BASS: Blockchain-Based Asynchronous SignSGD for Robust Collaborative Data Mining

Chenhao Xu*, Youyang Qu†¶, Yong Xiang*, Longxiang Gao‡, David Smith†, and Shui Yu§
*Deakin Blockchain Innovation Lab, School of Information Technology, Deakin University, Geelong, Australia.
†Data 61 Australia Commonwealth Scientific and Industrial Research Organization, Australia.
‡Qilu University of Technology and Shandong Computer Science Center, China.
§School of Computer Science, University of Technology Sydney, Australia.
Email: *{xueri, yong.xiang}@deakin.edu.au †{youyang.qu, david.smith}@data61.csiro.au
‡gaolx@sdas.org §shui.yu@uts.edu.au

2022 IEEE 9th International Conference on Data Science and Advanced Analytics (DSAA) | 978-1-6654-7330-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/DSAA54385.2022.10032341

***Abstract*—Federated learning (FL) is a machine learning framework for collaborative data mining in many scenarios (e.g. Internet of Things) due to its privacy-preserving feature. However, various attacks arise security concerns of FL, such as poisoning, backdoor, and DDoS attacks. Several blockchain-based FL schemes strengthen credibility and security without considering the increased communication overhead. Some existing work compresses local updated gradients to sign vectors to lower communication overhead at the expense of model accuracy. To address the above concerns, this paper offers a blockchain-based asynchronous SignSGD (BASS) scheme. A novel asynchronous sign aggregation algorithm is introduced to ensure model accuracy even if the local updated gradients are compressed to sign vectors. Considering the unstable network connection on IoT, a consensus algorithm that elects multiple leader nodes enables reliable global model aggregation. The introduced blockchain improves credibility and security without downgrading efficiency. Empirical studies show that BASS outperforms other schemes in efficiency, model accuracy, and security.**

***Index Terms*—Federated Learning, SignSGD, Blockchain, IoT**

I. Introduction

Federated learning (FL) is a privacy-preserving distributed machine learning (ML) framework that allows participants to mine data collaboratively by training a global model without revealing local data [1]. As participants do not need to send raw data to a server and wait for trained models, FL is more efficient than classic ML schemes and is suitable for real-time scenarios. Therefore, some existing work integrates FL with the Internet of Things (IoT) to enable efficient edge intelligence while maintaining privacy [2], [3]. However, some security issues, such as poisoning and backdoor attacks, hinder FL from being applied to IoT [4], [5].

To mitigate the security concerns, some existing work integrates the blockchain with FL [6], [7]. The blockchain is a tamper-resistant distributed storage system with a fault-tolerant consensus algorithm and a self-executing smart contract [2]. Therefore, it is able to build trust among nodes by managing the model training process and tracing down malicious local models and nodes in FL. However, blockchain brings a lot of communication burdens to FL, causing efficiency degradation, especially on bandwidth-limited IoT [8].

SignSGD is a gradient compression approach that reduces communication overhead by only uploading the signs of local updated gradients (sign vectors) to the aggregation server [9], [10]. However, the lost mantissa of gradients reduces the model accuracy to some extent. Besides, since IoT devices have different computing and communication resources (device heterogeneity), varied training time forces nodes to wait for each other before global model aggregation. Asynchronous federated learning (AFL) is a scheme that reduces the time nodes wait for the others by performing aggregation immediately when a new local model comes [8], [11]. However, AFL has lower model accuracy than classic FL, as there are stale local models trained from outdated global models [12].

To address the above issues, this paper offers a blockchain-based asynchronous SignSGD (BASS) scheme. Specifically, the blockchain is utilized as a distributed cache to store sign vectors. The smart contract assigns reasonable weights to sign vectors during aggregation. The consensus algorithm elects multiple leaders for reliable global model aggregation. Experiments conducted on heterogeneous devices demonstrate that BASS has exceptional model accuracy, communication efficiency, and security. The main contributions are as follows.

- A reliable blockchain-based asynchronous SignSGD scheme with an efficient consensus algorithm is designed for collaborative data mining on IoT.
- An asynchronous sign aggregation algorithm developed on the smart contract ensures model accuracy and security while preserving efficiency.
- Experiments conducted on heterogeneous devices verify the efficiency, model accuracy, and security of the proposed scheme on various models and datasets.

II. Related Work

This section demonstrates related work from three aspects: blockchain-based federated learning, SignSGD, and asynchronous federated learning.

¶ is the corresponding author.

