Lightweight Blockchain-empowered Secure and Efficient Federated Edge Learning

Rui Jin, Jia Hu, Geyong Min, Jed Mills

Abstract—Federated Learning (FL) has emerged as a privacy-preserving distributed Machine Learning paradigm, which collaboratively trains a shared global model across a number of end devices (clients) without exposing their raw data. However, FL typically assumes that all clients are benign and trust the coordinating central server, which is unrealistic for many real-world scenarios. In practice, clients can harm the FL process by sharing poisonous model updates while the server could malfunction or misbehave. Moreover, the deployment of FL for real-world applications is hindered by the high communication overhead between the server and clients that are often at the network edge with limited bandwidth. To address these key challenges, we propose a lightweight Blockchain-Empowered secure and efficient Federated Learning (BEFL) system. BEFL is built by integrating a communication-efficient and mutual-information guarded training scheme, a cost-effective Verifiable Random Function (VRF)-based consensus mechanism, and Inter-Planetary File System (IPFS)-enabled scalable blockchain architecture. Extensive simulation experiments using two benchmark FL datasets demonstrate that BEFL is resistant against byzantine clients launching data poisoning and model poisoning attacks, fault-tolerant against colluded malicious blockchain nodes, scalable to a large number of blockchain nodes, and communication-efficient at the network edge.

Index Terms—Blockchain, Federated learning, Mutual information, Edge computing

1 Introduction

Recent years have witnessed the rapid development of Deep Learning technology, which has achieved groundbreaking success in a wide range of applications such as robotics, autonomous vehicles, and healthcare. The impressive performance of Deep Learning models relies substantially on large-scale and high-quality data for model training. Nowadays, networked edge devices, such as smartphones, vehicles, and sensors contain large amounts of valuable data [1]. However, collecting these raw data comes with inevitable communication and security issues, as many edge devices have limited bandwidth, and user-generated data on devices often contains privacy information (*e.g.*, locations and health records). To cope with these issues, Federated Learning (FL) emerges as a promising solution that enables multiple clients (data owners) to construct a joint learning model (*i.e.*, global model) without exposing their private training data. Orchestrated by a central server, clients periodically send their model updates to the server and retrieve the aggregated global model for the next round of training, until the global model converges.

However, the conventional FL process assumes that the server and clients are benign and reliable, which is unrealistic for many real-world scenarios. In practice, malicious clients may send poisonous model updates to the server. Recent studies show that FL is prone to these poisoning attacks [2]. Even one byzantine client can degrade the global model significantly [3]. Moreover, the centralized orchestration and aggregation of FL put the server at a dominant position, rendering the whole learning process to be highly vulnerable to malfunction or misbehaviour of the server. To mitigate these two issues of malicious clients and server bottleneck, many research endeavours toward robust and decentralized FL have been carried out. Defence mechanisms such as Multi-krum [2], Trimmed Mean (TM) [4], Coordinate-wise Median (CM) [4], Bulyan [5], and Robust Federated Averaging (RFA) [6] have been proposed to protect against malicious clients in FL. To enable decentralized aggregation and eliminate the single point of failure in FL, a promising approach is the blockchain technology, which has attractive security features including tamper-resistance, transparency, traceability, *etc*.

There have been a number of research works dedicated to blockchain-based FL. For example, some papers proposed blockchain-based FL systems using full-chain deployment on client devices [7]–[11]. However, those systems are costly in terms of communication and storage due to the large block size (within the ever-growing blockchain) containing local model updates and the global model. The work in [8] alleviates this problem by allowing clients to delete historical blocks according to a local resource budget, but its credibility of security inevitably decreases due to the deletion of blocks. Several other blockchain-based FL works proposed to deploy blockchains to edge nodes with greater capacity than client devices [12] [13]. However, the ChainsFL [12] requires the sharing of a test dataset for local model update validation, which is not compatible with the privacy-preserving objective of FL, while [13] is vulnerable to the single point of failure since the centralised aggregation is carried out only by the task publisher. Another branch of works uses a popular blockchain platform like Ethereum with smart contracts [14] [15], however, they are costly to operate, as every interaction with the blockchain consumes a monetary gas fee. Thus, there is a lack of a cost-effective, secure and reliable blockchain-based FL system that runs at

- *Rui Jin, Jia Hu, Geyong Min and Jed Mills are with the Department of Computer Science, University of Exeter, Exeter EX4 4QJ, U.K. (e-mail: {rj390; j.hu; g.min; jm729}@exeter.ac.uk)*
- *Corresponding authors: Jia Hu and Geyong Min.*

vulnerable network edge with resource-constrained client devices.

To fill this gap, we propose a novel lightweight, secure and efficient Blockchain-Empowered Federated Learning (BEFL) system tailored for the wireless edge, which has promising applications in areas such as connected and autonomous vehicles, mobile crowdsourcing, and smart healthcare. BEFL eliminates the role of the FL server, with a group of edge nodes running and maintaining a blockchain system for FL model aggregation and distribution. Considering the massive communication cost that arises from clients iteratively sending model updates to the edge, we exploit a state-of-the-art compression mechanism PowerSGD [16] to compress model updates into low-rank matrices to reduce the communication overhead on the client side with tolerable computation cost. To enable the byzantine robustness over compressed model updates which inevitably entail biases compared to raw model updates, BEFL utilizes the Mutual Information (MI) between clients' models to capture their inherent correlation and choose reliable updates according to their MI values. BEFL also incorporates Verifiable Random Function (VRF) and Inter-Planetary File System (IPFS) to empower the lightweight blockchain system, with a set of selected blockchain nodes (committee) in charge of block generation to ensure the reliability of global model. The major contributions of this paper are summarized as follows:

- The proposed BEFL moves the model aggregation work from the FL server to the blockchain nodes (*i.e.,* edge servers), thus eliminating the single point of failure in conventional FL. BEFL also utilises the IPFS to store the global model with its address recorded on the chain instead of the whole model, to reduce the communication cost of block propagation and the storage cost of the ever-growing blockchain.
- We propose a novel byzantine-resilient communication-efficient training scheme leveraging the MI between clients and the PowerSGD method, to protect against malicious model updates whilst reducing the communication overhead. This new training scheme is embedded in the operating process of the BEFL system.
- Instead of the conventional computation-intensive Proof-of-Work (PoW) mechanisms to achieve the consensus in the blockchain, we propose an energy-efficient committee-based consensus protocol using VRF. This new protocol selects committee members with probabilities proportional to their stakes to protect against malicious blockchain nodes and to validate a candidate block that contains the IPFS address of the global model.
- Extensive experiments were conducted with benchmark non-independent identically distributed (non-i.i.d) datasets (FEMNIST and CIFAR10) for FL under both honest and adversarial settings. Experimental results show that BEFL is resistant to clients acting maliciously and launching data poisoning and model poisoning attacks. BEFL could also achieve better performance than baselines under the benign setting with reduced communication overhead. BEFL's performance was unaffected with $\frac{1}{3}$ malicious blockchain nodes in the network. Furthermore, the block generation and distribution time barely increased when the network size scaled up from 100 to 800 nodes.

The rest of the paper is organized as follows. We describe the related work in Section 2 and the preliminaries in Section 3. We detail our system design in Section 4. The implementation and evaluation of our system are presented in Section 5. Finally, we conclude our work in Section 6.

2 Related Work

The highly attractive privacy-preserving nature of FL has resulted in intense research activity, but its centralized orchestration and lack of control of clients leave huge attacking space to the server and clients. To tackle the threat induced by malicious clients who launch poisoning attacks, robust FL has been extensively explored. Blanchard *et al.* [2] proposed Multi-krum, leveraging the Euclidean distance between model update vectors to select updates "close to the barycenter" for the averaging aggregation step. Different from Multi-krum, TM and CM perform element-wise aggregation [4]. CM takes the coordinate median of each parameter across all collected local model updates and TM averages each coordinate parameters without the largest and the smallest β fraction of values. Combining Multi-krum and TM, Bulyan [5] was proposed to reduce the leeway of byzantine clients. However, the effectiveness of these works has only been demonstrated using i.i.d datasets, which is not consistent with the non-i.i.d FL environment. In this regard, Pillutla *et al.* [6] proposed RFA to approximate the geometric median of model updates vectors as the global model parameters and achieved byzantine robustness in the non-i.i.d FL setting, but tripled the communication overhead. These robust statistics-based guarding mechanisms, however, fail to consider the situation when the server malfunctions or misbehaves.

