Bift: A Blockchain-Based Federated Learning System for Connected and Autonomous Vehicles

Ying He, *Member, IEEE*, Ke Huang, Guangzheng Zhang, F. Richard Yu, *Fellow, IEEE*, Jianyong Chen, and Jianqiang Li

***Abstract*—Machine learning (ML) algorithms are essential components in autonomous driving. In most existing connected and autonomous vehicles (CAVs), a large amount of driving data collected from multiple vehicles are sent to a central server for unified training. However, data privacy and security have become crucial during the data-sharing process. Federated learning (FL) for data security has arisen nowadays, and it can improve the data privacy of distribute machine learning. However, the malicious attackers can still be able to attack the training process. Due to the complete reliance on the central server, FL is very fragile. To address the above problem, we propose Bift: 1) a fully decentralized ML system combined with FL and 2) blockchain to provide a privacy-preserving ML process for CAVs. Bift enables distributed CAVs to train ML models locally using their own driving data and then to upload the local models to get a better global model. More importantly, Bift provides a consensus algorithm named Proof of Federated Learning to resist possible adversaries. We evaluate the performance of Bift and demonstrate that Bift is scalable and robust, and can defend against malicious attacks.**

***Index Terms*—Blockchain, connected and autonomous vehicles (CAVs), consensus algorithm, federated learning (FL), off-chain storage.**

I. Introduction

DATA play an important role in the development of connected and autonomous vehicles (CAVs). The powerful sensors on CAVs have access to an unprecedented amount of data [1]–[3]. Driving data are essential for model training in machine learning (ML) algorithms, which are crucial for most autonomous driving techniques (ADTs) [4]–[6].

Traditionally, the data are sent back to a central server for learning autonomous driving algorithms, mostly the ML models [7]–[10]. However, it is difficult for this approach to guarantee the privacy and security of the driving data collected from individual vehicles during data transmission and data sharing [11]–[13]. Moreover, a large amount of data are held by a few companies, which seriously hinders the further development of ADTs. The privacy and security of the data exchange seem to be an irreconcilable contradiction.

Federated learning (FL) is proposed to meet the problem of data privacy protection [14]: clients train a model locally without exposing their local data, and then they update the local model parameters to a central server to help build a new powerful global FL model [15]. Since clients never upload their local data, FL improves user privacy. FL is also widely used in the field of IoTs. Imteaj *et al.* [16] introduced the application of FL in the field of resource-constrained IoT devices.

Despite these benefits, using FL introduces some new threats as follows.

1) Malicious clients can easily attack the FLs process through poisoning attacks, e.g., a dishonest client could upload incorrect gradients or parameters to damage the global model [17]–[19].
2) The central server, which collects and aggregates model parameters, is prone to a single point of failure, causing the service to collapse. Once the single server is broken, it will cause a massive accident.
3) It is difficult for the central server to withstand the massive throughput from many clients, and CAVs on the road have needs to transfer data to each other, e.g., the high-precision map that every CAV collects at every moment.
4) It is difficult to evaluate the contribution of a specific node to the model. In the real world, a client is reluctant to contribute his own data without incentive.

To meet the security and privacy of federated learning, we propose Bift, a fully decentralized ML system with blockchain, interplanetary file system (IPFS), and federated learning [15] to provide a privacy-preserving ML process for CAVs. Bift provides the ability to resist malicious attacks based on blockchain and a unique consensus protocol proof of federated learning (PoFL). In addition, the system framework we designed shows good performance. Our primary contributions are summarized as follows.

1) Compared with other blockchain applications [20] implemented with Ethereum or other public blockchain frameworks, Bift is designed and coded by ourselves. Bift ensures decentralized security through blockchain

Manuscript received August 12, 2021; revised November 4, 2021; accepted November 29, 2021. Date of publication December 14, 2021; date of current version July 7, 2022. This work was supported in part by the Natural Science Foundation of China (NSFC) under Grant 62002238 and Grant 61836005; in part by the Distinguished Young Talents in Higher Education of Guangdong under Grant 2019KQNCX125; in part by the Guangdong Basic and Applied Basic Research Foundation under Grant 2019A1515110429; in part by the National Key Research and Development Program of China under Grant 2020YFA0908700; in part by the Tencent "Rhinoceros Birds"—Scientific Research Foundation for Young Teachers of Shenzhen University; and in part by the Guangdong Pearl River Talent Recruitment Program under Grant 2019ZT08X603. *(Corresponding author: F. Richard Yu.)*

Ying He, Ke Huang, F. Richard Yu, Jianyong Chen, and Jianqiang Li are with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China (e-mail: richard.yu@carleton.ca).

Guangzheng Zhang is with the Computer Science and Engineering, Jocobs School of Engineering, San Diego, CA 92093 USA.

Digital Object Identifier 10.1109/JIOT.2021.3135342

TABLE I
MAIN ABBREVIATIONS

Abbreviation	Explanation
CAV	Connected and Autonomous Vehicle
ML	Machine Learning
ADTs	Autonomous Driving Techniques
FL	Federated Learning
PoFL	Proof of Federated Learning
NIST	National Institute of Standards and Technology
PoW	Proof of Work

and encourages CAVs to actively train and upload their local models through feedback stakes.

2) We propose a unique consensus protocol named PoFL that combines the traditional hash-based protocol Proof of Work (PoW) and three kinds of aggregation rules. By modifying these three rules from the FedSGD algorithm to the FedAvg algorithm, Bift can reduce the communication and computing overhead of the FL process. Moreover, even in the presence of malicious nodes, Bift can achieve good performance.
3) We build a distributed file storage and access module in Bift. By applying IPFS [21], a peer-to-peer hypermedia protocol, we make data exchange in CAVs more robust and efficient. Compared with the traditional C/S architecture, CAVs in Bift can get rid of the dependence of the central server, and have a faster convergence speed and a more stable network environment.

We evaluate Bift by deploying 30 peers in docker containers [22] and study its performance, scalability, and ability to with stand different attacks. The results show that Bift is more robust and has better network performance than centralized service. Furthermore, we show that Bift can prevent malicious upload models by using the PoFL algorithm.

The remainder of this article is organized as follows. In Section II, we present the background of this article. In Section III, we present the detailed design of Bift. The PoFL algorithm is described in Section IV. Section V presents the distributed storage module in detail. The experimental results are presented in Section VI. Finally, Section VII concludes this article.

II. BACKGROUND

In this section, we introduce blockchain, federated learning, byzantine-robust aggregation rules, and IPFS. The main abbreviations are shown in Table I.

A. Blockchain

The blockchain is essentially a secure, decentralized, and nontamperable distributed database [23]. It was proposed in 2008 by Nakamoto, which combined previous technologies, such as P2P network, cryptography, and timestamp [24]. The main feature of the blockchain is that it allows untrusted participants to communicate with each other and send status update messages in a secure manner without the participation of a fully trusted third party or authorized central node. Blockchain can solve the single point of failure, data security, and lower-than-attack problems in various fields, such as currency, data storage, and financial auditing [25]–[28]. For supply chain, [29] proposed an IoT-based blockchain framework to improve the pharmaceutical supply chain.

The consensus algorithm is the core component of blockchain. Clients in the blockchain need to achieve a consensus whenever a block is mined. PoW is the earliest consensus algorithm, and it consumes computing power to reach consensus. Besides, there are many other consensus algorithms, such as Proof of Stake (PoS) in Ethereum, practical Byzantine fault tolerance (PBFT) in Hyperledge, and delegated PoS (DPoS) in EOS.

Blockchain has been widely applied in solving the single point of failure and security problems in FL [30], [31]. FLChain [32] has been proposed to upload and trade FL models in a distributed market. FedCoin [33] uses the Shapley value algorithm to incentivize users to participate in the FL process. In [34], a blockchain-based FL architecture is proposed by applying blockchain instead of the central server.

B. Federated Learning

As a new type of basic artificial intelligence technology for data islands and privacy protection, FL was first proposed by Google to solve the problem of local update models of Android mobile phone terminal users [15].

FL allows different users who have different data sets to collectively obtain the benefits of shared models trained from these rich data without losing control of it. The ML tasks are carried out by a consortium of devices (which we called clients), and then a central server aggregates the upload data. The final model can benefit from all local training data sets, and a client will never upload its data set to the server.