A. Blockchain-based Federated Learning

The blockchain is a distributed storage system with a consensus algorithm and smart contracts ensuring that each transaction is validated and replicated across nodes. As a result, numerous FL schemes adopt the blockchain to resist attacks or tolerant faults [4], [6], [13]–[16]. Specifically, the methods of utilizing the blockchain include: (1) Encourage nodes to engage in the training and establish a positive reputation to earn trust [4], [13], [14]; (2) Cross-validate models, signatures, and global model aggregation to ensure proper training [4], [6], [15]; (3) Decentralize the global model aggregation process to avoid message floods and improve reliability [6], [16]. However, these schemes hardly consider additional communication overhead, which lowers training efficiency and downgrades the utility of FL on IoT devices.

B. SignSGD

To reduce the communication overhead in FL, SignSGD, which extracts and aggregates the signs of the updated stochastic gradient descent is proposed [9], [10]. Some existing work utilizes SignSGD to protect the privacy of FL [17]. However, as the mantissa of gradients is removed before being aggregated, the global model in SignSGD is less accurate than in SGD. As a result, several methods for improving the model accuracy of SignSGD are proposed: (1) Calculate accumulated residual errors to compensate for the disparity between local updated gradients and global model updates [18]; (2) When uploading sign vectors, add an extra scaling factor [19]. However, none of these schemes takes security into charge, as these additional factors increase the possibility of tampering by malicious nodes. By contrast, the proposed asynchronous sign aggregation algorithm improves global model accuracy without downgrading security.

C. Asynchronous Federated Learning

As the computing power of IoT devices is usually heterogeneous, asynchronous FL schemes are proposed to mitigate the time spent waiting for slow nodes [8], [20]. However, as stale local models from slow nodes are aggregated less frequently, several approaches are presented to increase the utilization of stale local models. In [12], the authors assume that a larger training dataset causes stale local models, thus they assign stale local models a higher weight during aggregation. In [21], the authors assign a static staleness coefficient to stale local models without conducting further analysis. In [22], a semi-asynchronous FL scheme with a centralized cache on the server is presented for better exploitation of the stale local models. However, none of these schemes consider reducing communication overhead, despite the fact that asynchronous aggregation would result in a more frequent model transfer. Besides, these schemes do not look into security issues.

III. System Model

This section illustrates system model from several aspects, including workflow, asynchronous sign aggregation, consensus algorithm, and model discussion.

A. Workflow

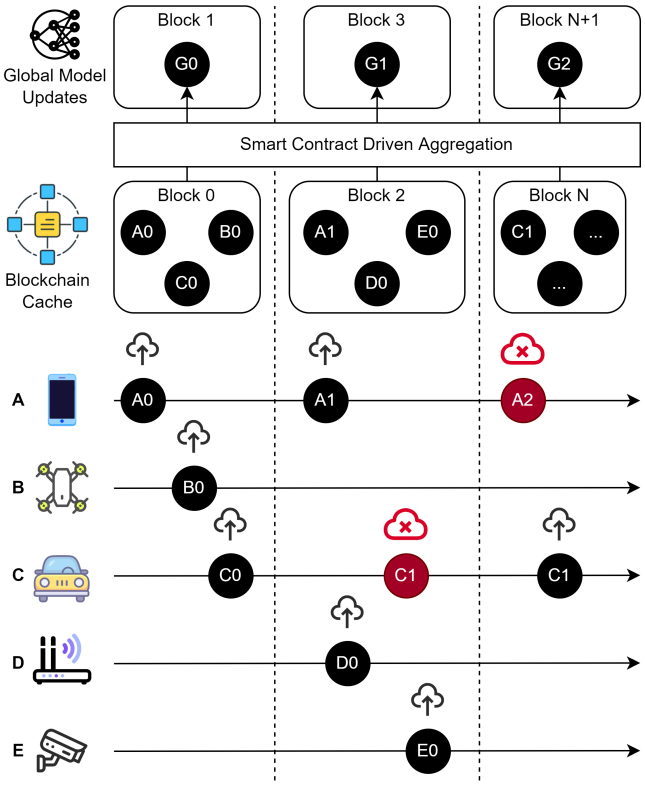


Fig. 1. The workflow of BASS. Cache level is 60% (3/5 Nodes). The global model updates are also included in blocks.