To handle the issue brought about by an unreliable server, blockchain-based FL becomes a promising solution. The peer-to-peer distributed nature of blockchain makes the decentralized aggregation of global model available, shedding new light on handling the single point of failure in conventional FL. Research works [7]–[10], [12], [14], [17]–[21] move the aggregation step from the server to the blockchain nodes, while Qu *et al.* [22] and Mugunthan *et al.* [15] eliminate the role of the aggregator by letting clients themselves aggregate model updates obtained from the blockchain network. To ensure the reliability of the global model, most blockchained FL frameworks embed local model update verification into their design. Under different assumptions on blockchain nodes, the verification methods vary. Several blockchain-integrated FL frameworks assume that clients join the blockchain network and participate in FL task simultaneously, so that their training data could be used as the validation datasets [8], [9], [17], thus the validation accuracy of local model updates (recorded in the transaction) could be further utilized in the aggregation process. Li *et al.* [8] introduced K-fold cross-validation where K committee members test the model update on their training data and take the median of the accuracy values

as the score of the update. Model updates with qualified scores would be selected for global model aggregation. Lu *et al.* [17] applied a similar strategy to select model updates with accuracy lying within a certain range to update the global model. This accuracy-based validation mechanism has also been utilized in blockchain-based FL systems where nodes have no access to the training data. Mugunthan *et al.* [15] implemented smart contract with a cross-verification procedure that clients cross-verify others' model updates by testing them over their local training datasets, and send the accuracy values back for the calculation of contribution scores, which would be used in the weighted aggregation stage.

Instead of evaluating model updates with the help of clients, which would inevitably increase the communication burden and be restricted with clients' active status, recent blockchain-based FL frameworks reaped recent advances in Robust FL to protect against malicious clients. Biscotti [7], SPDL [11] and Omnilytics [14] applied the Multi-krum as the validation mechanism and give passes to model updates with closer Euclidean distances. Biscotti introduces noiser, verifier and aggregator nodes, where noisers produce differentially private (DP) Gaussian noise to be added to the model updates, verifiers run Multi-krum to sign commitments for passed updates, and aggregators aggregate unmasked passed model updates via a secure protocol. Similar to Biscotti, SPDL leverages DP Gaussian noise to local model update and Multi-krum in model updates aggregation to provide private and secure FL with theoretical convergence guarantee. Omnilytics was implemented using smart contract with incentives and punishments to honest and malicious clients respectively. However, the proposed designs have only been tested using i.i.d datasets, so their performance under non-i.i.d FL setting has not been explored.

The deployments of proposed blockchain-based FL systems in the research community roughly fall into three categories: to client devices [7]–[10], to edge nodes [12], [13], [23], and to the existing blockchain platform Ethereum with smart contracts [14], [15]. Considering the limited resources of client devices, running and maintaining blockchain locally is costly in terms of communication and storage for the large propagated block in the network and the ever-growing blockchain. For PoW-based blockchain systems [10], the additional computation cost of running the blockchain is also non-negligible. The existing mature platform, Ethereum, that enables smart applications running on the blockchain provides secure execution of FL tasks but is monetary costly in terms of the gas fee induced by every interaction with the blockchain. The deployment to edge nodes is promising for their broad capacity in terms of computation, communication and storage, but existing blockchain-based FL works targeting this scenario fail to address the security issue brought by malicious clients and an unreliable server, and the high communication overhead for resource-constrained clients concurrently. In this regard, we propose BEFL to enable secure and efficient FL at wireless edge with practical potential.

3 Preliminaries

We will introduce the basics of blockchain, IPFS and VRF in this section.

3.1 Blockchain

Since the inception of Bitcoin, the underlying blockchain technology has received considerable attention from both academia and industry for its peer-to-peer, transparent, and immutable features. Popular blockchain systems (e.g., Bitcoin and Ethereum) enable a consensus among different nodes via Proof-of-Work (PoW) where miners (nodes) competitively solve a mathematical puzzle to grow the blockchain, with the longest valid chain regarded as authoritative. However, PoW allows the possibility of forks, where two chains have the same length. It requires more blocks to be appended to mitigate the forks, thus leading to long transaction confirmation times. In this regard, consensus mechanisms based on Byzantine Fault Tolerance (BFT) with built-in block finalization have been proposed, such as the protocols deployed in Algorand [24], Casper FFG [25], and Hyperledger Fabric [26]. Taking advantage of the built-in consensus finality, high transaction capacity, and great fitness to permissioned blockchain of BFT protocols, we apply a similar strategy with Algorand to reach the consensus among blockchain nodes.

3.2 IPFS

Inter-Planetary File System is a peer-to-peer distributed file system that provides a high throughput content-addressed block storage model with content-addressed hyperlinks (the unique hash values) [27]. Its Distributed Hash Table (DHT) and Merkle Directed Acyclic Graph (DAG) make efficient data storage and file retrieval. Any modification of a file would result in a different hash value, thus the integrity of the stored file is ensured. We use IPFS to store the global model on the BEFL nodes, reducing the storage cost of the blockchain and ensuring the integrity of the global model.

3.3 VRF

Verifiable Random Function [28] is a public-key pseudorandom function that provides proof for its random outputs with certain inputs. VRF provides blockchain nodes a non-interactive way to independently determine if they were chosen to be the committee members [24]. These unique characteristics of VRF allow us to use it to allocate the blockchain committee in BEFL.

4 Blockchain-Empowered secure and efficient Federated Learning (BEFL)

We propose the BEFL system with the following goals: 1) prevent the single point failure and abnormal aggregation operations, 2) reduce the communication cost of clients without degrading the global model performance, 3) mitigate the influence of potential malicious clients, 4) be effective under the realistic non-i.i.d FL setting, and 5) be lightweight and able to scale to a large number of blockchain nodes and complex FL tasks (*i.e.*, large deep learning models with millions of parameters).

