic Artificial Intelligence Group, and he has extensive experience working on complex multidisciplinary research projects. He has authored or coauthored more than 80 papers in leading journals and conferences. His current research interests include cybersecurity, blockchain engineering, software engineering, machine learning, and artificial intelligence.

Dr. Svetinovic is a Senior Member of ACM.