In BASS, IoT devices (i.e. nodes) have the same ML target in the network, such as predicting traffic conditions. As shown in Fig. 1, smart phones, unmanned aerial vehicles, smart vehicles, roadside units, and cameras continually train local models based on their local data. After training, nodes upload their local models to the blockchain. Each node has a blockchain peer deployed to communicate with other nodes and maintain a shared ledger. The smart contract running on all nodes manages the model training process. The shared ledger on nodes ensures the models are non-tamperable during the training process. The blocks in the blockchain are utilized as a cache for local models, where the cache level is calculated by dividing the number of cached local models by the total number of nodes. For example, the cache level is 60% means three local models are cached in a block for asynchronous aggregation when there are five nodes in the network. When enough local models are cached, the smart contract performs aggregation immediately and generates an updated global model, which is then stored in the next block of the blockchain.

The workflow of BASS is demonstrated in Fig. 1. Before training, an initial global model is pre-defined in BASS and stored in the blockchain. The smart contract on each node triggers the training of local models. Assume there are five nodes in the network (A to E), and nodes A, B, and C are the first to complete the training of the three local models A0, B0,

and C0. Then, A0, B0, and C0 are packed in Block 0 (called the local block) for the aggregation of the global model G0. A new block Block 1 (called the global block) is created for storing the latest global model. After that, nodes A, D, and C finish their local model training and upload A1, D0, and C1 to the blockchain. However, the upload of the local model C1 is unsuccessful due to the poor network connection of the vehicle. Without waiting for C1 to upload successfully, after the local model E0 is uploaded, A1, D0, and E0 are packed in Block 2 for the aggregation of the global model G1. A global block Block 3 is created for storing the global model G1. Then, the upload of the local model A2 is failed and it is left out of the following block. However, the outdated local model C1 is uploaded successfully and will be included in the next block for aggregation.

B. Asynchronous Sign Aggregation

Algorithm 1 Asynchronous Sign Aggregation Algorithm

1: **function** LOCALUPDATE(k) ▷ *On all nodes*
2: initialize $\mathbf{x}_G^0$ and $\mathbf{m}_k^0$
3: **for** $t = 1, 2, \ldots, E$ **do**
4: fetch $\mathbf{x}_G^t$ from the blockchain
5: $\Delta\mathbf{x}_G^t \leftarrow \mathbf{x}_G^t - \mathbf{x}_G^{t-1}$ *// global model update*
6: $\mathbf{g}_k^t \leftarrow \text{stochasticGradient}(\mathbf{x}_G^t)$ *// local training*
7: $\mathbf{m}_k^t \leftarrow \beta\mathbf{m}_k^{t-1} + (1-\beta)\mathbf{g}_k^t$ *// momentum*
8: $\mathbf{s}_k^t \leftarrow \text{sign}(\mathbf{m}_k^t)$ *// sign vector extraction*
9: $\mathbf{p}_k^t \leftarrow \text{bitpack}(\mathbf{s}_k^t)$ *// bit pack of sign vectors*
10: upload $\mathbf{p}_k^t$ to the blockchain
11: **end for**
12: **end function**
13:
14: **function** GLOBALUPDATE ▷ *On leader nodes*
15: collect enough number of $\{\mathbf{p}_k^t \mid k \in K\}$
16: $\{\mathbf{s}_k^t \mid k \in K\} \leftarrow \text{unbitpack}(\{\mathbf{p}_k^t \mid k \in K\})$
17: $\mathbf{x}_G^t \leftarrow \sum_{k\in K} \mathbf{s}_k^t \;/\; \text{count}(\{\mathbf{s}_k^t \mid k \in K\})$
18: upload $\mathbf{x}_G^t$ to the blockchain
19: **end function**

In order to improve communication efficiency, an asynchronous sign aggregation algorithm is proposed. The algorithm is composed of two parts, including local update and global update, both of which are governed by the smart contract and execute on nodes in a distributed manner. Specifically, the local update is performed on all nodes and is mostly concerned with the local model training, whereas the global update is performed on leader nodes and is primarily concerned with the global model aggregation.

As shwon in Algorithm 1, when executing the local update function, node k first initializes a global model $\mathbf{x}_G^0$ and momentum $\mathbf{m}_k^0$ for the future training process. The values of the global model $\mathbf{x}_G^0$ are pre-defined by the smart contract and globally unified across nodes. Then, node k starts E epochs of local training, as shown in lines 3 to 11. By fetching the latest global model $\mathbf{x}_G^t$ from the blockchain and comparing it with the previous global model $\mathbf{x}_G^{t-1}$, node k computes the update of the global model and gets $\Delta\mathbf{x}_G^t$, as shown in lines 4 and 5. Then, node k performs its local training based on its local dataset and obtains updated gradients $\mathbf{g}_k^t$ in line 6. Momentum is introduced to stabilize the update of the global model, with β controlling the weight of the updated gradients $\mathbf{g}_k^t$, as shown in line 7. In order to improve communication efficiency, node k extracts the signs of the momentum and obtains the sign vectors $\mathbf{s}_k^t$. The sign vectors are then packed to bits $\mathbf{p}_k^t$ and uploaded to the blockchain, as shown in lines 8 to 10.