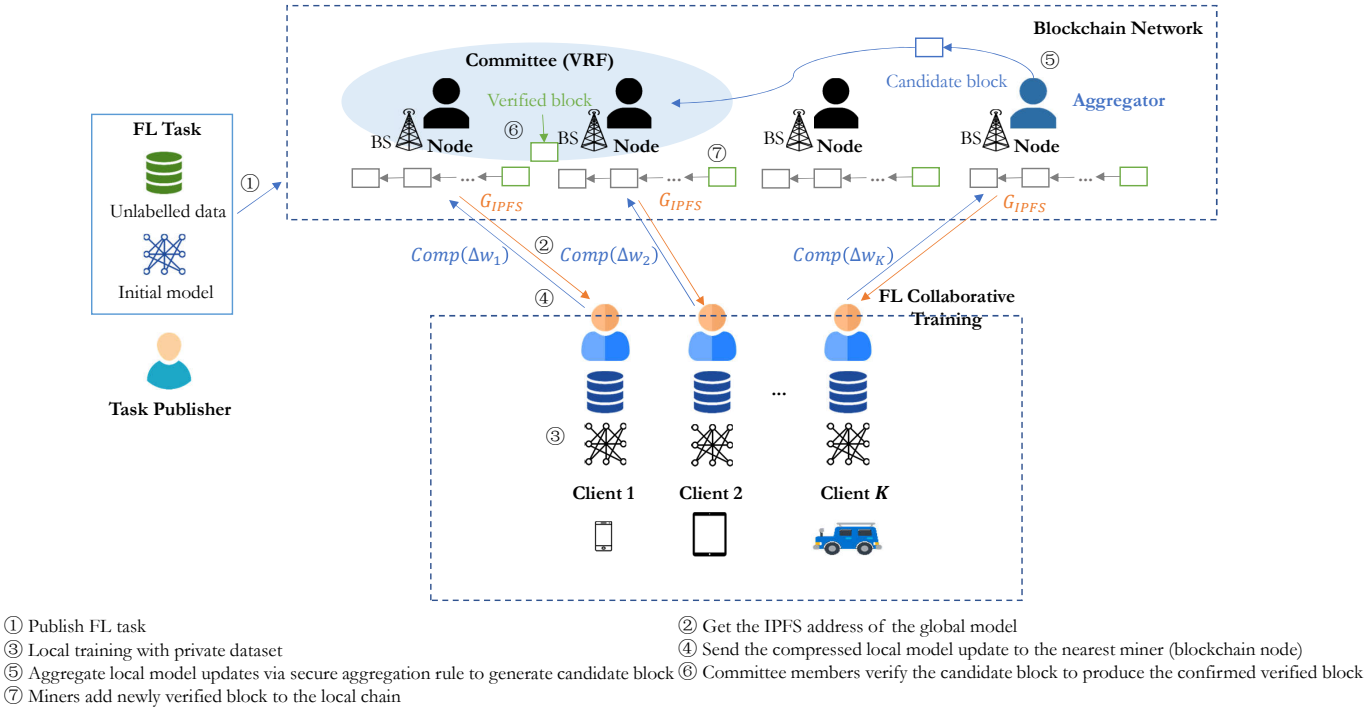

Fig. 1. The system overview of BEFL

Design Overview: As shown in Fig. 1, we deploy our blockchain system at the network edge due to the high computation, communication, and storage capacity of edge nodes (*i.e.*, Base Stations (BSs)). We assume that the task publisher has access to a set of unlabeled data containing all the classes of training data on clients, with which we calculate the MI between clients. This assumption is easily satisfied, as in FL, the task publisher (or the server) usually collects its own data for model validation, which contains all the classes and is i.i.d, while collecting unlabeled data is simpler than labeled ones [29].

The task publisher first publishes the FL task with unlabeled data and the initial global model that stored in the IPFS to the blockchain network (Step ①). The blockchain nodes retrieve the task information from the network and keep a replica of the unlabeled dataset. Clients then could get the IPFS address of the global model via querying the nearest active node (Step ②). Utilising the content-addressed IPFS, the global model could be easily downloaded. Starting with this global model, clients run local training steps using their private datasets. Once the local training is complete, clients compress the model updates and send the compressed gradients ($Comp(\Delta w)$) together with their signatures as transactions to the nearest active node (Step ③ and ④). Receiving a model update from a client, the blockchain node first verifies whether it is issued by the client via checking the signature and recovering the compressed update to see if its shape is consistent with the global model. If the two requirements are met, the node disseminates this transaction to the network. Once enough pending transactions (local model updates) are received, nodes competitively calculate the global model for next round following the secure aggregation protocol we designed, generate the candidate block and send it to the committee that was allocated with VRF (Step ⑤). Committee members then verify if the global model is aggregated correctly with their "Yes" or "No" votes. If the candidate block receives more than $\frac{2}{3}$ agreements from the committee, it is regarded as a verified block and propagated to the network. After receiving the newly verified block, nodes check the signatures of the committee members and add it to the local chain (Step ⑥).

4.1 Initialization

We assume that a trusted authority bootstraps the permissioned blockchain system with eligible edge nodes (*i.e.*, BSs). Upon registration, blockchain nodes are allocated a VRF key pair (SK, PK) for committee constitution, a signature commitment key pair (sk, pk), and the default stake. The genesis block of the blockchain is a fixed block containing information about the system. The task publisher publishes the FL task with the unlabeled data X, the IPFS address G^0_{IPFS} of the initial model W^0_g and the momentum $\mathbf{m^0}$ which is initialized with 0, and the hyperparameters of the FL model.

4.2 Stake-based Committee Constitution

Using VRF, BEFL selects committee members in a private and non-interactive manner. Based on VRF, we propose a novel sortition algorithm for choosing a random subset of nodes to constitute the committee. Each node runs the sortition algorithm independently with a public seed (the latest block of blockchain) as its input to see if itself is selected as a candidate committee member. It is non-trivial that the sortition selects nodes in proportion to their owned stakes; otherwise, it would be vulnerable to Sybil attack [30],

Algorithm 1 Sortition

Input: the latest block $seed$, stake value of node i s_i, the total stake of blockchain system S, hyperparameter α and the committee size K.

Output: $select, hash, proof$

$hash, proof \leftarrow VRF_{SK_i}(seed)$
$r = \frac{hash}{2^{hashlen}}$
if $\frac{r}{s_i} < \frac{\alpha K}{S}$ **then**
  $select =$ **true**
else
  $select =$ **false**
end if

where a single faulty entity presents multiple entities, thus controlling a substantial fraction of the system. As shown in the *Sortition* algorithm, if the generated unique random $hash$, determined by the secret key SK_i and the input $seed$, satisfies the condition, the node i becomes a candidate committee member and sends the eligible information ($hash$ and $proof$) to the network. The r generated via VRF is a unique uniform random value that lies within the range of $[0, 1]$, thus its probability density function is $f(r) = 1$. The probability of blockchain node i being chosen to be a committee member could be computed as follows:

$$p_i = Pr(\frac{r}{s_i} < \frac{\alpha K}{S}) = Pr(r < \frac{\alpha K s_i}{S}) = \frac{\alpha K s_i}{S}, \quad (1)$$

it could be seen from the above equation that the more stake the node owns, the higher probability of it being chosen for the committee. The committee constitution phase ends when the correct size of the committee is achieved, first K committee members become the authoritative ones. Any node could get the committee information from the network and verify the identity of committee members using their public keys of VRF, the latest block of the blockchain, and the stakes they owned. The hyperparameter α controls the expected number of nodes that could be chosen to become the candidate committee members. As each node of being a candidate committee member follows the Bernoulli distribution, we let X_i denote the trial of node i, where $X_i = 1$ with probability p_i denoting success trial, and $X_i = 0$ with probability $1 - p_i$ indicating unsuccessful one, thus the successful trials $X = \sum_{i=1}^{n} X_i$. The expectation of X is computed as follows:

$$E(X) = \sum_{i=1}^{n} p_i = \alpha K. \quad (2)$$

According to the Chernoff bound on the sum of independent Bernoulli trials, we could get

$$Pr(X \leq (1-\delta)\alpha K) \leq e^{-\alpha K \delta^2/2}, \quad (3)$$

for any $0 \leq \delta \leq 1$. Taking $\delta = \frac{\alpha - 1}{\alpha}$, where α lies within $(1, \frac{n}{K}]$, we obtain

$$Pr(X \leq K) \leq e^{-K(\alpha-1)^2/2\alpha}. \quad (4)$$

To ensure enough candidate committee members exist in each committee constitution phase, the probability equation (4), indicating not having $K + 1$ candidate committee members should be small to 0. We discuss the choice of K and α in the Performance Evaluation section.

4.3 Communication-efficient Distributed Training

Instead of obtaining global model from the server, BEFL enables participating clients to get the latest global model from the nearest active blockchain node through sending their requests. The node then returns the IPFS address of the latest global model with its recorded relative training round, so that the client can check whether the global model is the newly updated one. Taking advantage of the content-addressed IPFS, clients can download the global model efficiently as the file is provided by the available nearest IPFS servers. Upon obtaining the global model W_g^t, new local model w^t is trained using conventional Stochastic Gradient Descent (SGD) algorithm on the private training data D.

To save the communication cost of uploading the model update $\Delta w^t = w^t - W_g^t$ for global aggregation, we apply the state-of-the-art gradient compression mechanism PowerSGD, which incorporates error-feedback [16], to compress the transmitted update. As shown in Algorithm 2,