Generally, for an ML problem whose training model has n parameters, the optimization problem is to minimize $f(\beta)$ where

$$f(\beta) = \frac{1}{n}\sum_{i=1}^{n} f_i(\beta) \quad f_i(\beta) = \mathcal{L}(x_i, y_i; \beta_i). \tag{1}$$

$\mathcal{L}$ is the loss of the prediction on example(x_i, y_i) where the model parameters are β. In the scenario of FL, assume there are C clients in the FL process, and the local dataset of C_i client is $\mathcal{D}_i$. The problem can be rewritten as minimizing $f(\beta, \mathcal{D}_i)$ where

$$f(\beta, \mathcal{D}_i) = \sum_{c=1}^{c} \frac{n_c}{n} F_c(\beta, \mathcal{D}_i) \quad F_c(\beta, \mathcal{D}_i) = \frac{1}{n_k}\sum f_i(\beta, \mathcal{D}_i). \tag{2}$$

$f_i(\beta, \mathcal{D}_i)$ is the objective function for C_i who have local data set $\mathcal{D}_i$ and indicates how well the parameters β model the local training data set on C_i.

In FL, it has a central server to obtain a global model through aggregating local models uploaded from the C clients. Moreover, the training process needs several iterations, and clients and server communicate with each other in each iteration. Specifically, FL performs the following three steps in each iteration.

1) The central server sends the current global model parameters to each client C_i.
2) When C_i receives the global model parameters, it starts training another unique local model based on the global model and its local data set $\mathcal{D}_i$, and then it sends the updated model parameters back to the central server.
3) When the central server collects enough local models from different clients, it aggregates them to obtain a new global model via a specific aggregation algorithm.

There are two main aggregation algorithms in FL: 1) FederatedSGD [35] and 2) FederatedAvg [15]. The two algorithms have their own advantages and disadvantages, and we will discuss them in Section IV.

C. Byzantine-Robust Aggregation Rules

In original aggregation algorithms, such as FedAvg and FedSGD, the central server aggregates all local updates without inspecting them. The two naive algorithms are widely used in nonadversarial settings [14], [36]. But in the real world, especially in a federation that is extremely decentralized, there may be malicious clients that intend to destroy FL model [37]–[42], so the naive algorithms are not robust under the CAVs scenarios. Therefore, some Byzantine-robust aggregation rules are proposed to resist malicious clients in FL.

Trimmed Mean [43]: It transplants the trimmed mean algorithm in statistics to FL and treats each gradient of model parameters independently. Specifically, for each ith model parameter, the central server sorts the gradient of the ith model parameter from all j local models, and then the result gradient list is $G_{1i}, G_{2i}, G_{3i}, \ldots, G_{ji}$, where G_{ji} is the ith gradient of the jth local model. We define a static number n, then remove the smallest and largest n gradients, and compute the mean of the remaining gradients as

$$\overline{G_i} = \frac{G_{(n+1)i} + G_{(n+2)i} + \cdots + G_{(j-n)i}}{j - 2n}. \tag{3}$$

Suppose at most m malicious clients in this system, the trimmed algorithm can achieve *order-optimal* error rate when $m \leq n \leq (j/2)$, and the order-optimal error rate is $\widetilde{O}([m/j\sqrt{d}]+[1/\sqrt{jd}])$, where d is the dimension of the model (assuming the dimensions of all models are the same).

Krum [44]: Krum chooses one of the j local models, which is most similar to others as the global model. Krum rejects malicious updates by discarding them. The influence is limited even if the selected local model is from a malicious client, because the difference between the selected one with other honest model is small. Suppose at most m malicious clients in this system and n local models are updated to the central server, Krum computes the Euclidean distance between $n - m - 2$ gradients, and then selects the one to the other. This algorithm attempts to find out f malicious models in n updates such that $2f + 2 < n$.

Bulyan [45]: The Euclidean distance between two local models could be greatly affected by single gradient, so Krum could be incorrect (e.g., Krum will throw away it when a honest client updates its model, but has one single abnormal gradient). To address this issue, Bulyan proposed combining both Krum and Trimmed mean. Specifically, Bulyan aggregates gradients by the following two steps: 1) Bulyan iteratively selects $\beta(\beta \leq n - 2m)$ local updated gradients via Krum and 2) Bulyan uses the Trimmed mean algorithm to aggregate these β gradients and then generates the final model.

D. InterPlanetary File System

The IPFS is a peer-to-peer hypermedia protocol designed to make the Web faster, safer, and more open [46]. IPFS is actually a P2P file exchange technology, and it aims to surpass the HTTP protocol to build a better Web.

The principle of IPFS is to replace domain-based addresses with content-based addresses, that is, what users are looking for is not an address but content stored in a certain place. There is no need to verify the identity of the sender, but only the hash of the content.

IPFS has many excellent features compared to the HTTP protocol, such as: 1) IPFS permanently saves data and provides version traceability, data will be broken down into multiple pieces and sent to different computers, and the same data will be copied to multiple computers. IPFS implements a concept similar to Git, so users can retroactive the version of files; 2) IPFS improves speed and reduces bandwidth waste, and this protocol stores files in fragments on various nodes. When users are looking for a file, the system will select the storage node closest to the file, then transfer it through the IPFS protocol, and integrate it into a complete file for them. Because all files are transferred through the closest node, this not only greatly improves the transfer speed but also saves a lot of bandwidth; and 3) as a decentralized cloud storage system, IPFS effectively prevents us from storing some crucial files on a central server, facing the problem of shutting down at any time.

The network topology of IPFS is similar to blockchain, and due to the limitation of the block size, researchers usually combine IPFS and blockchain to use IPFS as the storage layer of the blockchain [47], [48].

III. System Design

In this section, we present the proposed system named Bift, which are used by CAVs in training ML models in a distributed manner. In this process, Bift provides safety, robustness, and high efficiency.

A. System Overview

Fig. 1 shows an overview of our Bift system architecture. Bift implements efficient and safety peer-to-peer FL based on blockchain. For this process, Bift's design has the following goals.

1) Generate an optimal global model, whose performance is about the same as the origin FL without adversaries, even in a limited malicious environment (up to 30% of malicious nodes).
2) Poisoning is prevented by our Proof of FL consensus algorithm, which is combined with multiple Byzantine-robust aggregation rules.

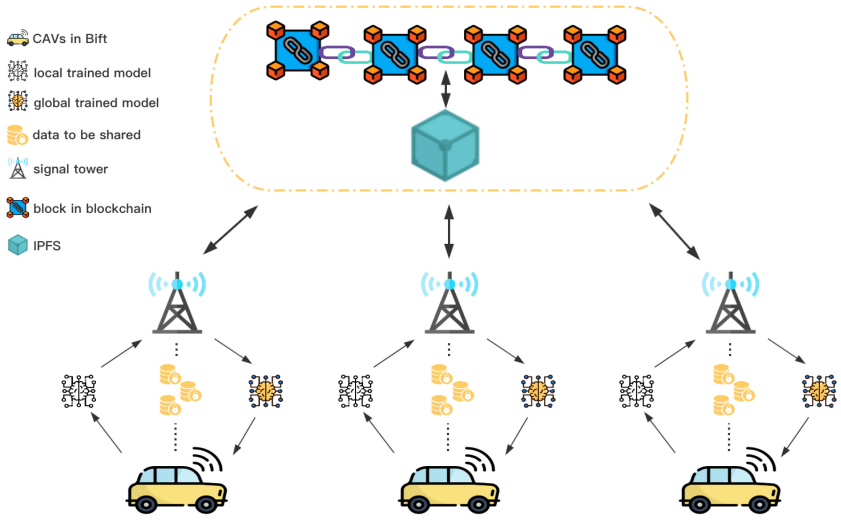

Fig. 1. Overview of Bift.

3) No need for a central server, which may collapse at any time, and is controlled by a single company. Individuals are all equal in our system.
4) CAVs in our system have the ability to share data with highly efficient use of network bandwidth and low latency via a peer-to-peer file exchange system.

Bift solves these goals through a blockchain-based method that we will describe in this section.

