PIRATE: A Blockchain-based Secure Framework of Distributed Machine Learning in 5G Networks

Sicong Zhou, Huawei Huang, Wuhui Chen, Pan Zhou, Zibin Zheng, Song Guo

***Abstract*—In the fifth-generation (5G) networks and the beyond, communication latency and network bandwidth will be no more bottleneck to mobile users. Thus, almost every mobile device can participate in the distributed learning. That is, the availability issue of distributed learning can be eliminated. However, the model safety will become a challenge. This is because the distributed learning system is prone to suffering from byzantine attacks during the stages of updating model parameters and aggregating gradients amongst multiple learning participants. Therefore, to provide the byzantine-resilience for distributed learning in 5G era, this article proposes a secure computing framework based on the sharding-technique of blockchain, namely PIRATE. To prove the feasibility of the proposed PIRATE, we implemented a prototype. A case-study shows how the proposed PIRATE contributes to the distributed learning. Finally, we also envision some open issues and challenges based on the proposed byzantine-resilient learning framework.**

I. Introduction

Machine learning has spawned a lot of useful applications, such as computer vision, and natural language processing, etc. However, the parties who benefit from the technology are mostly large organizations, e.g., commercial companies and research institutes. Individuals with limited computing-resource cannot take part in machine learning tasks.

In fact, the combined power of individuals has been much underestimated. Taking these advantages into account, distributed learning [1] enables individual devices to learn collaboratively.

Generally, factors that affect an ideal distributed learning are included as follows:

- High availability: any device can perform learning anytime.
- High scalability: the learning framework should support high concurrency, high communication efficiency and low storage complexity.
- Decentralization: the intervention of a centralized third-party is minimum.
- Byzantine-resilient model safety: the future distributed learning should ensure byzantine-resiliency [2], which indicates that the distributed learning can endure arbitrary attacks on learning convergence.

S. Zhou, H. Huang, W. Chen and Z. Zheng are with the School of Data and Computer Science, Sun Yat-Sen University, China. Email: huanghw28@mail.sysu.edu.cn

P. Zhou is with School of Electrical Information and Communication Engineering, Huazhong University of Science & Technology, Wuhan, 430074, China.

S. Guo is with the Department of Computing, the Hong Kong Polytechnic University. Email: song.guo@polyu.edu.hk

Communication latency and bandwidth are still viewed as the bottleneck of distributed machine learning [3]. This situation makes distributed learning highly unavailable for the majority.With significantly improved network conditions, 5G technologies enable high availability.

As do all distributed systems, the learning system is prone to *byzantine attacks* [2], especially when the availability is high. Therefore, more sophisticated approaches that can ensure the byzantine-resilience are required.

Recently, byzantine-resilient machine learning under master/slave settings has gained much attention [4]–[6]. In a byzantine-resilient distributed-learning task, two things are risk-prone: 1) gradient aggregation, and 2) model parameters.

Conventionally, the byzantine-resilient distributed-learning tasks are conducted under centralized settings, in which the byzantine-tolerant components rely on a globally trusted third-party as the *parameter server*. The problem is that the workload-handling capacity of such centralized parameter server is usually a bottleneck while performing the distributed learning. Moreover, to provide reliable services against the vulnerability of single-point-of-failure (SPOF), the redundant deployment of resources is entailed at the centralized third-party. Therefore, the centralized byzantine-resilient learning induces a high operational-expenditure (OPEX).

To achieve high availability while fulfilling the requirement of OPEX-efficiency and byzantine-resiliency, the distributed learning system would ideally be decentralized. However, to the best of our knowledge, the byzantine-resilient learning with decentralized configuration has not been well studied.

To this end, this paper proposes a sharding-based blockchain framework named PIRATE, for byzantine-resilient distributed-learning under the decentralized 5G environment to protect learning convergence. Decentralized convergence-security, and trusted credit feedback for candidate management are the key features of PIRATE.