Algorithm 2 Distributed model update with PowerSGD

The error $\mathbf{e}$ is initialized with $0 \in R^d$, $\Delta w \in R^d$. In the compression phase, vector Δw would be reshaped into matrices. For each matrix $M \in R^{m \times n}$, a corresponding $Q \in R^{m \times r}$ is initialized from an i.i.d standard normal distribution
Client execute:
at each training round $r = 0, ...$ **do**
  Compute a stochastic gradient g_w
  $\Delta w = g_w + \mathbf{e}$
  $\widehat{\mathbf{P}}, \mathbf{Q} \leftarrow COMPRESS(\Delta w)$
  $\mathbf{e} \leftarrow \Delta w - DECOMPRESS(\widehat{\mathbf{P}}, \mathbf{Q})$
  upload $\widehat{\mathbf{P}}, \mathbf{Q}$

function COMPRESS(Δw)
  $\{M_1, M_2, ...M_W\} \leftarrow \Delta w$
  for $i = 1, 2, .., W$ **do**
    $P_i \leftarrow M_i Q_i$
    $\widehat{P_i} \leftarrow ORTHOGONALIZE(P_i)$
    $Q_i \leftarrow M^T \widehat{P_i}$
  end for
  $\widehat{\mathbf{P}} = \left\{\widehat{P_1}, \widehat{P_2}, ..., \widehat{P_W}\right\}$
  $\mathbf{Q} = \{Q_1, Q_2, ..., Q_W\}$
  return compressed representation $(\widehat{\mathbf{P}}, \mathbf{Q})$
end function

function DECOMPRESS($\widehat{\mathbf{P}}$,$\mathbf{Q}$)
  for $\widehat{P}$ in $\widehat{\mathbf{P}}$, Q in $\mathbf{Q}$ **do**
    $M = \widehat{P} Q^T$
  $\Delta w^* \leftarrow \mathbf{M}$
  return Δw^*
end function

the low-rank PowerSGD decomposes the gradient matrix $M \in R^{n \times m}$ into $\widehat{P} \in R^{n \times r}$ and $Q \in R^{m \times r}$. Compared to traditional FedAvg [31], the additional computation cost brought by compression is relatively low, which involves one left multiplication, one right multiplication, and an orthogonalization (achieved by Gram-Schmidt procedure

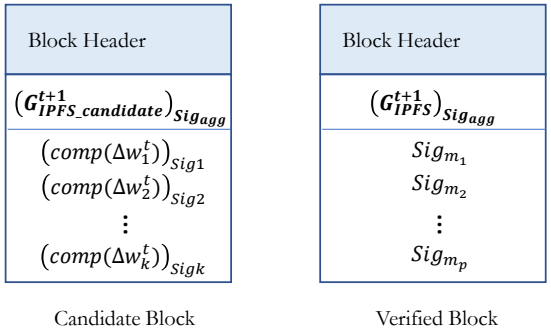

Fig. 2. Block Structure

with $O(mn^2)$, where m and n are the dimensions of the matrix) on each matrix in gradient vector Δw.

4.4 Transaction and Block Design

A blockchain is a list of blocks linked with hash pointers. Each block has a block header, consisting of the index of the block, timestamp, and the hash string of previous block, and a block body that contains transactions. As shown in Fig. 2, each compressed model update with the signature signed by the client forms the transaction. Model updates utilized for global model aggregation together with the IPFS address of the aggregated model W_g^{t+1} and updated momentum $\mathbf{m^{t+1}}$ signed by the aggregator would be packed into the body of the candidate block. Once the candidate block has been confirmed, local model updates contained in this block will be removed for their uselessness to future rounds of learning process and benefit of storage saving. Instead, signatures of the block header and IPFS address of the confirmed global model issued by committee members who vote "Yes" to the candidate block would be recorded.

4.5 Secure Aggregation Protocol

The privacy-preserving principle of FL makes it susceptible to malicious clients who may send poisonous model updates to corrupt the joint learning process. The low-rank approximated model update inevitably introduces biases to the raw model update, thus making the identification and mitigation of potential malicious model updates more difficult. As shown in [32], the MI between honest clients shows an increasing trend with training round, instead of the Euclidean distance. We propose a novel robust aggregation mechanism leveraging the underlying MI between clients to capture the differences between honest and malicious clients.

MI is a powerful statistic for measuring the degree of dependence [33], it captures the amount of shared information between two random variables. We treat the output of client's local model as a random vector due to the stochastic property of SGD. We assume that the output F_i of local model w_i from client i and output F_j of local model w_j from client j follow Gaussian functions with respective variances σ_i^2 and σ_j^2. The MI between client i and j, denoted by $MI_{i,j}$ could be calculated as [34]:

$$MI_{i,j} = MI(F_i, F_j) = H(F_i) + H(F_j) - H(F_i, F_j), \quad (5)$$

where $H(F_i)$ is the entropy of F_i, $H(F_j)$ is the entropy of F_j, and $H(F_i, F_j)$ is joint entropy of F_i and F_j, which are defined as:

$$\left.\begin{aligned} H(F_i) &= \tfrac{1}{2}[1 + \log(2\pi\sigma_i^2)] \\ H(F_j) &= \tfrac{1}{2}[1 + \log(2\pi\sigma_j^2)] \\ H(F_i, F_j) &= 1 + \log(2\pi) + \tfrac{1}{2}\log[\sigma_i^2\sigma_j^2(1 - \rho_{ij}^2)] \end{aligned}\right\}. \quad (6)$$

In Eq. (6) the correlation coefficient ρ_{ij} is defined as:

$$\rho_{ij} = \frac{E[(F_i - E[F_i])]E[(F_j - E[F_j])]}{\sigma_i^2\sigma_j^2}, \quad (7)$$

where $E[\cdot]$ denotes the mathematical expectation. From Eq. (5) and (6), we could get

$$MI_{i,j} = -\frac{1}{2}\log(1 - \rho_{ij}^2). \quad (8)$$

It can be seen from Eq. (8) that, 1) when the outputs F_i and F_j are highly correlated, the correlation coefficient ρ_{ij} is close to 1 and thus the MI value would be noticeably large; 2) when F_i and F_j are barely correlated, the correlation coefficient ρ_{ij} is close to 0 and their MI becomes very small. The outputs of clients' local model are obtained from decompressing their model updates and the unlabeled dataset provided by the FL task publisher.

Algorithm 3 BEFL aggregation

The global model W_g^t and momentum value $\mathbf{m^t}$ of last round, unlabeled dataset of FL task X, and the received valid model updates set $S = \{Comp(\Delta w_1), Comp(\Delta w_2), \ldots, Comp(\Delta w_k)\}$
for each compressed model update $Comp(\Delta w_i)$ in S **do**
  $\Delta w_i \leftarrow DECOMPRESS(Comp(\Delta w_i))$
  $w_i = \Delta w_i + W_g$
end for
for each w_i **do**
  Compute $MI_{i,j}, (j = (1, \ldots, k), j \neq i)$
  $MI_i = \mathbf{mean}\{MI_{i,j} : j = (1, \ldots, k), j \neq i\}$
end for
$MI_{mad} \leftarrow MAD(\mathbf{MI})$
$MI_{madn} = \frac{MI_{mad}}{0.6745}$
$\mathbf{\Delta w} \leftarrow 0$
for $i = 1, \ldots, K$ **do**
  $t_i = \frac{MI_i - Median(\mathbf{MI})}{MI_{madn}}$
  $count = 0$
  if $t_i \leq 2$ **then**
    $\mathbf{\Delta w} = \mathbf{\Delta w} + \Delta w_i$
    $count++$
  end if
end for
$\mathbf{m^{t+1}} = \beta\mathbf{m^t} - \eta\frac{\mathbf{\Delta w}}{count}$
$W_g^{t+1} = W_g^t - \beta\mathbf{m^t} + (1 + \beta)\mathbf{m^{t+1}}$

As shown in Algorithm 3, the MI values between different clients calculated via Eq. (8) are leveraged to the secure aggregation process. The correlation coefficient ρ_{ij} is:

$$\rho_{ij} = \frac{\sum_{q=1}^{N}(F_i(x_q) - E[F_i(x_q)])(F_j(x_q) - E[F_j(x_q)])}{\sqrt{\sum_{q=1}^{N}(F_i(x_q) - E[F_i(x_q)])^2 (F_j(x_q) - E[F_j(x_q)])^2}}, \quad (9)$$

where x_q denotes the unlabeled data sample and N is the number of data samples. To quantify the correlation of each model update to others, the MI values between each other model update are averaged to produce the MI score. Similar to the Multi-krum [2] robust aggregation rule, our method selects model updates "close to the barycenter". Model updates with excessively low MI scores are suspected to be malicious for the intrinsic large difference, while the one with a very high score tends to increase the redundancy in model aggregation which may slow down the convergence speed. To capture the distance of each MI score to the barycenter, we utilize the standard deviation (SD) robust alternative, median absolute deviation about the median (MAD) which is computed as:

$$MI_{mad} = MAD(\mathbf{MI}) = Median(\mathbf{MI} - Median(\mathbf{MI})), \tag{10}$$

where $\mathbf{MI} = \{MI_1, MI_2, \ldots, MI_k\}$ and k is the number of received valid pending transactions that contain the model updates from clients. To make MAD comparable to SD, we normalize the MAD as:

$$MI_{madn} = \frac{MI_{mad}}{0.6745}, \tag{11}$$

where 0.6745 is the MAD value of a standard normal distribution. To filter out potential malicious model updates as well as less contributive ones, we use the standard heuristic of taking all values lying within two SDs of the mean (95% probability) as empirically useful. We replace mean and SD here to median and normalized MAD (MADN) as robust location and dispersion measures [9]. Model updates with MI scores lie within the 95% confidence level are selected for aggregation. The global model is updated with Nesterov's momentum using the selected gradients.