CAVs join Bift and collaboratively train a global FL model. As shown in Fig. 1, each block in blockchain contains a record of the last FL iteration. Each CAV can get information of the latest model, such as the *model_id*, *ipfs_hash* of model, the required *total_iteration*, the *current_iteration*, and the *accuracy* of the current global model from blockchain. CAV can upload a training task when the last task has been completed, and Bift will broadcast the block, which contains the information of FL training task to other CAVs. Once a CAV receives the broadcast, it will start training a local ML model using its own data and computing power. When enough CAVs have uploaded their own local model via IPFS, Bift will aggregate them through the PoFL algorithm and send the aggregated global model back to CAVs, and then CAVs start the next iteration of this task until the model is convergent. To be specific, we clarify the iteration process of Bift in four detailed steps as follows.

Step 1 (Proposing a Training Task): A single CAV requests to propose a training task when the last task has been completed (total_iter − current_iter = 0), and then it pushes the initial model to IPFS and gets *ipfs_hash* of this model (the whole model is too big to store it in the blockchain). Finally, the information of this model will be uploaded to blockchain and broadcasted to other CAVs.

Step 2 (Training Local Model): Once a CAV has received the newest training task, it will get the model information from blockchain and get the initial model through IPFS, then load this initial model, and train it using its own driving data and computing power. After the local training is finished, the CAV will obtain the local model and the model accuracy, and then upload them to the blockchain.

Step 3 (Aggregating Local Models and Give Rewards): Bift waits for a certain period of time, and when there are enough local models uploaded within a certain period of time, Bift will collect these local models and discard other delayed uploads. Then, Bift filters out model parameters from all models, discards malicious model and aggregates honest ones with PoFL, and generates the newest global model. Bift will upload the global model to the blockchain with setting current_iter = current_iter + 1. Finally, Bift adds up all CAVs that contributed to this iteration of training and gives them rewards by transferring some stacks or coins.

Step 4 (Iterating Train Local Model): When CAVs receive the broadcast of the newest global model from Bift, they will judge whether *current_iter* is equal to *total_iter*; if it is, all CAVs know that the current FL task is completed and then wait for the next task. Otherwise, they continue to iteratively train the local model on the basis of the newest global model by repeating steps 2 and 3.

B. Blockchain

A consortium blockchain is used in Bift to take the place of the centralized server in FL. Due to the insufficient scalability of the existing blockchain system, it is not easy to modify the blockchain logic on the existing basis to meet the needs of FL, so we develop the blockchain system by ourselves with Python.

The block struct in Bift is shown in Fig. 2. A block is divided into two parts: 1) body and 2) header. The block header mainly includes six fields: 1) *index*; 2) *hash*; 3) *nonce*; 4) *previous hash*; 5) *timestamp*; and 6) *model flag*. When a new block comes in, the blockchain will verify the legitimacy of the new block that compares to the previous included block. Specifically, suppose the fields in the previous block are index_p, hash_p, nonce_p, previous_hash_p, and timestamp_p, fields in the newly coming block are: index_c, hash_c, nonce_c, previous_hash_c, and timestamp_c, and the default difficulty is *difficulty*. One new block must meet the following constraints

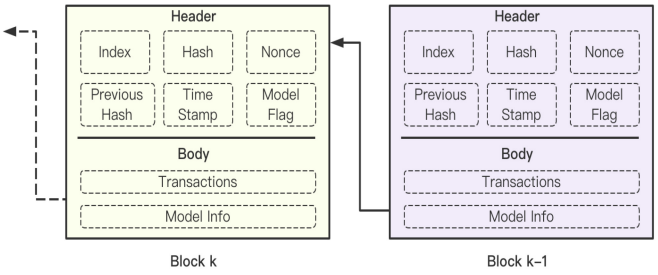

Fig. 2. Block struct in Bift.

TABLE II
EXPLANATIONS OF MODEL INFO

Field	Explanations
$model\_id$	The Identity of FL task
$init\_ipfs\_hash$	The ipfs_hash of this model
$total\_iter$	Total iteration the FL task needs
$current\_iter$	Current iteration now
$from\_addr$	Which address the model is from
$accuracy$	The accuracy of this uploaded model

before it can be included in the blockchain, or it will be considered illegal and discarded:

$$\begin{aligned} \text{index}_c &= \text{index}_p + 1 \\ \text{previous\_hash}_c &= \text{hash}_p \\ \text{timestamp}_c &> \text{timestamp}_p \end{aligned} \tag{4}$$

and

$$\text{SHA}_{256}(\text{hash}_c + \text{nonce}_c) < \text{difficulty}. \tag{5}$$

Among them, SHA_{256} is a common secure hash algorithm designed by the National Institute of Standards and Technology (NIST) and it calculates a unique 256-bit summary for each file. To reach consensus, each block must be constantly changing its value of nonce to match the *difficulty* condition. This process is called PoW, which is widely used in blockchain systems [49]. Bift includes PoW in its PoFL, and we will describe them in detail in Section IV.

The block body includes *transactions* and *model info*. *transactions* contain transaction information that is included in this block, and a transaction information contains where the transaction is from, to which address, and the coin number to exchange. Once a transaction is included in the blockchain, the balance of both the receiver and the sender will change accordingly.

The *model info* in the block body indicates the current model information. At the beginning of each FL iteration, each CAV gets the model from last block and then starts local training based on the model info. At the end of each FL iteration, Bift will generate the *model info* according to the result of PoFL and push it into the block. Table II presents the explanation of each field in *model info*.

TABLE III
COMPARISON OF BIFT AND OTHER SOLUTIONS

Solution\Ability	Privacy Preserving	Storage System	Incentive Mechanism
Bift	✓	✓	✓
FLChain	✓	–	✓
OpenMined	✓	–	–
DeepChain	✓	–	✓
Solution\Ability	**Robust Algorithm**	**Gradient Encryption**	**Distribution**
Bift	✓	–	✓
FLChain	–	–	✓
OpenMined	–	✓	–
DeepChain	–	–	✓

Besides, the blockchain in Bift provides an incentive mechanism for CAVs. Whenever a single CAV contributes to federated learning, Bift will reward it with a certain amount of stake (we initialize this value to 20). This stake can be freely traded between CAVs. In the future, it can even be used to trade pretrained models and anything else with each other. This incentive mechanism can encourage CAVs to actively train and upload their local models.

C. Comparison of Bift and Related Works

As we described in Section II, blockchain has been widely applied in FL. FLChain [30] has been proposed to upload and trade FL models in a distributed market. DeepChain [50] provides secure deep learning training and blockchain-based incentive. OpenMined [51] is an open-source community, which provides gradient encryption for FL.

Although some excellent works have been done on blockchain-based federated learning, most existing works focus on introducing blockchain to provide an incentive mechanism for federated learning. However, the ability to resist malicious attacks is very important because there may be malicious attackers in a distributed ML environment. When malicious attacks are not considered, the process of learning will be destroyed causing serious consequences.

We compare Bift and other solutions from the perspective of the abilities provided. As shown in Table III, we can see that Bift provides more comprehensive abilities than other solutions. Especially in terms of robustness algorithms, Bift performs better than other solutions.

IV. PROOF OF FEDERATED LEARNING

The consensus algorithm is the core component of blockchain, which is used to solve the consistency problem in distributed systems. There may be the following three consistency problems in a distributed system: 1) unreliable communication between nodes; 2) the processing of the node may be wrong, or even the node itself may be down at any time; and 3) malicious nodes. In order to make the global consensus reach a consistent result in such an environment, a consensus algorithm must be reliable and robust.

We divide the consensus problem in Bift into three problems as follows.

1) How can a *miner* (CAV that participates in model aggregating) aggregate the local uploaded FL models efficiently and robustly?

2) Which *miner* has the right to write his aggregate results into blockchain?
3) After the result links to the current blockchain, how to make the other CAVs recognize it?

To meet these three problems, a unique consensus algorithm named PoFL is proposed. PoFL uses *mixed aggregation algorithm*, *node selection*, and *chain fork* to solve these three problems. In this section, we will describe these three methods in detail.

A. Mixed Aggregation Algorithm

In pure blockchain systems, the consensus algorithm is only responsible for solving the consistency problem. However, in Bift, we must face the problem of how to aggregate local FL models efficiently and robust.