II. Preliminaries of Distributed Machine Learning in 5G Networks

A. Consensus Protocols for Decentralized Learning in 5G

To achieve global agreement within a decentralized setting, a byzantine-tolerant consensus protocol for state machine replication (SMR) is needed [2]. The consensus protocol should ensure that honest nodes can reach agreement on the order and the correctness of model updates, even when a certain amount of byzantine effort exists. In this section, we first briefly review the existing byzantine tolerant consensus protocols towards SMR, and then analyze what protocol

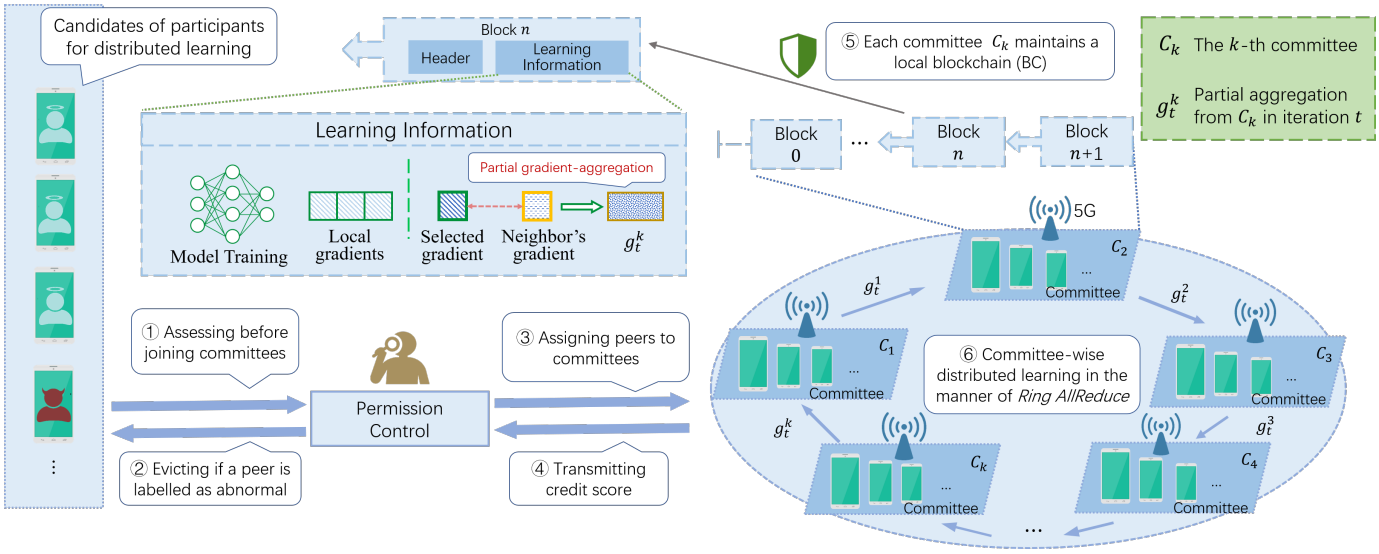

Fig. 1. The proposed PIRATE framework has two critical components: 1) permission control, and 2) blockchain-based learning committees for distributed SGD (D-SGD). Gradient aggregations and model parameters are protected by Hotstuff blockchain consensus protocol. Meanwhile, the permission control centre determines the joining and leaving of candidates.

is applicable for decentralized learning in the 5G era. For brevity, we call the byzantine-tolerant consensus protocol the consensus protocol in the remainder of this paper.

Consensus protocols can be categorized into 2 types: the competition-based and the communication-based. The leader of competition-based consensus, e.g., the Proof of Work (PoW) adopted by bitcoin [7], needs to earn his leadership through a "fair" competition. Communication-based consensus protocols (e.g., Hotstuff [8]) select leaders through a deterministic way, or based on an unbiased randomness generated collaboratively.

In the context of blockchains, for competition-based consensus protocols, blocks are appended on the chain *before* consensus while the communication-based protocols append blocks thereafter. For competition-based methods, larger scales inevitably incur higher chance of forking. Plagued by the byproduct, competition-based methods struggle to achieve a high scalability.

While communication-based consensus protocols have no concern of forks, they require multiple rounds of communication to reach agreement. The high communication overhead also hurts the scalability of communication-based protocols.