After the aggregation, the updated global model W_g^{t+1} and momentum $\mathbf{m^{t+1}}$ will be uploaded to the IPFS by the aggregator, with a unique hash string denoting the storage address returned back. The aggregator could then generate the candidate block with obtained IPFS address and pending transactions and send it to the committee for verification.

4.6 Consensus Achievement

As the defining technology behind the security of blockchain, the consensus protocol is of significant importance. The VRF-enabled committee constitution guarantees resistance against Sybil attacks as no blockchain node could predict the next generated block in advance, and there are limited stakes that malicious nodes could hold. As shown in Algorithm 4, when a new candidate block is distributed to the network, committee members that are in charge of verifying candidate blocks first check whether enough pending transactions are collected for global model aggregation and the signature of the IPFS address is issued by the aggregator using its public key pk_{agg}. If the aforementioned condition is met, committee members calculate the global model and momentum values following the Secure Aggregation Protocol (see Section 4.5) and compare them with the ones obtained from the IPFS. If the results are the same as the aggregator's, committee members vote "Yes" to the candidate block with their signatures regarding the block header and the signed IPFS address. We use the Elliptic Curve Digital Signature Algorithm (ECDSA) to make the signature commitment which is utilized in Bitcoin. If more than $\frac{2}{3}$ agreements from committee members are received, the candidate block is regarded as the valid verified block with transactions replaced by signatures from supported committee members. This $\frac{2}{3}$ consensus threshold is based on the concept of the Byzantine Fault Tolerance (BFT) [35]. In a distributed system like a blockchain, some of nodes (or validators) may be faulty and behave maliciously or erroneously, leading to inconsistencies in the system. The BFT consensus algorithm is designed to tolerate up to one-third of the nodes being faulty. Setting the consensus threshold to $\frac{2}{3}$ ensures that BEFL can tolerate up to one-third of the committee members being faulty (either due to malicious intent or technical issues), while still achieving consensus on the verified block. In BEFL, the voting process for each candidate block occurs only for a specified duration. Any candidate block that fails to receive enough agreements during the time duration will be dropped and the committee will verify the next candidate block. For the committee constituted each time, BEFL specifies its maximum voting step. If there is no candidate block being confirmed during these voting rounds, a new committee will be constituted for verifying new candidate block.

Once the block is confirmed, stake reward would be distributed to both the committee members and the block generator. Any transactions in the pending transaction pool will be cleared: BEFL does not append stale updates to the model.

Algorithm 4 Consensus

Candidate Block $cBlock$, maximum vote round $MaxStep$, maximum voting time $MaxVoteDuration$.
$step \leftarrow 1$
while $step < MaxStep$ **do**
  if voting time is within $MaxVoteDuration$ **then**
    $votes, sigs \leftarrow$Committee members vote on $cBlock$
    if "Yes" votes are more than $\frac{2K}{3}$ **then**
      $vBlock \leftarrow cBlock, sigs$
      return $vBlock$
    else
      $step^{++}$; wait for new candidate block
    end if
  end if
end while
Committee reconstruction; wait for new candidate block.

5 Performance Evaluation

We implemented BEFL with Go 1.16 and Python 3.7. The networking and distributed aspects of our design are built using Go. We utilized PyTorch to train models and generate SGD updates. BEFL interfaces Go and python via the *go-python3* library [36]. We used *go-ipfs-api* [37] to connect with the IPFS. The cryptography part of our design is implemented with *Coniks* [38] library which is used for VRF implementation and built-in crypto package of Go that contains ECDSA implementation.

TABLE 1
The default parameters of BEFL

Parameter	Value
Committee size K	15
Control parameter α	3
Maximum voting round $MaxVoteStep$	5
Voting time out $MaxVoteDuration$	200 seconds
Number of blockchain nodes	100
Rank of FEMNIST SGD update	2
Rank of CIFAR10 SGD update	4
Beta β	0.9
Eta η	1
Learning rate of client's SGD	0.1
Batch size of client's model training	64
Local steps of client's model training	5
Initial stake	uniform, 1 each
Stake update	linear, +1

We deployed BEFL to a machine with one NVIDIA Getforce GTX 1080 Ti GPU, one Intel (R) Xeon (R) CPU E5-2630 v3 @ 2.40GHz and 64GB of RAM. All experiments were executed over non-i.i.d datasets: FEMNIST and CIFAR10. FEMNIST was preprocessed using the Leaf federated benchmark tool [39] consisting of 62 different classes (numbers and letters). We utilized the non-i.i.d CIFAR10 partition constructed from LotteryFL [40] where clients can have different classes of unbalanced data with different degrees. We assigned each CIFAR10 client 10 classes of data with the unbalanced degree of 0.75. All FL tasks were loaded with 50 workers and 20 of them are selected randomly to participate in each FL training round. The parameter values of BEFL are presented in Table 1 unless stated otherwise. To ensure enough candidate committee members exist in each committee constitution phase, we set committee size K and control parameter α with 15 and 3 respectively, so that the probability of not having $K+1$ candidate committee members is smaller than 4.54×10^{-5} according to the equation (4), indicating it would barely happen. We took 1000 random samples without their labels of test dataset as the unlabeled dataset. We implemented a custom CNN comprising: a convolutional layer with relu, a max-pooling layer, a convolutional layer with relu, a max-pooling layer, a fully connected layer with relu, and the output layer with softmax, for the FEMNIST task, and a ResNet14 [41] for CIFAR10. Each experiment was repeated 3 times, with the mean and 95% confidence interval (CI) plotted in the relevant figures.

We compare BEFL with FedAvg and Biscotti. We chose Biscotti as the baseline as it shares the same objective of defending against malicious clients and solving the centralized server bottleneck issue in federated learning using blockchain technology.

5.1 Communication efficiency

In this section, we evaluate the communication cost of each client for participating in a training round. The sizes of transmitted model updates for both FEMNIST and CIFAR10 tasks are reduced significantly, from 3399.742kb to 50.727kb in FEMNIST, and 764.602kb to 151.070kb in CIFAR10. By compressing the model updates into low-rank matrices, the uploading message became $67\times$ smaller of FEMNIST task and $5\times$ smaller of CIFAR10 task. We calculate the average Euclidean distance between the decompressed and the original data (i.e., model updates) of clients of the two FL tasks in which 500 training rounds have been performed with 20 clients participating in each round. The average Euclidean distance between the decompressed data and the original data is 1.771 in FEMNIST and 1.496 in CIFAR10. We also calculate the average absolute error per parameter between the decompressed data and the original data, which is 0.002 for FEMNIST and 0.001 for CIFAR10. From the calculated numerical results, we could see that there is not a big difference between the decompressed data and the original data. As shown in Fig. 3, BEFL achieves equivalent performance compared to conventional FedAvg [31] under the honest setting. Due to non-i.i.d data, Biscotti filtered out partial contributive model updates in terms of their long Euclidean distances to others resulting in 3.55% and 3.92% accuracy dropping down for the FEMNIST and the CIFAR10 task respectively. Moreover, BEFL converges faster than traditional FedAvg and Biscotti due to the intrinsic model update selection and momentum update of the global model. Model updates with excessively high MI scores tend to increase the redundancy in the aggregation process and ones with very low MI scores are suspected to induce high variance. Using momentum of SGD updates has proven to accelerate the network training [42] and the oscillations brought by compression could be dampened.