Efficiently Aggregate Algorithm: In the process of FL, communication will be the bottleneck because the bandwidth of network transmission is relatively small. The basic aggregation algorithm of FL is *FedSGD*, which requires each client c with local data set D_c and a fixed learning rate η to compute $g_c = \nabla F_k(\beta_t, D_c)$, where g_c is the average gradient on the local data at the current model β_t of client c. Then, the central server updates β_t by $\beta_{t+1} \leftarrow \beta_t - \eta \sum_{c=1}^{c}(n_c/n)g_c$. That is, in each step of gradient descent, each client must upload its result and then the central server aggregates it.

The *FedSGD* approach requires a huge number of rounds of communications to generate a good model (e.g., Ioffe and Szegedy trained MNIST for 50 000 steps [52]), and it is difficult to bear in a distributed environment. In order to improve *FedSGD*, *FedAvg* is proposed in [15]; FedAvg adds more computation to each client, and local data are trained by E times before updating to the central server and the server aggregates model parameters by $\beta_{t+1} \leftarrow \sum_{c=1}^{c}(n_c/n)\beta_{t+1}^{c}$.

In our Bift, we choose *FedAvg* as our base method to reduce the rounds of communications in FL. Moreover, all Byzantine-robust aggregation rules mentioned in Algorithm 2 are based on *FedAvg*. Especially, in the pure *FedAvg*, a C fraction of clients is selected in each round. However, in Bift, we select clients by the arriving time to the blockchain. Algorithm 1 shows the *FedAvg* aggregation algorithm in Bift.

Mixed Aggregation Algorithm: Because Bift is a decentralized system, we have no other methods to control user behavior. We must consider the presence of malicious nodes. Malicious nodes can intentionally upload fake model parameters (β in Algorithm 1) to prevent the model from converging or even affect the output of the model.

As we introduced in Section II, some Byzantine-robust aggregation rules are proposed to resist malicious nodes. However, these rules have their own limitations and can only defend against certain specific attack methods; hence, we combine the mentioned three rules into our PoFL to handle more specific situations. We modified the three algorithms mentioned earlier from FedSGD applicable to FedAvg applicable and combined them in Bift to defense attackers. The mixed aggregation algorithm is shown in Algorithm 2. In each iteration of aggregation, we randomly select one of the rules to execute. Especially, this algorithm filters out honesty values from many uploaded local models $\beta_1, \beta_2, \ldots, \beta_i$, and each β_i contains specific weights $w_{i1}, w_{i2}, \ldots, w_{ij}$ and finally, outputs an aggregated global model.

Algorithm 2 is actually a plug-in of Algorithm 1, which helps resist malicious nodes.

Algorithm 1 Federated Averaging in Bift

INPUT: c: The C clients are indexed by c
B: The local minibatch size
E: The number of local epochs
T: The total training round
D_c: The Data set of client c
η: Fixed learning rate
OUTPUT: The global model parameters β_T
Miner:
initialize β_0
for each round t from 1 to T **do**
  for each client c in C **do**
    β_{t+1}^{c} = ClientUpdate (c, β_t)
  end for
  $S = \{\beta_{t+1}^{1}, \beta_{t+1}^{2}, \ldots, \beta_{t+1}^{c}\}$
  $\beta_T = Mixed\ Aggregation\ Algorithm(S)$
end for
return β_T
ClientUpdate(c, β_t)**:**
$\mathcal{B}$ = split D_c into batches of size B
for each local epoch i from 1 to E **do**
  for batch b in $\mathcal{B}$ **do**
    $\beta = \beta - \eta \nabla \ell(\beta; b)$
  end for
end for
return update β to blockchain

B. Node Selection

Nodes who want to aggregate local models and push them to the blockchain are called *miner*. In a blockchain system, the question of which miner has the right to create a new block is an important issue. Assuming that each miner can create a block at any time and link it to the blockchain, it will inevitably cause confusion (the block information of each node will be inconsistent), so the blockchain must select precisely one miner to create a new block in each round of block production. In pure blockchain systems, the consensus algorithm is mainly used to solve this problem: in PoW nodes with larger computing power are more likely to be selected, and in PoS nodes with larger stakes are more likely to be selected.

The earliest consensus algorithm in blockchain is PoW. However, it has shown an obvious shortcoming in operation till now. In order to have a better chance of being selected, miners will increase their computing power as much as possible. This problem causes a huge waste of computing resources [53], and more seriously, the concentration of computing power will affect the security of the blockchain [54].

In our Bift, the two shortcomings can be avoided. First, Bift only allows real CAVs to join and in the real world, the computing power of each CAV is fixed. Therefore, there will be no

Algorithm 2 Mixed Aggregation Algorithm

INPUT: n: The num of total upload model
m: The num of malicious node
S: The set of upload model, $S = [\beta_i | i = 1, 2...n]$
t: timestamp of last upload model
T_m: The max wait time
OUTPUT: The global model parameters β_T

Krum:
for β_i in S **do**
  for β_j in S **do**
    calculate the *Euclidean Metric* ρ
    $D_{i,j} = \rho(\beta_i, \beta_j)$
  end for
end for
for i in n **do**
  for j in $n - i$ **do**
    $Sum_i = Sum_i + D_{i,j}$
  end for
end for
$\beta_T = \beta$ which has the minimum Sum_i

Trimmed Mean:
for each i in n **do**
  W = sorted array of each weight $w_{i1}, w_{i2}, \ldots, w_{i3}$
  $w_i = \frac{Sum\ of\ Middle\ \ (n-m)\ \ weights}{n-m}$
end for
$\beta_T = \{w_1, w_2, ...w_i\}$

Bulyan:
$SelectionSet = \emptyset$
while $|SelectionSet| < n - 2m$ **do**
  $\beta_i = Krum()$
  $SelectionSet = SelectionSet \cup \beta_i$
end while
$\beta_T = Trimmed\ Mean()$

Main Process:
$flag = t\ mod\ 3$
if $T_{now} - t < T_m$ **then**
  wait
end if
if $flag == 1$ **then**
  return $Krum()$
end if
if $flag == 2$ **then**
  return $Trimmed\ Mean()$
end if
if $flag == 3$ **then**
  return $Bulyan()$
end if

Algorithm 3 PoW in Bift

INPUT: I: Index of Latest block
T: Timestamp
D: The Value of Difficulty
$Nonce$: Nonce of This Block
TXs: Transactions are mined
$Hash_p$: Transactions are mined
OUTPUT: *True* or *Flase*

set $D = 0x0000$
$M = PoFL(TXs)$
$Hash_c = SHA_{256}(I + T + Nonce + M + Hash_p)$
$Re = First\ Four\ Digits\ of\ Hash_c$
while $Re\ < EXCLAMATION >= D$ **do**
  $Nonce = Nonce + 1$
  $Hash_c = SHA_{256}(I + T + Nonce + M + Hash_p)$
end while
$I_c = Get\ Newest\ Block\ Index$
if $I_c = I$ **then**
  return *true*
else
  return *false*
end if

problem of too concentrated computing power in Bift. Second, Bift evaluates node contributions not by computing power but on the quality of their models, so the value of *difficult* is defined as small as possible to reduce the consumption of computing resources of each CAV.

The PoW process in our Bift is shown in Algorithm 3. As mentioned before, we set the value of D to be very small to reduce the waste of CAV's computing resource.

C. Chain Fork

As mentioned before, in each round, miners will aggregate local models via *mixed aggregation algorithm*, and Bift selects exactly one miner to create a new block. However, there is another tricky question: how to verify the authenticity of blocks uploaded by the miner? Our previous algorithm solves the problem of malicious local models, but if the miner is malicious or his upload value is incorrect, it will be devastating.

We use the *chain fork* technique to solve this problem in Bift. When a new block is linked to the blockchain, other miners will immediately verify the authenticity of the content. If this block is incorrect, it will not be recognized by other miners. Other miners who disagree with this block will generate a new block using their own aggregated result and then restart the *node selection* process. If a new block is correct, it will be recognized by other miners and there is a newer block linked behind it. In short, when there are disagreements in the blockchain, the blockchain will fork into several branches, each miner votes for the branch it recognizes by appending new blocks behind it. As a result, the shorter branches are discarded and the longest one is kept. The process of chain fork is shown in Fig. 4.