Sharding-based consensus protocols [9] can achieve scalable consensus in the permissionless blockchain. They benefit from the instant finality of the communication-based methods, and permissionless resiliency of the competition-based methods.

In 5G networks, we need a consensus protocol that is available for a large scale of participants. An ideal consensus protocol should be scalable and permissionless. Sharding-based protocols make full use of the integrated resources by splitting a workload and allocating tasks among multiple committees. Such scalable strategy can fit perfectly in distributed learning. However, permissionless distributed-learning is challenging due to the volatile states of participants. Thus, a consistent assessment of training-reliability is required for the learning system task to function efficiently. Accordingly, we adopt a permissioned version of sharding-based consensus protocol in our proposed framework.

B. Configurations of Distributed Machine Learning

Machine learning problems mostly rely on some optimization problems. In order to orchestrate a distributed optimization, as shown in Fig. 1, computing nodes need to carry out the following steps. First, each node computes local gradients. Then, nodes communicate with aggregator(s) to get a globally aggregated gradient for model update. Based on how they communicate, two typical styles of configuration have been proposed:

Master/slave style: A centralized parameter server aggregates gradients and sends the aggregated result to each computing node.

Decentralized style: All nodes are aggregators. In an Allreduce manner, nodes communicate with neighbors to exchange gradients.

We then discuss the superiority of decentralized style:

- **Communication efficiency**. A recent work [3] demonstrated that the decentralized settings can better exploit the bandwidth resource, avoid traffic jams and share workloads among peers than the centralized master/slave settings.
- **Cost efficiency**. When the scale of participation grows substantially, no single party should be responsible for maintaining the system. Analogous to the situation of cloud netdisk services nowadays, if a service provider plays such a dominant role, the OPEX cost of the centralized system eventually transfer to clients. As a feasible solution, service providers enforce clients to choose between low quality of service and expensive membership fee. This is an opposite of win-win for providers and clients, provided that there are better solutions.
- **Reliability**. The centralized design of master/lave settings suffer from SPOF problem. Once the centralized server

is overloaded or under attack, the whole system ceases to function. Thus, the communication burdens and the attack risks on one single server impair the reliability of the system.

III. State-of-the-art studies of Byzantine-Resilient Machine Learning

A. Byzantine-Resilient Machine Learning

As the scale of participants grows, the behaviors of computing nodes become more unpredictable. The distributed stochastic gradient descent (D-SGD) [10] framework should tolerate byzantine attacks, i.e., arbitrary malicious actions to prevent convergence, such as sending harmful gradients and corrupting training models as shown in Fig. 2. Current studies on byzantine-resilient machine learning mostly focus on protecting gradient aggregation. However, the parameters of training models owned by each data trainer are also vulnerable. We analyze existing frameworks that can protect both, and elaborate the protection of gradient aggregation in the next section.

A blockchain-based method, *LearningChain*, was proposed by Chen *et al.* [11], which is able to simultaneously protect gradient aggregations and model parameters, by storing them together on-chain. Exploiting the traceability of blockchain, erroneous global parameters can be rolled back to its unfalsified state. Historical parameter records cannot be falsified due to the tamper-proof characteristic of blockchain. They proposed "l-nearest gradients aggregation" to ensure that if byzantine computing nodes yield local gradients to prevent convergence of the learning algorithms, their effort would be mitigated. However, *LearningChain* still utilizes a master/slave setting for D-SGD where the parameter server is elected by PoW competition. In addition, the on-chain data could be potentially oversized, because all nodes would have to store all the historical model parameters and gradients. Such architecture is prone significant traffic congestion and substantial storage overhead.

In terms of reliability, rollbacks can possibly fail if two consecutive byzantine leaders collude. The essential problem is that, the model update is examined by only one leader when a proposal is submitted. In contrast, our proposed framework adopts a more decentralized setting, in which all nodes can naturally participate in validating the model updates, and every node maintains its own training model.