When executing the global update function, nodes first collect enough number of $\{\mathbf{p}_k^t\}$ for creating a new block. In other words, blocks are used as caches for $\{\mathbf{p}_k^t\}$, as shown in line 15. The bit packs are then unpacked to sign vectors, as shown in line 16. Then, to calculate the new global model, the sum of sign vectors is divided by the number of sign vectors, as shown in line 17. That is, the direction of the global model update is determined by the majority of the nodes, preventing a few malicious nodes from acting maliciously.

C. Consensus Algorithm

As the process of asynchronous sign aggregation algorithm is dispersed across nodes, it is critical to ensure that the values calculated on each node are the same.

The period between the generation of two new global models is called a term. For each term, a certain number of nodes are elected as leaders, and they are in charge of the global update function in Algorithm 1. The election of leader nodes is based on verifiable random functions [23]. When electing leader nodes, each node puts its private key and the hash value of the latest block into the verifiable random function to generate a verifiable hash and a parameter π. To determine which node will become the leader, a specific range $\mathcal{C}$ is introduced, where $\mathcal{C} = [0, 0.5)$. Assuming the verifiable hash is h, the owner of h becomes a leader only if $h/2^{|h|} \in \mathcal{C}$. In other words, half of the nodes will become leader nodes in each term. This ensures that the global update function is run by a sufficient number of nodes, even if some nodes go offline unexpectedly.

There are three advantages of using verifiable random functions. First, the election of leader nodes is guaranteed to be random, as the hash value of the latest block is unpredictable. The latest block before the election is usually the one that stores global model updates, which are derived by averaging all cached local models and are not determined by any particular node. As a result, the leader nodes that take charge of global model aggregation are random for each term, preventing malicious nodes from doing evil. Second, there is no requirement for a trustworthy third party for leader election and validation. The processes are fully decentralized and efficient, allowing nodes to re-elect leaders for each term without putting a huge strain on the network. Third, it is easy to scale the network without affecting the participating nodes. Whenever a new node comes, it only needs to provide other nodes with its public key to take part in the election of leaders in the next term.

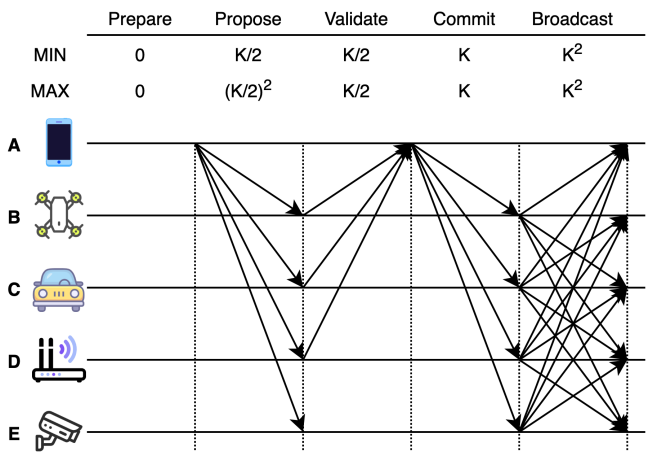


Fig. 2. The consensus process of the global update function.

The global update function requires the leader nodes to reach a consensus on the new global model. To ensure security and efficiency, the consensus process is composed of five parts, as indicated in Fig. 2. Assuming node A, B, C, and D are leader nodes in the term, node A is the first one that finishes its calculation and proposer of the new global model. Besides, the number of nodes in the network is K and the number of leader nodes is expected to be $K/2$. The details of each step in the consensus process are as follows.