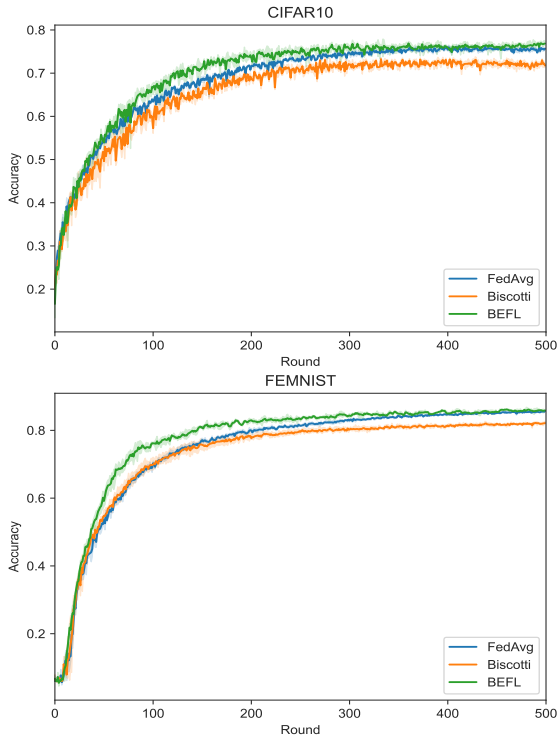

Fig. 3. Test accuracy of FedAvg, Biscotti and BEFL over CIFAR10 and FEMNIST datasets when there is no attack. Curves are averages over 3 random trials, shaded regions represent 95% CIs

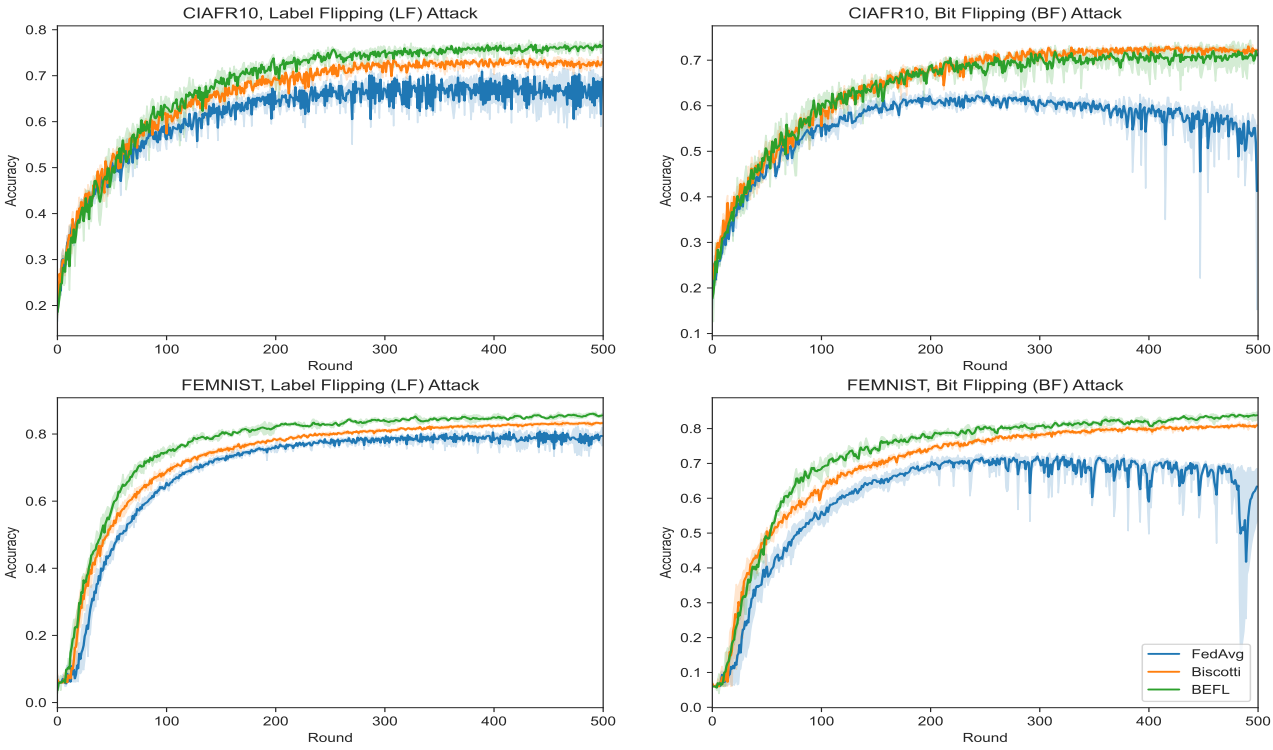

Fig. 4. Test accuracy of FedAvg, Biscotti and BEFL over CIFAR10 and FEMNIST datasets when adversaries perform LF and BF attacks. Curves are averages over 3 random trials, shaded regions represent 95% CIs

5.2 Resistance to poisoning attacks

In this section, we evaluate BEFL's performance against both data poisoning and model poisoning attacks. Data poisoning happens during the data collection phase. Malicious clients inject data samples into the training dataset. One common data poisoning attack in the FL scenario is the Label-Flipping (LF) [3] attack in which adversaries replace the targeted label with the desired one. In model poisoning attacks, adversaries adjust the training model directly and tailor its output to have a similar distribution with the correct model updates. Bit-Flipping (BF) attack [43] (also known as sign-flipping) is one of the common model poisoning attacks in which adversaries send the negative of the gradients to the master. We launched LF and BF attacks to both the CIFAR10 and FEMNIST tasks with 20% clients being malicious in each training round. In particular, malicious clients flipped the label l_c to $9 - l_c$ in CIFAR10 and l_f to $61 - l_f$ in FEMNIST for the LF attack and multiplied by -1 their gradients before compression in the BF attack. Uncompressed model updates were uploaded when performing FedAvg and Biscotti aggregation. As presented in Fig. 4, BEFL is resilient to both LF and BF attacks as conventional FedAvg struggles to converge. BEFL shows superior performance than Biscotti in FEMNIST task under both attacks. The accuracy achieved by BEFL is higher than that of Biscotti. For CIFAR10 task, it also performed better when adversaries launching LF attack. The performance of BEFL and Biscotti under BF attack is comparable as they achieved nearly the same accuracy.

5.3 Performance and scalability

In this section, we evaluate the overhead of each stage in BEFL and investigate the effect of byzantine nodes in the blockchain network. We also measure the performance of BEFL when scaling up committee size and joining of more nodes.

Overhead breakdown: To measure the overhead of each main stage of our design, we simulated BEFL over a varying number of blockchain nodes with varied committee sizes. We captured the amount of time spent in major stages: 1) committee constitution: blockchain nodes run the Sortition algorithm to constitute the committee; 2) candidate block generation: blockchain nodes (except committee members) collect model updates and aggregate them following the BEFL aggregation rule to generate a block; 3) voting: committee members vote for the candidate block and 4) verified block propagation: the distribution of the verified block to each node. As shown in Table 2, we deployed BEFL with 100 and 200 nodes with committee sizes of 15 and 30. When $K = 30$, the probability of not having enough candidate committee members is smaller than 2.07×10^{-9}, according to equation (4). The values recorded in the table are the averaged time (seconds) of 50 training rounds. Voting takes the longest time among the four stages and it doubles with the committee size since there are twice as many "yes" votes required to confirm the legitimacy of candidate block. The voting time needed in FEMNIST task is lower than that of CIFAR10 task since the compressed model size of FEMNIST update is smaller than CIFAR10's, thus it takes less time for transmitting candidate block to each committee member. The time needed for committee constitution and verified

TABLE 2
Breakdown of time in different phases of BEFL under different settings when processing CIFAR10 and FEMNIST tasks

	CIFAR10				FEMNIST			
("number of blockchain nodes", "committee size")	(100, 15)	(100, 30)	(200,15)	(200,30)	(100, 15)	(100, 30)	(200,15)	(200,30)
committee constitution (second)	0.115	0.207	0.170	0.256	0.115	0.185	0.154	0.257
candidate block generation (second)	7.719	7.592	7.449	7.562	7.233	7.221	7.445	7.492
voting (second)	**92.448**	**185.005**	**93.899**	**186.360**	**80.272**	**160.619**	**81.568**	**162.968**
verified block propagation (second)	0.312	0.587	0.609	1.180	0.298	0.570	0.632	1.139
total time (second)	100.595	193.391	102.128	195.358	87.918	168.595	89.800	171.857