B. Byzantine Protection on Gradients

Before updating training models, computing nodes need to aggregate their local gradients. Aggregation solutions of simple linear combinations (e.g., averaging [12]) cannot tolerate one byzantine worker [4]. Thus, byzantine protection on gradients aggregation has gained much growing attention. Basically, the existing byzantine-based approaches can be classified into 2 categories: the tolerance-based and the detection-based.

Blanchard *et al.* [4] proposed a byzantine tolerant method called *Krum*. Instead of using a simple linear combination, Krum precludes gradients too far away from the majority and chooses one local gradient based on a spatial score. Experiments show that even with 33% of omniscient byzantine workers, the error rate is almost the same as that of 0%.

The tolerant approach l-nearest gradients proposed by Chen *et al.* [11] cannot guarantee safety against omniscient attacks. The aggregation solution is to aggregate l gradients closest, based on their cosine distances, to the sum of the received gradients. If an omniscient attacker manages to acquire all local gradients for other workers in time, the byzantine worker can yield a local gradient that changes the global aggregation arbitrarily [4].

One byzantine-detection method proposed by Li *et al.* [6] is designed for federated learning (FL). Existing byzantine-tolerant aggregation methods are mostly inefficient due to the non-identically and independently distributed training data. Experiments show that their detection-based method has a better performance than tolerance-based methods in FL. In their algorithm, a credit score was assigned by a pre-trained anomaly detection model to each local gradient. Since the weight of the local gradient was determined by the credit score, the weighted sum aggregation can filter out the byzantine local gradients.

Apart from detection methods, another machine learning method was proposed by Ji *et al.* [13] to learn the gradient aggregation. Different from the deterministic aggregation solution, they model the aggregation as a learning problem.

Tolerance-based methods are mostly designed under an independent identically distributed (i.i.d) assumption. Therefore, in the settings of FL where data is non-i.i.d, most tolerance-based methods do not perform well. However, tolerance-based methods have the benefit of simplicity that do not require additional training.

As shown in Table. I, we compare the performance of different protection approaches for gradient-aggregation under the normal setting and the FL setting.

C. Risks of Decentralization

In a decentralized scheme, every node has a greater impact on the global aggregation than a centralized scheme. As shown in Fig. 2, the existing protection methods are not applicable to the decentralized settings because of the following reasons.

- Attacks on partial aggregation are detrimental. Every node aggregates a partial aggregation provided by another node. For byzantine nodes, they have more attack patterns, such as sending falsified partial aggregation results or sending nothing to stall the aggregation process. Thus, without a quorum of validators, partial aggregation process cannot be trusted.
- Anomaly detection [6] would be challenging. Credit scores cannot be trusted without proper validation mechanisms.
- Synchronization of model parameters could be a problem. Every node needs to maintain a local training model. Once contaminated by attackers, computing nodes will have no actual contribution to the holistic learning task.

As a solution, blockchain as a decentralized SMR system, can provide quorums of validators and practical synchroniza-

TABLE I
GRADIENT PROTECTION METHODS

Method types	The tolerance-based		The detection-based	The learning-based
Representative studies	Krum [4]	l-nearest gradients [11]	Anomaly Detection [6]	Learning to learn [13]
Resiliency under 30% attack	High	Medium	Unknown	High
Resiliency under 30% attack (FL)	Low	Unknown	High	Unknown
Resiliency under majority attack	Low	Medium	Unknown	High
Resiliency under majority attack (FL)	Low	Unknown	High	Unknown
Computation complexity	O(n^2)	O(n)	O(n)	Model-related
Other functions except *aggregation*	None		Autoencoder training	Aggregator training

tion mechanism to achieve byzantine-resiliency in decentralized learning.

IV. Our Proposal - PIRATE: A Machine Learning Framework based on Sharding Technique

A. Overview of PIRATE

Generally, we propose a framework of blockchain-based protection for the distributed machine learning named PIRATE, targeted for the convergence risks brought by decentralized D-SGD. PIRATE consists of two major parts, i.e., a permission control center and learning committees. The permission control center provides reliability assessment to candidates and assigns candidates into learning committees. Meanwhile, learning committees collaborate to solve a decentralized D-SGD problem with additional verification from blockchains and anomaly detection. As a result, malicious nodes are replaced and their harmful gradients are filtered.