- *Prepare*: Leader nodes collect bit-packed sign vectors and produce a new global model according to the global update function in Algorithm 1. There is no communication among leader nodes.
- *Propose*: One of the leader nodes proposes a new block with a new global model included. The proposal is typically broadcasted to all nodes in the network (the number of messages is $K/2$), as a leader does not have a list of other leader nodes in a term. However, in certain circumstances, leader nodes propose new blocks at the same time, resulting in at most $(K/2)^2$ messages in the network.
- *Validate*: Leader nodes validate the new block by running the global update function locally. If a leader node does not have sign vectors in its local cache that are used to create the new global model, the validation fails. Furthermore, if the global model calculated by a leader node is not consistent with the one in the proposal, the validation fails, too. These verification methods ensure that the new global model is not fabricated by malicious leader nodes. A leader node only votes for one correct global model for each term. Therefore, the message number in the network is $K/2$.
- *Commit*: After receiving enough signatures for the new block from leader nodes, the proposer broadcast its new block to all nodes in the network. Nodes can only accept new blocks signed by more than $1/3$ of the leader nodes. In other words, $1/6$ of the leader nodes are allowed to be offline or malicious for each term. The number of messages in this step is K.
- *Broadcast*: Finally, all nodes assist in broadcasting the new valid block to other nodes in the event that some of them have poor network connections and miss the block. The number of messages in this step is K^2. This step overlaps with the preparing step for the next term, which means that this step will not slow the process of preparing the global model for the following term on leader nodes.

TABLE I
THE PARAMETER SETTINGS

Parameter	Value
The number of nodes K	10
The number of epochs E	200
The time of training (s)	300
The learning rate η	0.01, 0.001, 0.0001
The local data size	1500

IV. SYSTEM EVALUATION

This section evaluates the proposed scheme based on three criteria: efficiency, model accuracy, and security.

A. Experiment Setup

The parameter settings utilized in experiments are illustrated in Table I. The network includes ten nodes, each with 8 CPU cores and 8GB of memory. The software environment includes Python 3.8 and Pytorch v1.8.1 which are utilized for the model training. Hyperledger Fabric v2.3.0 is used as the basement of the blockchain, upon which a consensus algorithm is developed. By default, the training continues for 200 epochs or 300 seconds. To improve the model accuracy while maintaining its stability, the learning rate for followers and leaders is identical but dynamically changed. Specifically, the learning rate is 0.01 for the first 100 training rounds for a faster convergence speed. The learning rate is then modified to 0.001 for rounds 101 to 150 and 0.0001 for rounds 151 to 200 to keep the stability of the model accuracy. Besides, the number of training and test samples on each node is 1500 to simulate the working environment of IoT devices in real-world scenarios.

The CIFAR-10 and FMNIST datasets are well-known benchmark datasets utilized in the experiments to evaluate convergence speed and model accuracy [8]. To validate the effectiveness of BASS, both the CNN and MLP models are trained and tested over the CIFAR-10 and FMNIST datasets.

In order to evaluate the performance of BASS, several state-of-the-art schemes are introduced for comparison.

- **BSSS**: The synchronous version of BASS. The global model is constructed after all local models from all nodes have been collected.
- **EFSIGN**: The SignSGD scheme with accumulated residual error feedback [18]. There is no blockchain incorporated.

- **ASOFED**: An AFL scheme with static decay coefficient balancing the previous and current gradients [12]. There is no blockchain incorporated.
- **BDFL**: An AFL scheme with all models stored in the blockchain [16].
- **FEDAVG**: A classic FL scheme without compression of models, asynchronous aggregation strategy, and the blockchain [1].
- **LOCAL**: A local training scheme where each node trains its model locally. The overall model accuracy is evaluated by averaging the accuracy on each node.

To evaluate efficiency of different schemes, the sizes of models to be transmitted when training on different datasets are compared. This demonstrates how compact the model is and how much the transmission burden is decreased in BASS. Besides, the time cost of different steps in a training round is compared when training the CNN model on the CIFAR-10 dataset. This reveals how compressed local models and asynchronous aggregation in BASS save time in a training round.

To evaluate the model accuracy and convergence speed of different schemes, the accuracies of models trained on diverse datasets are compared. Specifically, four VM nodes are substituted by Raspberry PI devices in the network to simulate the diverse computing power of nodes in real-world scenarios. The model accuracy is sampled every three seconds from all nodes and averaged.

To evaluate the security of different schemes, poisoning attacks, backdoor attacks, and DDoS attacks are introduced during the training process. Specifically, attacks are launched by the first three nodes in the network and last until the finish of the training. When launching poisoning attacks, the malicious nodes upload their randomly generated local models to the blockchain. When launching backdoor attacks, the malicious nodes add a specific mark to the original data and label it to a different class during the training of the local model. When launching DDoS attacks, the malicious nodes continually send invalid messages instead of local models to the blockchain.

B. Efficiency

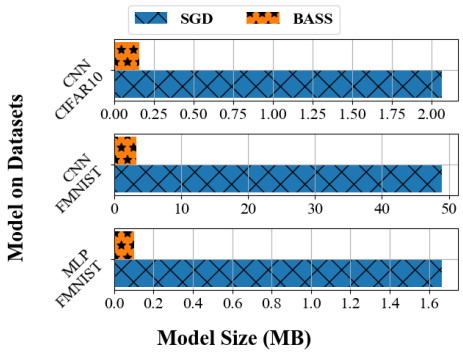


Fig. 3. Model size comparison.