By applying chain fork, we believe that Bift is safe in the case of no more than 50% of malicious miners.

V. Distributed File Storage and Access

The blockchain is becoming popular in more and more fields because of its decentralized feature. Some people think that the blockchain is essentially a distributed database, but there is a huge flaw in most blockchains: to ensure efficiency and safety, the storage space of the blockchain is costly and inefficient. Dai *et al.* [55] showed that the complete ledger of the Bitcoin

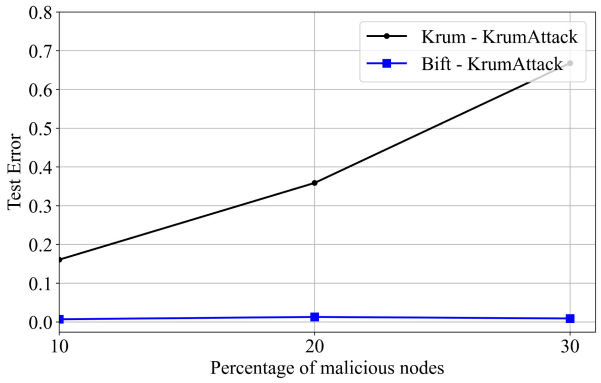

Fig. 3. Bift's performance under different percentage of KrumAttack malicious nodes.

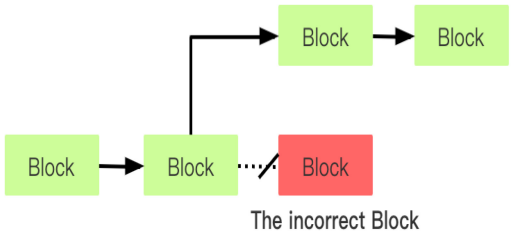

Fig. 4. Example of chain fork.

network constantly increases at about 0.1 GB per day and now it has reached more than 300 GB, it takes a single user almost three days to synchronize Bitcoin Ledger. Ethereum can be used in more fields because of the realization of smart contracts, but its storage consumption is very expensive: users need to pay 68 000 gas for every 1 MB data they store (about U.S. $11).

Most blockchain applications that need to store data use off-chain storage, and the fusion of http or https urls is one method, but data in http urls are controlled by the service provider and may be deleted at any time. As mentioned in Section II, using IPFS is a better way in blockchain applications and it is also for Bift. In traditional Internet of Vehicles (IoVs), vehicles receive and send information through services provided by some companies, and it is expensive for companies to provide bandwidth and storage space and is unreliable for vehicles to store and exchange crucial data. IPFS will generate a hash sequence for each uploaded file (e.g., upload model and high precision map) and then Bift stores the hashes of files instead of actual ones (actually, a block cannot store a large file at all). By the hash of a file, CAVs can locate and get the actual file from an open, flat IPFS net.

We first clarify that data need to be classified stored in Bift as follows.

1) *Stored in IPFS:* For large data, we store them in IPFS. In the FL scenario, big data refer to the local model generated in each iteration of CAV and the global model aggregated by Miner. Furthermore, we reserve related interfaces to provide more possibilities for data sharing between CAVs. For example, CAVs can continuously collect high-precision map information of roads while driving and then upload them to IPFS for sharing. By sharing map information point-to-point, CAVs can obtain the current road conditions in real time. IPFS acts as an offchain-storage middleware role in Bift, reducing the load on the blockchain by storing and forwarding large files
2) *Stored in Blockchain:* For small data, we store them in the blockchain. In Bift, the descriptive information about the model is stored in the *model_info* field of the blockchain, and the transaction record about the token transfer is stored in the *transaction* field. We noticed that after classified storage, the size of a block would not exceed 100 kB, so storing small descriptive fields directly on the blockchain will not cause the node storage space to explode, and it can avoid multiple requests when obtaining key fields.

TABLE IV
TYPICAL CONTROL COMMANDS OF BIFT

Module	Action	Describe
account	create	Create new account
miner	start	Start mining with current account and port
miner	with defend	Decide whether to use PoFL
node	run	Run a node without mining
node	use GPU n	Choose the GPU to train a model
node	is malicious	Decide whether it is a maliciout node
node	submit model	Submit a initial model
tx	transfer	Transfer coin to another address
system	show	Show the Bift status

VI. EXPERIMENTAL RESULTS

To evaluate the proposed Bift, we deployed Bift to train an ML model in 30 docker containers. The base server has 4 CPU cores, 512 GB of RAM and 8 GPUs of GTX 1080. Docker is an open-source application container engine that allows developers to package their applications and dependent packages into a portable image. Containers use the sandbox mechanism completely, and there will be no interfaces between them. We deployed Bift into 30 docker containers to simulate 30 CAV nodes, and each container has its own network, compute resource, memory, and CPU time. Specifically, each Bift container has 900-MB GPU memory and 1-GB RAM.

A. System Control

We developed the control panel for each CAV node or miner. Because Bift's blockchain bottom layer and FL layer are developed based on Python, each node can run the console.py script on the linux console to control all its behaviors and view the system status. Table IV shows some typical control commands of our Bift, and we have too many commands, so it is not convenient to describe them in detail here. In short, we have stipulated a detailed unified interface to facilitate each node to use the system easily.

TABLE V
HYPERPARAMETERS AND DATA SET USED IN THE EXPERIMENTS

Dataset	Model Type	Train/Test Examples	Batch Size	Learning Rate
Traffic lights	CNN	15000/3000	10	0.001

TABLE VI
COMPOSITIONS AND PARAMETERS OF BIFT

Composition System	Value
Nodes	30
Percentage of Malicious Nodes	0.3/0.2/ 0.1
Local Epoch	10
Consensus Algorithm	PoFL
Initial Stake	0 of each node
Update Stake	+20 linear
Data Share System	IPFS

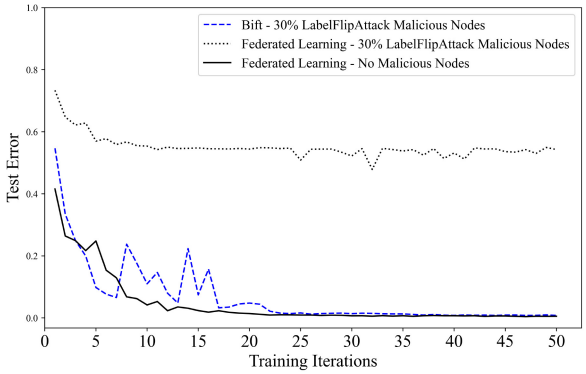

Fig. 5. Bift's performance under 30% LabelFlip malicious nodes.

For the FL process, we used the convolutional neural network (CNN) to recognize Traffic-Light data sets and evaluated the PoFL algorithm. Although the data set and CNN networks we chose in the experiment were very basic, we claim that Bift can handle any tasks about ML due to the general-purpose PyTorch API.

In summary, Table V shows the information of Traffic-Light we used in our experiments and Table VI shows the system config of Bift. The detail of Traffic-Light is shown in Fig. 3.

B. Tolerating Malicious Nodes With PoFL

As mentioned earlier, the pure FL does not have the ability to resist malicious nodes, because it assumes that it is in an "alliance" environment of interagency cooperation. However, in the CAV scenario, we are in a completely distributed and decentralized environment, and it is difficult to directly eliminate malicious nodes from a complex environment.

To evaluate the ability to tolerate malicious nodes of Bift, we first compared the performance of Bift and pure FL to a common LabelFlip attack. Then, we compared Bift with the three single Byzantine-robust aggregation rules by using several unique attacks.

LabelFlip Attack: We first evaluate Bift's performance when we subject the system to a label flipping poisoning attack

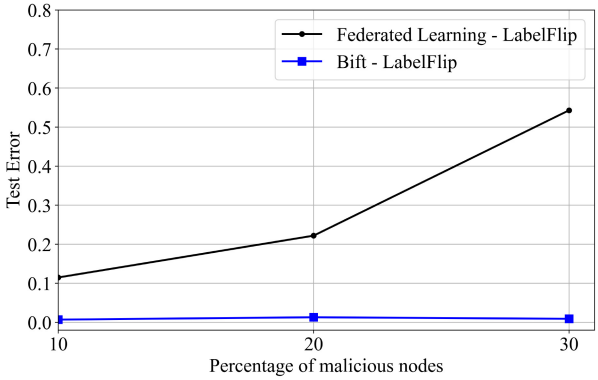

Fig. 6. Bift's performance under different percentage of LabelFlip malicious nodes.