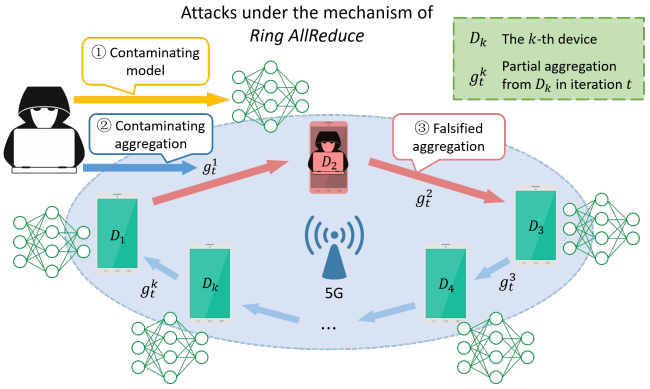

Fig. 2. While adopting the *Ring Allreduce*[1] mechanism, malicious attackers can perform attacking from both inside and outside. ① Attackers from the outside can contaminate training models in target nodes. ② The outside attackers can also attack partial gradient-aggregation. ③ Byzantine computing nodes can send harmful aggregations that damage the convergence of learning tasks.

B. Permission Control

In a distributed learning system with high availability, reliability assessment is essential, especially for mobile devices. Allowing devices in bad states to participate learning tasks would slow down the entire learning process. Thus, real-time reliability assessment and permission control are needed.

We propose a centralized solution for permission control. Fig. 1 depicts the permission control of PIRATE. Before actually contributing to the global learning task, all candidates are assessed by a *permission control center* based on their computation ability, network condition, join/leave prospect and historical credit scores. Accordingly the permission control center determines whether a candidate can join a learning task and the workload to assign. If granted permission, candidates are to replace nodes with low accumulated credit scores during reconfiguration. During training, validated credit scores generated by committees are transmitted to the permission control center.

C. Sharding-based Blockchain Protection towards Decentralized Distributed-Learning

We propose a sharding-based blockchain mechanism for the protection of decentralized distributed-learning. We randomly split the computing nodes into multiple committees, in which partial aggregations are agreed. No longer centralized, the burden of aggregation workloads is mitigated.

Let n denote the total number of computing nodes, c denote the size of a committee. We then discuss the key actions of the learning committees.

Random committee construction. All nodes would be assigned a random identity by the permission control center. According to the identities, committees of size c are formed. Every committee member knows the identity of all honest peers in their committee and their neighbor committees.

Intra-committee consensus. Every committee maintains a separate blockchain with Hotstuff [8] to reach consensus on local aggregations, training models and credit scores. With an honest majority of members having agreed and partially signed (threshold signature) on the data, the data is tamper-proof.

Global consensus. In a committee-wise Ring Allreduce[1] manner, committees communicate locally-agreed aggregations and their threshold signatures with their neighbor committees. Since committee members know all honest members in their neighbor committees, by checking whether an aggregation is signed by an honest majority, committee members can verify the aggregation from neighbors. With $2(n/c-1)$ consensus steps, every node would have a globally agreed aggregation,