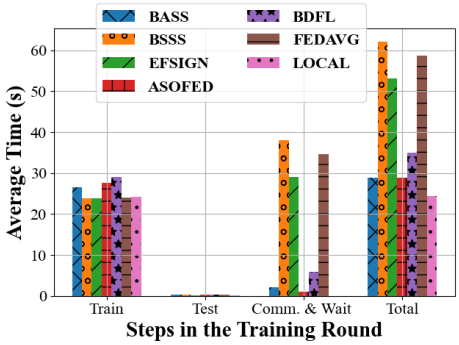


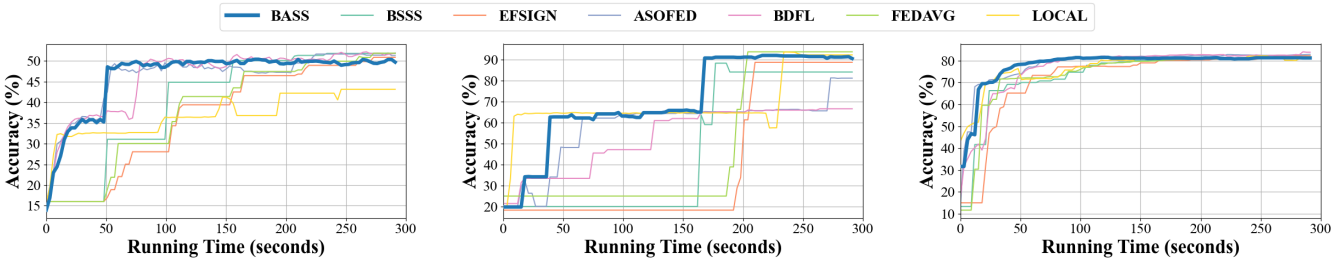
Fig. 4. Time cost comparison.

As shown in Fig. 3, the model size of BASS is around 93% less than that of SGD-based schemes regardless of the dataset. This is because only gradient signs are transmitted instead of entire float numbers. For even less space use, BASS compresses the sign vectors into bits before transmission. The space savings are especially noticeable (roughly 45 MB in size) when training the CNN model on the MNIST dataset, as the original model is 49 MB and much larger than the others. This reveals that BASS has a more obvious advantage over other SGD-based schemes in terms of efficiency when it comes to training more complicated models.

As shown in Fig. 4, BASS has a little higher time cost in the training step (around 3 seconds). This is because of the additional computing burden from the blockchain consensus, the more frequent model aggregation, the sign vector extraction, and the bit-packing compression. However, when it comes to the communication step, the time cost of BASS is far less than that of BSSS, EFSIGN, and FEDAVG, which is around 27 to 36 seconds less. In other words, the small time cost increase in the training step is compensated by the time cost saved in the communication step. This is also evidenced by the total time required to complete a round of training, with BASS taking around 20 to 30 seconds shorter than BSSS, EFSIGN, and FEDAVG. Another asynchronous scheme, ASOFED, is as efficient as BASS, but it excludes the blockchain, implying that BASS minimizes the impact of the blockchain in terms of efficiency. This is also evidenced by another blockchain-based AFL scheme, BDFL, whose time cost is 7 seconds more than BASS and ASOFED. Because there is no requirement for communication, LOCAL is the only scheme that takes less time than BASS, but it comes at a cost in terms of model accuracy and security.