TABLE VII
ACCURACY EVALUATION OF DIFFERENT APPROACHES

Approaches	Bift	Krum	Trimmed Mean	Bulyan	No Defense
LabelFlip Attack	92.3%	89.8%	90.4%	91.2%	54.1%
Krum Attack	92.8%	53.8%	85.3%	83.5%	87.6%
Trimmed Attack	91.3%	76.3%	52.1%	85.2%	73.1%
Bulyan Attack	91.0%	82.5%	88.6%	85.7%	88.2%

(LabelFlip attack) as in Huang *et al.* [56]. Especially, a malicious node will label all green lights as "Red Circle." Fig. 5 shows the test error of Bift for as compared to FL in the case of 30% malicious nodes and no malicious nodes. The result shows that our Bift can achieve similar performance contrast with the baseline pure FL even in the presence of 30% malicious nodes. Especially, after 50 iterations, the accuracy of FL with no malicious node is 92.5%, the accuracy of FL with 30% malicious nodes is 54.1%, and the accuracy of Bift with 30% malicious nodes is 91.8%.

The test error with different percentage of LabelFlip malicious nodes in Bift and FL is shown in Fig. 6, with a different number of malicious nodes, Bift's test error is always lower than FL. As the percentage increases, Bift is stable, but the test error of FL increases.

Local Model Poisoning Attacks: The common Byzantine-robust aggregation rules (Krum, Trimmed Mean, and Bulyan) can indeed defend against some attacks, but they lack the robustness to some carefully constructed attacks. In [17], several local model poisoning attacks are proposed to attack these three aggregation rules.

For all local model poisoning attacks, malicious nodes must exchange attack parameters with each other before uploading the model. Therefore, compared with honest nodes, malicious nodes must have a time delay. Our Bift will abandon the uploaded models that arrive late, so we can theoretically resist model poisoning attacks. Specific experiments have also proved this.

We reproduced the attacks in [17] and evaluate their performance on Bift. Especially, we assume all malicious nodes have full knowledge (the attacker knows the training data set itself and local model on every honest or malicious nodes). Suppose there are c malicious nodes in m CAVs and in an iteration, w_i represents local model parameters the ith

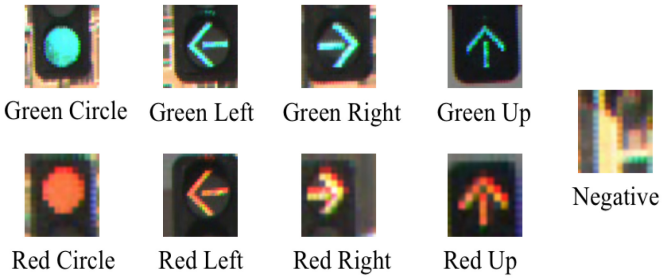

Fig. 7. Detail of traffic-light data set.

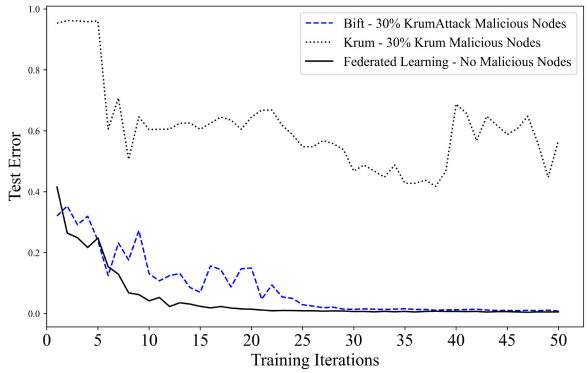

Fig. 8. Bift's performance under 30% KrumAttack malicious nodes.

node intends to upload, for the three aggregation rules, there are three corresponding attack methods, respectively.

1) *Krum Attack:* Recall that Krum selects one local model, which has the closest Euler distance to others, and the attack method is to make Krum select a certain crafted model. Especially, we construct a specific local model w_m by $w_m = w_r - \alpha$, and for all malicious nodes, they upload w_m as their local models. Note that w_r is the model of last iteration and we get the value of α by

$$\alpha \leq \sqrt{\frac{1}{(m-2c-1)d} \cdot \min_{c+1\leq i\leq m} \sum_{l\in\Gamma_{w_i}^{m-c-2}} D^2(\mathbf{w}_l, \mathbf{w}_i)} + \frac{1}{\sqrt{d}} \cdot \max_{c+1\leq i\leq m} D(\mathbf{w}_i, \mathbf{w}_{Re}). \quad (6)$$

2) *TrimmedMean Attack:* Trimmed mean discards the largest and smallest local parameters, and then takes the average of the rest. Suppose w_{ij} is the jth model parameter on the ith node and $w_{\max,j}$ $w_{\min,j}$ are the maximum and minimum of the jth model parameters on the honest nodes. We construct the specific local model of all malicious nodes by making each $w_{i,j}$ in the interval $[w_{\max,j}, 2 * w_{\max,j}]$ (when $w_{\max,j} > 0$) or $[w_{\max,j}, w_{\max,j}/2]$ (when $w_{\max,j} < 0$).
3) *Bulyan Attack:* Because Bulyan is actually a combination of Krum and Trimmed Mean, we use the Krum attack for Bulyan attack.

For the KrumAttack, the result is shown in Figs. 7 and 8. We can see that KrumAttack is able to attack Krum defense methods well. However, our Bift can defend it and make the training effect the same as when there is no attack. By changing the percentage of malicious nodes, Bift's performance is

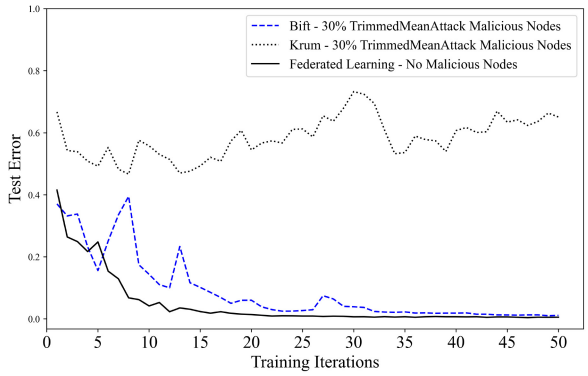

Fig. 9. Bift's performance under 30% TrimAttack malicious nodes.

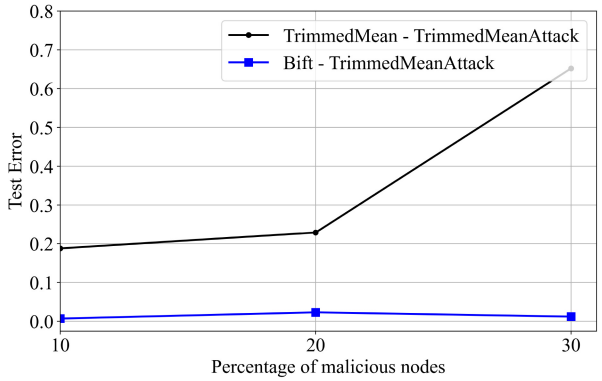

Fig. 10. Bift's performance under different percentage of TrimAttack malicious nodes.

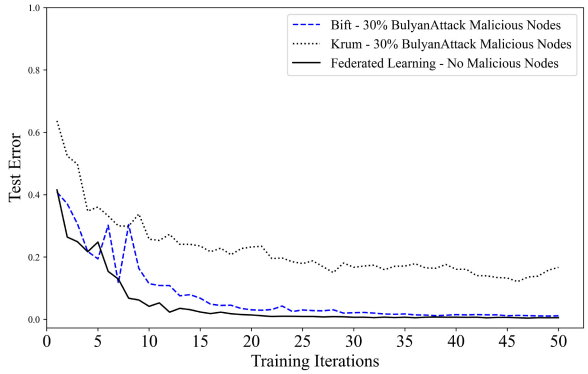

Fig. 11. Bift's performance under 30% BulyanAttack malicious nodes.

stable and Krum's performance is worse as the percentage increases.