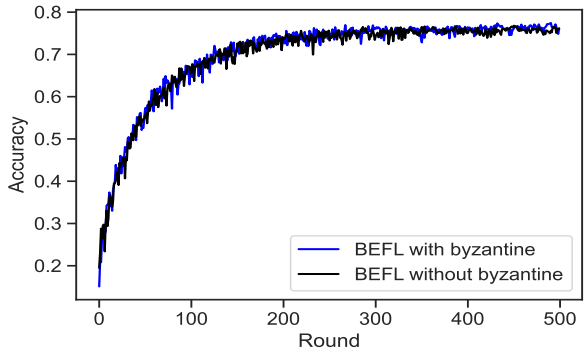

Fig. 5. Test accuracy of BEFL under honest clients setting for CIFAR10 task

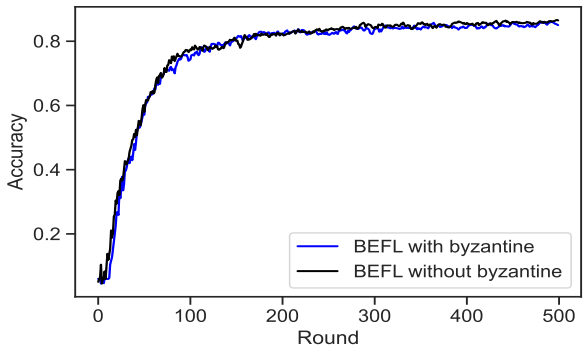

Fig. 6. Test accuracy of BEFL under honest clients setting for FEMNIST task

block propagation is almost the same in the two tasks, both remain very low. The committee constitution time doubles with the committee size, while propagation time doubles with the number of blockchain nodes.

Security analysis: The safety of BEFL is guarded by the committee-based consensus protocol, but proving this experimentally requires testing all possible attack strategies which is infeasible. We measure the resistance of BEFL to byzantine blockchain nodes with the specified attacking strategy: the malicious nodes actively participate in both committee constitution and candidate block generation. If the malicious node gets the chance to be a committee member, it votes "No" to the correct candidate block and votes "Yes" to the incorrect one. For generating the candidate block, instead of aggregating local model updates following the secure aggregation protocol, malicious nodes perform a Gaussian attack by setting random values following the standard Gaussian distribution as global model parameters. We assigned $\frac{1}{3}$ (the maximum byzantine fault tolerance in a distributed system) of blockchain nodes to be malicious and colluded together to mislead the learning process. As shown in Fig. 5 and 6, the performance of BEFL was unharmed by these byzantine nodes, demonstrating that BEFL is robust to this kind of attack scenario.

Scalability analysis: To measure the scalability of BEFL, we varied committee size with a fixed 100 blockchain nodes, and varied the number of blockchain nodes with a fixed committee size of 15. We reran CIFAR10 and FEMNIST tasks under different BEFL system settings for 50 training rounds. As shown in Fig. 7, the average total time per verified block generation and distribution grows almost linearly with the committee size as the dominant voting time increases. The

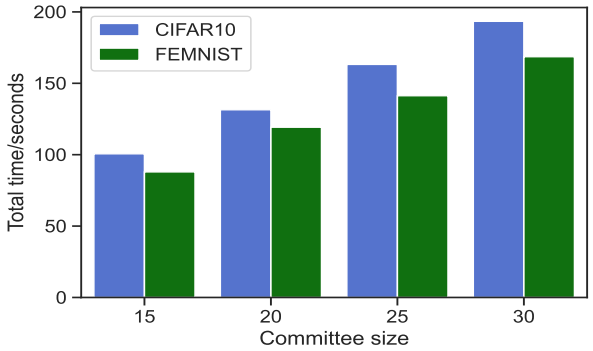

Fig. 7. The averaged total time of per verified block generation and distribution with varied committee size for CIFAR10 and FEMNIST tasks

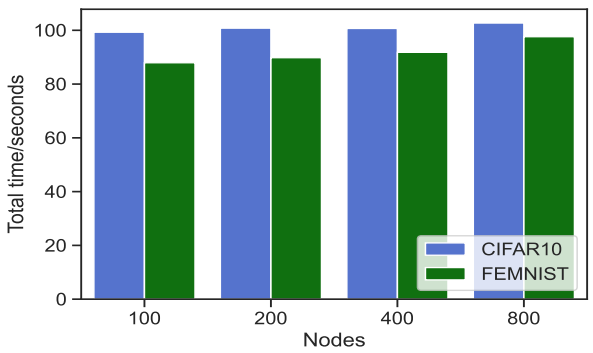

Fig. 8. The averaged total time of per verified block generation and distribution with varied blockchain network size for CIFAR10 and FEMNIST tasks

overall time for generating and distributing a valid block for the FEMNIST task is shorter than that of the CIFAR10 as the smaller transmitted model updates in the network. Although the larger the committee size, the more certainty of having enough candidate committee members in each committee constitution phase, and the higher credibility of the generated block as more agreements obtained, the additional communication overhead for reaching consensus on candidate block is non-negligible. Fig. 8 shows the performance of BEFL as the network size grows. It could be seen that the total time of block finalization and distribution barely increases in CIFAR10 task and the increment is almost negligible in FEMNIST task when the number of blockchain nodes increases from 100 to 800. Instead of saving the full shared global model and local model updates in a block, the recorded IPFS address and the removal of unnecessary stale model updates reduces the block size significantly, thus enabling fast block transmission in the network.

Considering the wireless edge execution environment of BEFL, we further monitored and calculated the resource consumption when simulating BEFL. The block size of BEFL is around 20KB for the integration of IPFS, making the block transmission and storage cost stay at a very low level, e.g., 1 million blocks would occupy 19.07 GB. The average peak computing power used for each blockchain node is around 3.43 GigaFLOPS (Floating point operations per second) in FEMNIST and 2.79 GigaFLOPS in CIFAR10. The bandwidth usage is the same in the two tasks, 0.014 kbps for sending and 0.012 kbps for receiving at each blockchain node. These costs of 19.07 GB, 3.43 and 2.79 GigaFLOPS, 0.014 and 0.012 kbps could be easily satisfied by a typical edge server, e.g., one with 2.3 GHz quad-core Intel i5 CPU (the computing performance is therefore 147.2 GigaFLOPS), 8 GB RAM, and 256GB storage [44], and the bandwidth is often above 1 Gb/s [45].

6 Conclusion

In this work, we proposed BEFL, a novel lightweight blockchain system for secure, efficient and practical FL at the wireless edge. To reduce the communication cost for participating clients and defend against malicious ones that send poisonous model updates, we proposed the communication-efficient MI-guarded training scheme exploiting PowerSGD. The reliability of the global model is ensured by a committee-based consensus protocol where the aggregation execution is verified. To reduce the communication cost of block propagation, and the storage cost of maintaining the blockchain, BEFL incorporates the IPFS to store the global model with its address recorded in the block. The performance results show that BEFL achieves communication efficiency with byzantine robustness, reliability of global model provided by the blockchain system, and high scalability enabled by VRF and IPFS.

Acknowledgment

This work was supported in part by EPSRC New Horizons Grant No. EP/X019160/1, UKRI Grant No. EP/X038866/1, and Horizon EU Grant No. 101086159. For the purpose of open access, the author has applied a 'Creative Commons Attribution (CC BY) licence to any Author Accepted Manuscript version arising.

References

[1] J. Park, S. Samarakoon, M. Bennis, and M. Debbah, "Wireless network intelligence at the edge," *Proceedings of the IEEE*, vol. 107, no. 11, pp. 2204–2239, 2019.

[2] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 118–128.

[3] C. Fung, C. J. Yoon, and I. Beschastnikh, "Mitigating sybils in federated learning poisoning," *arXiv preprint arXiv:1808.04866*, 2018.

[4] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *International Conference on Machine Learning*. PMLR, 2018, pp. 5650–5659.

[5] R. Guerraoui, S. Rouault *et al.*, "The hidden vulnerability of distributed learning in byzantium," in *International Conference on Machine Learning*. PMLR, 2018, pp. 3521–3530.

[6] K. Pillutla, S. M. Kakade, and Z. Harchaoui, "Robust aggregation for federated learning," *arXiv preprint arXiv:1912.13445*, 2019.

[7] M. Shayan, C. Fung, C. J. Yoon, and I. Beschastnikh, "Biscotti: A blockchain system for private and secure federated learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 7, pp. 1513–1525, 2020.