[1]https://github.com/baidu-research/baidu-allreduce

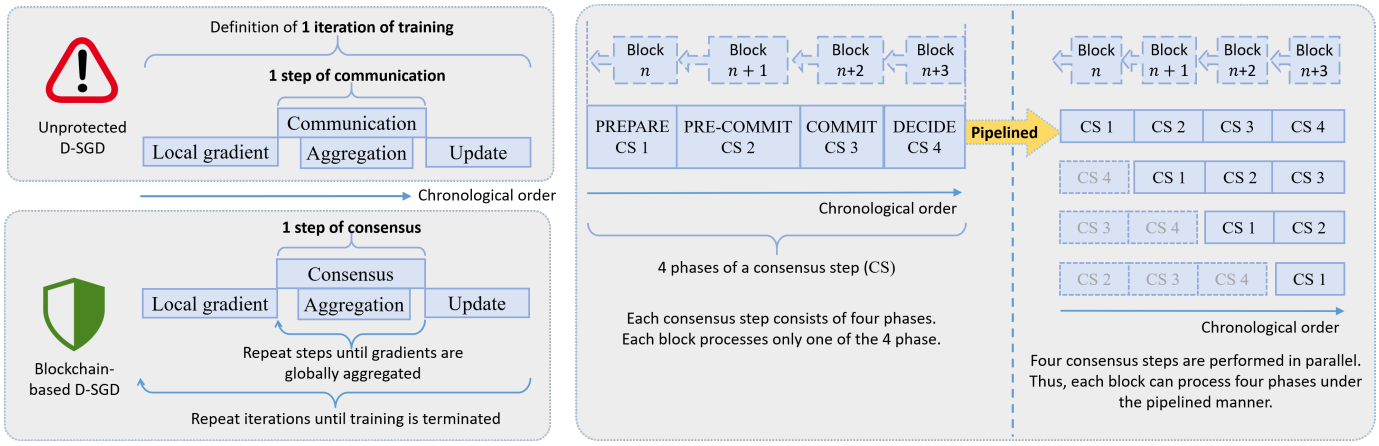

Fig. 3. We define the process of finishing a partial aggregation a *step*, the process of finishing a model update an *iteration*. A consensus step (CS) has 4 phases, each driven by leader. The protocol can be pipelined for performance enhancement, ideally executing 1 aggregation each block in average.

completing one iteration.

Reconfiguration. After finishing a learning task, nodes with low credit scores in every committee are replaced. The permission control center would consider these nodes as byzantine nodes and replace them with new nodes. Following Bounded Cuckoo Rule [9], old and new nodes in a committee are randomly evicted to some other random committees. The random eviction rule provides join/leave attack protection while guaranteeing a same resiliency of 1/3 in each committee.

D. Intra-Committee Consensus

As in Fig. 1, each committee maintains a blockchain to protect aggregations and training models within the committee. We adopt Hotstuff [8] as the consensus protocol of the blockchain. Fig. 3 depicts the intra-committee consensus process. We define the process of finishing a partial aggregation a *step*, the process of finishing a model update an *iteration*.

When a local gradient is computed, members take part in repeated consensus steps. A consensus step (CS) includes:

- **Component 1**: local gradient selection,
- **Component 2**: neighbor committee aggregation,
- **Component 3**: aggregation of Component 1 and Component 2.

For component 1, committee members can either collaboratively select c^2/n local gradients, or coordinate in a round robin manner to choose c^2/n local gradients.

For component 2, members wait for the leader of the neighbor committee C_{i-1} to broadcast the tamper-proof neighbor aggregation from last step. Since the aggregation is an agreed result from the neighbor's last step, members of this committee C_i can verify the result by checking the threshold signature. If the leader of C_{i-1} chooses to withhold the result from C_i, members of C_i would send requests to a random member of C_{i-1} for the result.

Finally for component 3, having a neighbor committee aggregation and a set of local gradients, nodes aggregate them using the detection-based BFT aggregation [6]. A pre-trained anomaly detection model would assign a weight to each proposed gradient according to the anomaly score. Zero weight would be assigned to a proposed gradient if its anomaly score surpasses a threshold, thereby harmful gradients that hinders convergence are "filtered". Meanwhile, members are required to validate and store historical credit scores of each other until a new committee is formed. These verifiable credit scores are transmitted to the permission control center during reconfiguration.

After all components being executed, the incumbent leader broadcast the partial aggregation and a digital digest of training parameters for members to verify and agree on. Having sufficient signatures, the leader would broadcast the decided aggregation in its committee C_i and in the neighbor committee C_{i+1}.

With Hotstuff [8] a consensus step requires four phases of communication, i.e., PREPARE, PRE-COMMIT, COMMIT and DECIDE. Each phase is driven by a leader issuing a block containing verifiable threshold signatures.

As shown in Fig. 3, only 1/4 of blocks are generating aggregations. To address this issue, we can pipeline consensus steps to achieve a better performance. Each leader would be responsible for driving four consensus steps in different phases. In an ideal scenario where no byzantine leader is elected, every block generates an agreed aggregation in average.