Similarly, Figs. 9 and 10 show the the comparison between the TrimmedMean algorithm and Bift under TrimmedMeanAttack. Figs. 11 and 12 show the comparison between Bulyan and Bift. Table VII shows the accuracy evaluation of different approaches. In general, Bift has better robustness than currently known algorithms in a malicious environment, and has practical application value.

C. Performance of Data Sharing

We have integrated IPFS in Bift to increase the speed and robustness of the file-sharing network. IPFS follows

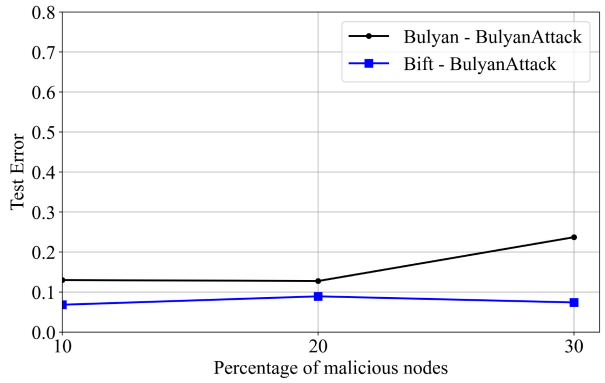

Fig. 12. Bift's performance under different percentage of BulyanAttack malicious nodes.

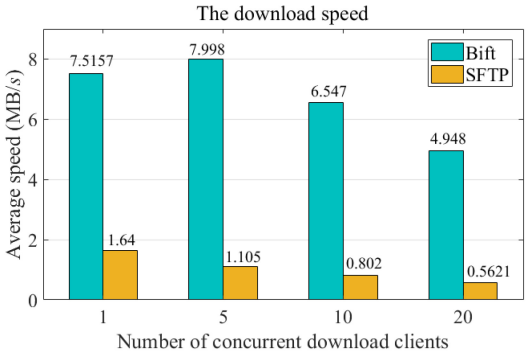

Fig. 13. Average download speed of Bift and SFTP.

content-addressed storage rather than location addressed. For traditional location-addressed storage, when a user wants to access a certain resource (such as URL of HTTP), he first needs to get the location address of this resource on the network. Then, he makes a request to the corresponding server according to the address. However, when the server fails or the resource is moved to another place, the user will not be able to get the resource. The content-addressed storage (IPFS) keeps record of the hash value of the transaction rather than keeping the record of location; hence, IPFS removes duplications across the network because the same file has the same hash. When a user searches for a file to view or download, he is asking the network to find the nodes that are storing the content behind that file's hash.

In Bift, our goal is to make the federate learning fully distributed and decentralized, so we choose IPFS as our data-sharing system rather than the traditional C/S architecture. With IPFS, we can address large amounts of data and put immutable, permanent links in the transactions.

We evaluate the upload and download speed of Bift compared to secure file transfer protocol (SFTP) by creating several threads to simulate download and upload behavior. Note that all downloads/uploads are from/to a single server, and the bandwidth of the test environment is 40M.

The download speed performance is shown in Fig. 13, and Bift's average download speed is much faster than SFTP because Bift has no server concurrent bandwidth limit. As for upload performance shown in Fig. 14, the gap between Bift

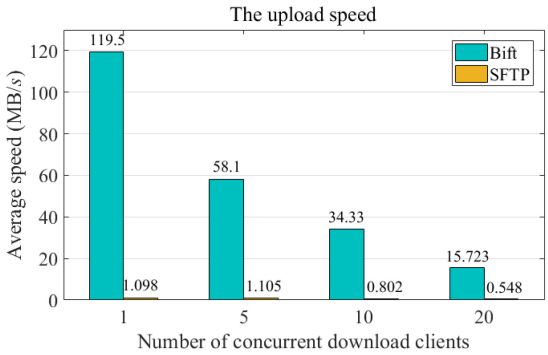

Fig. 14. Average upload speed of Bift and SFTP.

and SFTP is more obvious because Bift upload files do not need to be transmitted over the network. It just archives the files and tells other nodes the harsh value of the files.

Because of its excellent file sharing performance, Bift can enhance the efficiency of federated learning. Furthermore, Bift can provide good support for sharing and transaction data between CAVs.

VII. Conclusion and Future Work

In this article, we proposed a blockchain-based FL system for CAVs named Bift, and gave the detail of the system and working mechanism. A consensus algorithm named PoFL was proposed to defend malicious nodes. Compared with existing works, PoFL performs better in defending against malicious attacks. In addition, we have provided an efficient and stable data-sharing system through IPFS. The experiments show that our data-sharing system performs better than traditional SFTP. Our work can improve the performance of distributed ML of CAVs. Future work is in progress to consider energy efficiency issues in the proposed framework.

References

[1] N. Lu, N. Cheng, N. Zhang, X. Shen, and J. W. Mark, "Connected vehicles: Solutions and challenges," *IEEE Internet Things J.*, vol. 1, no. 4, pp. 289–299, Aug. 2014.

[2] C. Qiu, H. Yao, X. Wang, N. Zhang, F. R. Yu, and D. Niyato, "AI-chain: Blockchain energized edge intelligence for beyond 5G networks," *IEEE Netw.*, vol. 34, no. 6, pp. 62–69, Nov./Dec. 2020.

[3] H. Yao, C. Liu, P. Zhang, S. Wu, C. Jiang, and S. Yu, "Identification of encrypted traffic through attention mechanism based long short term memory," *IEEE Trans. Big Data*, early access, Sep. 20, 2019, doi: 10.1109/TBDATA.2019.2940675.

[4] Y. Liu, F. R. Yu, X. Li, H. Ji, and V. C. M. Leung, "Blockchain and machine learning for communications and networking systems," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 2, pp. 1392–1431, 2nd Quart., 2020.

[5] D. Xu *et al.*, "Learning from naturalistic driving data for human-like autonomous highway driving," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 12, pp. 7341–7351, Dec. 2021.

[6] Y. He *et al.*, "Deep-reinforcement-learning-based optimization for cache-enabled opportunistic interference alignment wireless networks," *IEEE Trans. Veh. Technol.*, vol. 66, no. 11, pp. 10433–10445, Nov. 2017.

[7] T. Okuyama, T. Gonsalves, and J. Upadhay, "Autonomous driving system based on deep *Q* learnig," in *Proc. Int. Conf. Intell. Auton. Syst. (ICoIAS)*, Singapore, 2018, pp. 201–205.

[8] M.-J. Lee and Y.-G. Ha, "Autonomous driving control using end-to-end deep learning," in *Proc. IEEE Int. Conf. Big Data Smart Comput. (BigComp)*, Busan, South Korea, 2020, pp. 470–473.

[9] S. Son, Y. Jeong, and B. Lee, "A driving decision strategy(DDS) based on machine learning for an autonomous vehicle," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, Barcelona, Spain, 2020, pp. 264–266.

[10] X. Wang, X. Ren, C. Qiu, Y. Cao, T. Taleb, and V. C. M. Leung, "Net-in-AI: A computing-power networking framework with adaptability, flexibility, and profitability for ubiquitous AI," *IEEE Netw.*, vol. 35, no. 1, pp. 280–288, Jan./Feb. 2021.

[11] A. Banihani, A. Alzahrani, R. Alharthi, H. Fu, and G. P. Corser, "T-PAAD: Trajectory privacy attack on autonomous driving," in *Proc. IEEE Conf. Commun. Netw. Security (CNS)*, Beijing, China, 2018, pp. 1–2.

[12] S. Karnouskos and F. Kerschbaum, "Privacy and integrity considerations in hyperconnected autonomous vehicles," *Proc. IEEE*, vol. 106, no. 1, pp. 160–170, Jan. 2018.

[13] M. Liu, F. R. Yu, Y. Teng, V. C. M. Leung, and M. Song, "Computation offloading and content caching in wireless blockchain networks with mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 11008–11021, Nov. 2018.

[14] 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*.

[15] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Int. Conf. Artif. Intell. Stat.*, 2017, pp. 1273–1282.

[16] A. Imteaj, U. Thakker, S. Wang, J. Li, and M. H. Amini, "A survey on federated learning for resource-constrained IoT devices," *IEEE Internet Things J.*, early access, Jul. 6, 2021, doi: 10.1109/JIOT.2021.3095077.