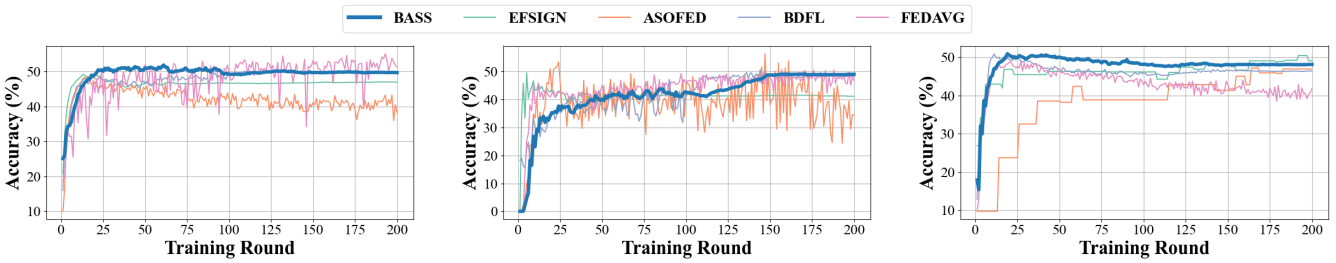
C. Model Accuracy

As shown in Fig. 5, the model accuracy of BASS is always higher than that of BSSS, EFSIGN, ASOFED, BDFL, and FEDAVG. The reasons come from three aspects: First, compared with blockchain-based FL schemes, BASS is more efficient in communication due to its asynchronous aggregation strategy and sign-based bit packs of gradients. Second, compared with SignSGD-based FL schemes, BASS converges faster due to



(a) CNN on CIFAR-10 (b) CNN on FMNIST (c) MLP on FMNIST

Fig. 5. The model accuracy is compared when training CNN and MLP models on different datasets. Four of the ten nodes are IoT devices to simulate the varying computing power in real-world scenarios and test the efficiency of BASS.



(a) Poisoning Attack (b) Backdoor Attack (c) DDoS Attack

Fig. 6. The model accuracy is compared when training CNN on CIFAR-10 under various attacks.

cached sign vectors and the asynchronous sign aggregation algorithm. Third, compared with asynchronous FL schemes, BASS compresses local models while preserving their utility. As there is no communication requirement in the LOCAL scheme and the CNN model has enough expression ability to fit the FMNIST dataset, the model accuracy of it rises faster than that of BASS at first, as shown in Fig. 5 (b). However, when dealing with more complicated datasets like CIFAR-10, the lack of communication between nodes during the training process causes the model accuracy of LOCAL to be worse than that of BASS after convergence, as shown in Fig. 5 (a). In summary, BASS strikes a compromise between convergence speed and model accuracy while improving communication efficiency as much as possible.

D. Security

Malicious nodes upload meaningless local models when launching poisoning attacks to decrease the accuracy of the global model. As shown in Fig. 6 (a), when facing poisoning attacks, BASS converges faster than FEDAVG and has a comparable but more stable model accuracy than FEDAVG. This is due to the asynchronous sign aggregation algorithm abandoning the mantissa of the local updated gradients, preventing malicious local models from causing too much damage to the global model. Compared with the conventional SignSGD-based FL scheme EFSIGN, BASS improves model accuracy due to its proportional aggregation on sign vectors. Backdoor attacks are launched by tagging data with a special mark and misclassifying it to another class. As shown in Fig. 6 (b), BASS outperforms other schemes in terms of model accuracy and stability after converging. When facing DDoS attacks, BASS is able to distribute the strain of local model caching and global model aggregation to numerous leader nodes in the network. Therefore, the convergence and the accuracy of the global model in BASS are hardly affected, as shown in Fig. 6 (c).

V. Conclusion

To reduce the communication overhead, as well as improve the model accuracy and reliability of FL, a blockchain-based asynchronous SignSGD (BASS) scheme is proposed in this paper. In order to improve model accuracy, an asynchronous sign aggregation that proportionally sums up sign vectors from local nodes is designed. Besides, a novel consensus algorithm is developed based on verifiable random functions to enable reliable global model aggregation while keeping efficient. Empirical studies validate the efficiency, model accuracy, and security of BASS. Future research includes investigating the effects of cache level, leader proportion, and the weight assigned to sign vectors during aggregation.

Acknowledgment

Chenhao Xu wants to express his thank to Chuying Wen for her invaluable love: “Will you marry me?”

References

[1] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[2] Yunlong Lu, Xiaohong Huang, Ke Zhang, Sabita Maharjan, and Yan Zhang. Blockchain empowered asynchronous federated learning for secure data sharing in internet of vehicles. *IEEE Transactions on Vehicular Technology*, 69(4):4298–4311, 2020.

[3] Xiangjie Kong, Haoran Gao, Guojiang Shen, Gaohui Duan, and Sajal K Das. Fedvcp: A federated-learning-based cooperative positioning scheme for social internet of vehicles. *IEEE Transactions on Computational Social Systems*, 2021.

[4] Youyang Qu, Shiva Raj Pokhrel, Sahil Garg, Longxiang Gao, and Yong Xiang. A blockchained federated learning framework for cognitive computing in industry 4.0 networks. *IEEE Transactions on Industrial Informatics*, 2020.

[5] Lei Cui, Youyang Qu, Gang Xie, Deze Zeng, Ruidong Li, Shigen Shen, and Shui Yu. Security and privacy-enhanced federated learning for anomaly detection in iot infrastructures. *IEEE Transactions on Industrial Informatics*, 2021.