[8] Y. Li, C. Chen, N. Liu, H. Huang, Z. Zheng, and Q. Yan, "A blockchain-based decentralized federated learning framework with committee consensus," *IEEE Network*, vol. 35, no. 1, pp. 234–241, 2020.

[9] H. Chen, S. A. Asif, J. Park, C.-C. Shen, and M. Bennis, "Robust blockchained federated learning with model validation and proof-of-stake inspired consensus," *arXiv preprint arXiv:2101.03300*, 2021.

[10] Y. Qu, S. R. Pokhrel, S. Garg, L. Gao, and Y. Xiang, "A blockchained federated learning framework for cognitive computing in industry 4.0 networks," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 4, pp. 2964–2973, 2020.

[11] M. Xu, Z. Zou, Y. Cheng, Q. Hu, D. Yu, and X. Cheng, "Spdl: A blockchain-enabled secure and privacy-preserving decentralized learning system," *IEEE Transactions on Computers*, 2022.

[12] S. Yuan, B. Cao, M. Peng, and Y. Sun, "Chainsfl: Blockchain-driven federated learning from design to realization," in *2021 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2021, pp. 1–6.

[13] J. Kang, Z. Xiong, D. Niyato, Y. Zou, Y. Zhang, and M. Guizani, "Reliable federated learning for mobile networks," *IEEE Wireless Communications*, vol. 27, no. 2, pp. 72–80, 2020.

[14] J. Liang, S. Li, W. Jiang, B. Cao, and C. He, "Omnilytics: A blockchain-based secure data market for decentralized machine learning," *arXiv preprint arXiv:2107.05252*, 2021.

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

[16] T. Vogels, S. P. Karimireddy, and M. Jaggi, "Powersgd: Practical low-rank gradient compression for distributed optimization," 2019.

[17] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Blockchain and federated learning for privacy-preserved data sharing in industrial iot," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4177–4186, 2019.

[18] L. Feng, Y. Zhao, S. Guo, X. Qiu, W. Li, and P. Yu, "Blockchain-based asynchronous federated learning for internet of things," *IEEE Transactions on Computers*, 2021.

[19] Y. Zhao, J. Zhao, L. Jiang, R. Tan, D. Niyato, Z. Li, L. Lyu, and Y. Liu, "Privacy-preserving blockchain-based federated learning for iot devices," *IEEE Internet of Things Journal*, vol. 8, no. 3, pp. 1817–1829, 2020.

[20] J. Weng, J. Weng, J. Zhang, M. Li, Y. Zhang, and W. Luo, "Deepchain: Auditable and privacy-preserving deep learning with blockchain-based incentive," *IEEE Transactions on Dependable and Secure Computing*, 2019.

[21] L. Feng, Y. Zhao, S. Guo, X. Qiu, W. Li, and P. Yu, "Bafl: A blockchain-based asynchronous federated learning framework," *IEEE Transactions on Computers*, vol. 71, no. 5, pp. 1092–1103, 2021.

[22] Y. Qu, L. Gao, T. H. Luan, Y. Xiang, S. Yu, B. Li, and G. Zheng, "Decentralized privacy using blockchain-enabled federated learning in fog computing," *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 5171–5183, 2020.

[23] S. Guo, K. Zhang, B. Gong, L. Chen, Y. Ren, F. Qi, and X. Qiu, "Sandbox computing: A data privacy trusted sharing paradigm via blockchain and federated learning," *IEEE Transactions on Computers*, 2022.

[24] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies," in *Proceedings of the 26th symposium on operating systems principles*, 2017, pp. 51–68.

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

[26] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich *et al.*, "Hyperledger fabric: a distributed operating system for permissioned blockchains," in *Proceedings of the thirteenth EuroSys conference*, 2018, pp. 1–15.

[27] J. Benet, "Ipfs-content addressed, versioned, p2p file system," *arXiv preprint arXiv:1407.3561*, 2014.

[28] S. Micali, M. Rabin, and S. Vadhan, "Verifiable random functions," in *40th annual symposium on foundations of computer science (cat. No. 99CB37039)*. IEEE, 1999, pp. 120–130.

[29] H.-Y. Chen and W.-L. Chao, "Fedbe: Making bayesian model ensemble applicable to federated learning," *arXiv preprint arXiv:2009.01974*, 2020.

[30] J. R. Douceur, "The sybil attack," in *International workshop on peer-to-peer systems*. Springer, 2002, pp. 251–260.

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

[32] P. Xiao, S. Cheng, V. Stankovic, and D. Vukobratovic, "Averaging is probably not the optimum way of aggregating parameters in federated learning," *Entropy*, vol. 22, no. 3, p. 314, 2020.

[33] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. USA: Wiley-Interscience, 2006.

[34] M. P. Uddin, Y. Xiang, X. Lu, J. Yearwood, and L. Gao, "Mutual information driven federated learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 7, pp. 1526–1538, 2020.

[35] L. LAMPORT, R. SHOSTAK, and M. PEASE, "The byzantine generals problem," *ACM Transactions on Programming Languages and Systems*, vol. 4, no. 3, pp. 382–401, 1982.

[36] DataDog, "Datadog/go-python3: Go bindings to the cpython-3 api." [Online]. Available: https://github.com/DataDog/go-python3

[37] Ipfs, "Ipfs/go-ipfs-api: The go interface to ipfs's http api." [Online]. Available: https://github.com/ipfs/go-ipfs-api

[38] Coniks-Sys, "Coniks-sys/coniks-go: A coniks implementation in golang." [Online]. Available: https://github.com/coniks-sys/coniks-go

[39] S. Caldas, S. M. K. Duddu, P. Wu, T. Li, J. Konečnỳ, H. B. McMahan, V. Smith, and A. Talwalkar, "Leaf: A benchmark for federated settings," *arXiv preprint arXiv:1812.01097*, 2018.

[40] A. Li, J. Sun, B. Wang, L. Duan, S. Li, Y. Chen, and H. Li, "Lotteryfl: Personalized and communication-efficient federated learning with lottery ticket hypothesis on non-iid datasets," *arXiv preprint arXiv:2008.03371*, 2020.

[41] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[42] T.-M. H. Hsu, H. Qi, and M. Brown, "Measuring the effects of non-identical data distribution for federated visual classification," *arXiv preprint arXiv:1909.06335*, 2019.

[43] S. P. Karimireddy, L. He, and M. Jaggi, "Learning from history for byzantine robust optimization," in *International Conference on Machine Learning*. PMLR, 2021, pp. 5311–5319.

[44] R. Ullah, D. Wu, P. Harvey, P. Kilpatrick, I. Spence, and B. Varghese, "Fedfly: Toward migration in edge-based distributed federated learning," *IEEE Communications Magazine*, vol. 60, no. 11, pp. 42–48, 2022.

[45] D. Kimovski, R. Mathá, J. Hammer, N. Mehran, H. Hellwagner, and R. Prodan, "Cloud, fog, or edge: Where to compute?" *IEEE Internet Computing*, vol. 25, no. 4, pp. 30–36, 2021.

Rui Jin received the BEng degree in information security from University of Science and Technology Beijing (USTB), China in 2019. She is currently working toward the PhD degree in computer science at the University of Exeter. Her research interests include federated learning, applied machine learning, blockchain, and mobile edge computing.

Jia Hu received the BEng and MEng degrees in electronic engineering from the Huazhong University of Science and Technology, China, in 2006 and 2004, respectively, and the PhD degree in computer science from the University of Bradford, UK, in 2010. He is a senior lecturer of computer science at the University of Exeter. His research interests include edge-cloud computing, resource optimization, applied machine learning, and network security.

Geyong Min received the BSc degree in computer science from the Huazhong University of Science and Technology, China, in 1995, and the PhD degree in computing science from the University of Glasgow, United Kingdom, in 2003. He is a professor of high performance computing and networking with the Department of Computer Science within the College of Engineering, Mathematics and Physical Sciences at the University of Exeter, United Kingdom. His research interests include computer networks, wireless communications, parallel and distributed computing, ubiquitous computing, multimedia systems, modeling and performance engineering.

Jed Mills is a Computer Science Ph.D. student in the Department of Computer Science at the University of Exeter, UK. He received a B.Sc. in Natural Science from the University of Exeter in 2018. His research interests include machine learning, federated learning, and mobile edge computing.