[17] M. Fang, X. Cao, J. Jia, and N. Gong, "Local model poisoning attacks to Byzantine-robust federated learning," in *Proc. 29th Security Symp. Security*, 2020, pp. 1605–1622.

[18] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, pp. 1–19, 2019.

[19] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *Proc. Int. Conf. Artif. Intell. Stat.*, 2020, pp. 2938–2948.

[20] Y. He, Y. Wang, F. R. Yu, Q. Lin, J. Li, and V. C. M. Leung, "Efficient resource allocation for multi-beam satellite-terrestrial vehicular networks: A multi-agent actor-critic method with attention mechanism," *IEEE Trans. Intell. Transp. Syst.*, early access, Dec. 7, 2021, doi: 10.1109/TITS.2021.3128209.

[21] R. Kumar and R. Tripathi, "Implementation of distributed file storage and access framework using IPFS and blockchain," in *Proc. 5th Int. Conf. Image Inf. Process. (ICIIP)*, Shimla, India, 2019, pp. 246–251.

[22] N. Marathe, A. Gandhi, and J. M. Shah, "Docker swarm and kubernetes in cloud computing environment," in *Proc. 3rd Int. Conf. Trends Electron. Informat. (ICOEI)*, Tirunelveli, India, 2019, pp. 179–184.

[23] C. Qiu, H. Yao, C. Jiang, S. Guo, and F. Xu, "Cloud computing assisted blockchain-enabled Internet of Things," *IEEE Trans. Cloud Comput.*, early access, Jul. 23, 2019, doi: 10.1109/TCC.2019.2930259.

[24] S. Nakamoto, *Bitcoin: A Peer-to-Peer Electronic Cash System*, Decentralized Bus. Rev., Seoul, South Korea, 2008, Art. no. 21260.

[25] Y. He *et al.*, "An efficient ciphertext-policy attribute-based encryption scheme supporting collaborative decryption with blockchain," *IEEE Internet Things J.*, early access, Jul. 26, 2021, doi: 10.1109/JIOT.2021.3099171.

[26] A. Azaria, A. Ekblaw, T. Vieira, and A. Lippman, "MedRec: Using blockchain for medical data access and permission management," in *Proc. 2nd Int. Conf. Open Big Data (OBD)*, Vienna, Austria, 2016, pp. 25–30.

[27] Y. He, Y. Wang, C. Qiu, Q. Lin, J. Li, and Z. Ming, "Blockchain-based edge computing resource allocation in IoT: A deep reinforcement learning approach," *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2226–2237, Feb. 2021.

[28] C. Qiu, X. Wang, H. Yao, J. Du, F. R. Yu, and S. Guo, "Networking integrated cloud–edge–end in IoT: A blockchain-assisted collective *Q*-learning approach," *IEEE Internet Things J.*, vol. 8, no. 16, pp. 12694–12704, Aug. 2021.

[29] R. Singh, A. D. Dwivedi, and G. Srivastava, "Internet of Things based blockchain for temperature monitoring and counterfeit pharmaceutical prevention," *Sensors*, vol. 20, no. 14, p. 3951, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/14/3951

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

[31] 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.

[32] X. Bao, C. Su, Y. Xiong, W. Huang, and Y. Hu, "FLChain: A blockchain for auditable federated learning with trust and incentive," in *Proc. 5th Int. Conf. Big Data Comput. Commun. (BIGCOM)*, QingDao, China, 2019, pp. 151–159.

[33] Y. Liu, Z. Ai, S. Sun, S. Zhang, Z. Liu, and H. Yu, "Fedcoin: A peer-to-peer payment system for federated learning," in *Federated Learning*, vol. 12500. Cham, Switzerland: Springer, 2020, pp. 125–138.

[34] 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.

[35] J. Chen, X. Pan, R. Monga, S. Bengio, and R. Jozefowicz, "Revisiting distributed synchronous SGD," 2016, *arXiv:1604.00981*.

[36] M. Abadi *et al.*, "TensorFlow: A system for large-scale machine learning," in *Proc. 12th Symp. Oper. Syst. Design Implement. (SOSD)*, 2016, pp. 265–283.

[37] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," 2012, *arXiv:1206.6389*.

[38] M. Jagielski, A. Oprea, B. Biggio, C. Liu, C. Nita-Rotaru, and B. Li, "Manipulating machine learning: Poisoning attacks and countermeasures for regression learning," in *Proc. IEEE Symp. Security Privacy (SP)*, San Francisco, CA, USA, 2018, pp. 19–35.

[39] B. Li, Y. Wang, A. Singh, and Y. Vorobeychik, "Data poisoning attacks on factorization-based collaborative filtering," in *Advances in Neural Information Processing Systems*, vol. 29. Red Hook, NY, USA: Curran, 2016, pp. 1885–1893.

[40] B. I. P. Rubinstein *et al.*, "ANTIDOTE: Understanding and defending against poisoning of anomaly detectors," in *Proc. 9th ACM SIGCOMM Conf. Internet Meas.*, 2009, pp. 1–14.

[41] O. Suciu, R. Mărginean, Y. Kaya, H. Daumé III, and T. Dumitraş, "When does machine learning FAIL? generalized transferability for evasion and poisoning attacks," in *Proc. 27th Security Symp. Security*, 2018, pp. 1299–1316.

[42] H. Xiao, B. Biggio, G. Brown, G. Fumera, C. Eckert, and F. Roli, "Is feature selection secure against training data poisoning?" in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1689–1698.

[43] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 5650–5659.

[44] E. M. E. Mhamdi, R. Guerraoui, and S. Rouault, "The hidden vulnerability of distributed learning in Byzantium," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 3521–3530.

[45] E. M. E. Mhamdi, R. Guerraoui, and S. Rouault, "The hidden vulnerability of distributed learning in Byzantium," 2018, *arXiv:1802.07927*.

[46] J. Benet, "IPFS-content addressed, versioned, P2P file system," 2014, *arXiv:1407.3561*.

[47] Y. Zhao *et al.*, "Privacy-preserving blockchain-based federated learning for IoT devices," *IEEE Internet Things J.*, vol. 8, no. 3, pp. 1817–1829, Feb. 2021.

[48] R. Kumar and R. Tripathi, "A secure and distributed framework for sharing COVID-19 patient reports using consortium blockchain and IPFS," in *Proc. 6th Int. Conf. Parallel Distrib. Grid Comput. (PDGC)*, 2020, pp. 231–236.

[49] B. Lucas and R. V. Páez, "Consensus algorithm for a private blockchain," in *Proc. IEEE 9th Int. Conf. Electron. Inf. Emerg. Commun. (ICEIEC)*, 2019, pp. 264–271.

[50] J. Weng, J. Weng, J. Zhang, M. Li, Y. Zhang, and W. Luo, "DeepChain: Auditable and privacy-preserving deep learning with blockchain-based incentive," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 5, pp. 2438–2455, Sep./Oct. 2021.

[51] "OpenMined." OL. [Online]. Available: http://www.openmined.org/ (Accessed: Oct. 20, 2021).

[52] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. 32nd Int. Conf. Mach. Learn.*, 2015, pp. 448–456. [Online]. Available: JMLR.org

[53] K. J. O'Dwyer and D. Malone, "Bitcoin mining and its energy footprint," in *Proc. 25th IET Irish Signals Syst. Conf. China Ireland Int. Conf. Inf. Commun. Technol. (ISSC/CIICT)*, 2014, pp. 280–285.

[54] C. Ye, G. Li, H. Cai, Y. Gu, and A. Fukuda, "Analysis of security in blockchain: Case study in 5%-attack detecting," in *Proc. 5th Int. Conf. Depend. Syst. Appl. (DSA)*, Dalian, China, 2018, pp. 15–24.

[55] M. Dai, S. Zhang, H. Wang, and S. Jin, "A low storage room requirement framework for distributed ledger in blockchain," *IEEE Access*, vol. 6, pp. 22970–22975, 2018.

[56] L. Huang, A. D. Joseph, B. Nelson, B. I. P. Rubinstein, and J. D. Tygar, "Adversarial machine learning," in *Proc. 4th ACM Workshop Security Artif. Intell.*, 2011, pp. 43–58.