[6] Chenhao Xu, Youyang Qu, Peter W Eklund, Yong Xiang, and Longxiang Gao. Bafl: An efficient blockchain-based asynchronous federated learning framework. In *2021 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–6. IEEE, 2021.

[7] Chenhao Xu, Yong Li, Yao Deng, Jiaqi Ge, Longxiang Gao, Mengshi Zhang, Yong Xiang, and Xi Zheng. Scei: A smart-contract driven edge intelligence framework for iot systems. *arXiv preprint arXiv:2103.07050*, 2021.

[8] Chenhao Xu, Youyang Qu, Yong Xiang, and Longxiang Gao. Asynchronous federated learning on heterogeneous devices: A survey. *arXiv preprint arXiv:2109.04269*, 2021.

[9] Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Animashree Anandkumar. signsgd: Compressed optimisation for non-convex problems. In *International Conference on Machine Learning*, pages 560–569. PMLR, 2018.

[10] Jeremy Bernstein, Jiawei Zhao, Kamyar Azizzadenesheli, and Anima Anandkumar. signSGD with majority vote is communication efficient and fault tolerant. In *International Conference on Learning Representations*, 2019.

[11] Chenhao Xu, Youyang Qu, Tom H Luan, Peter W Eklund, Yong Xiang, and Longxiang Gao. An efficient and reliable asynchronous federated learning scheme for smart public transportation. *arXiv preprint arXiv:2208.07194*, 2022.

[12] Yujing Chen, Yue Ning, Martin Slawski, and Huzefa Rangwala. Asynchronous online federated learning for edge devices with non-iid data. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 15–24. IEEE, 2020.

[13] Muhammad Habib ur Rehman, Khaled Salah, Ernesto Damiani, and Davor Svetinovic. Towards blockchain-based reputation-aware federated learning. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 183–188. IEEE, 2020.

[14] Jiasi Weng, Jian Weng, Jilian Zhang, Ming Li, Yue Zhang, and Weiqi Luo. Deepchain: Auditable and privacy-preserving deep learning with blockchain-based incentive. *IEEE Transactions on Dependable and Secure Computing*, 2019.

[15] Muhammad Shayan, Clement Fung, Chris JM Yoon, and Ivan Beschastnikh. Biscotti: A blockchain system for private and secure federated learning. *IEEE Transactions on Parallel and Distributed Systems*, 2020.

[16] Jin-Hua Chen, Min-Rong Chen, Guo-Qiang Zeng, and Jia-Si Weng. Bdfl: a byzantine-fault-tolerance decentralized federated learning method for autonomous vehicle. *IEEE Transactions on Vehicular Technology*, 70(9):8639–8652, 2021.

[17] Youyang Qu, Chenhao Xu, Longxiang Gao, Yong Xiang, and Shui Yu. Fl-sec: Privacy-preserving decentralized federated learning using signsgd for the internet of artificially intelligent things. *IEEE Internet of Things Magazine*, 5(1):85–90, 2022.

[18] Sai Praneeth Karimireddy, Quentin Rebjock, Sebastian Stich, and Martin Jaggi. Error feedback fixes signsgd and other gradient compression schemes. In *International Conference on Machine Learning*, pages 3252–3261. PMLR, 2019.

[19] Afshin Abdi and Faramarz Fekri. Quantized compressive sampling of stochastic gradients for efficient communication in distributed deep learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):3105–3112, Apr. 2020.

[20] Youyang Qu, Longxiang Gao, Yong Xiang, Shigen Shen, and Shui Yu. Fedtwin: Blockchain-enabled adaptive asynchronous federated learning for digital twin networks. *IEEE Network*, 2022.

[21] Yinghui Liu, Youyang Qu, Chenhao Xu, Zhicheng Hao, and Bruce Gu. Blockchain-enabled asynchronous federated learning in edge computing. *Sensors*, 21(10):3335, 2021.

[22] Wentai Wu, Ligang He, Weiwei Lin, Rui Mao, Carsten Maple, and Stephen A Jarvis. Safa: a semi-asynchronous protocol for fast federated learning with low overhead. *IEEE Transactions on Computers*, 2020.

[23] Chenhao Xu, Youyang Qu, Tom H. Luan, Peter W. Eklund, Yong Xiang, and Longxiang Gao. A lightweight and attack-proof bidirectional blockchain paradigm for internet of things. *IEEE Internet of Things Journal*, 9(6):4371–4384, 2022.