Members are required to store one set (4 sets if pipelined) of gradients for validation, which is composed of its own local gradient, aggregation of the corresponding neighbor committee and an aggregation proposal from the incumbent leader.

E. Committee-wise Ring Allreduce

In PIRATE, we adopot a committee-wise Ring Allreduce as the decentralized communication scheme. The committee-wise Ring Allreduce enables verification of aggregation in a fully decentralized setting. For classic Ring Allreduce, we refer our readers to baidu-allreduce[1]. The worker unit of committee-wise Ring Allreduce is no longer a processing unit, but an entire committee. Also, instead of segmenting one single gradient for transmission of each round as classical Ring Allreduce does, a committee transmits one whole gradient in

each round (consensus step) by controlling the selection ratio n/c^2 to 1 ($1/4$ if pipelined). The selection ratio ensures that the amount of whole gradients to transfer equals to the number of committees. After $2(n/c-1)$ rounds, all local gradients are aggregated.

F. Security and Complexity Analysis

1) Security Analysis of Convergence Attack: As shown in Fig. 2, various attack behaviors are considered. Training models and aggregations are two main targets of attack.

Training models are maintained by all computing nodes. Since training models are on-chain information, computing nodes can quickly recover an approved training model once it is contaminated. With Hotstuff [8], the recovery mechanism only fails if the committee is composed by over 33% of byzantine nodes.

Both local gradients and partial aggregations could be contaminated or falsified within a certain committee, either by outsiders or malicious participants.

When local gradients are contaminated, they would be effectively filtered by the gradient anomaly detection if the percentage of the contaminated gradients is less than 30% [6]. In terms of partial aggregations within committees, with each committee running consensus protocol that only approve aggregations with authenticators (i.e., threshold digital signatures), contaminated aggregations would not be accepted by committee members. And again, such mechanism only fails if the committee is composed by over 33% of byzantine nodes.

An authenticated partial aggregation is to be broadcast to members of the neighbor committee. Similar to the above mechanisms, passing partial aggregation to neighbors fails if the committee is composed by over 33% of byzantine nodes.

2) Security Analysis of Take-over Attack: In terms of take-over attacks, we adopt the Bounded Cuckoo Rule for reconfiguration, which is proven to keep committees *balanced* and *honest* in [9]. *Balanced* refers that the number of nodes in the committee is bounded , and *honest* refers that the fraction of byzantine nodes is less than 1/3. Given the two properties, the reconfiguration mechanism shields the system from take-over attacks.

3) Computation Overhead: Extra computation cost mainly comes from generating digital digests and verifying them. We adopt Hotstuff as the consensus protocol, which has a complexity of O(n) [8]. Such overhead is insignificant compared to gradient broadcasting. For instance, generating a 100 MB Merkle tree with a 3.5GHz processor and PySHA-3 would roughly takes 1 second, and verifying the tree would take way less than 1 second [14].

G. Applications

1) Decentralized Federated Learning: In FL, data privacy are protected with differential privacy mechanisms. In terms of convergence safety, FL relies on byzantine-resilient centralized D-SGD algorithms like [6] for protection. However, most of these algorithms alone cannot provide protection in a decentralized setting. PIRATE solves this problem with blockchains. Meanwhile, with anomaly detection and consensus mechanisms of PIRATE, verifiable credit scores can be utilized as a powerful index for client selection, a crucial stage of FL.

2) Big Data Analysis for Consortium Blockchains: Consortium blockchains are widely used in industry organizations for the benefits of a shared governance. With PIRATE, organizations can continuously conduct secure big data analysis using decentralized D-SGD on the shared data. The learnt results are trustworthy in an environment where learning devices owned by different organizations do not have to trust one another. Each organization is required to maintain its own permission control centre. Utilizing the credit score feedback, each organization would always have their most reliable devices on duty.

V. Case Study

We implemented a prototype of PIRATE based on Hotstuff[2]. To further evaluate the performance of PIRATE in a large-scale scenario, we conducted a simulation.

A. Security

To demonstrate feasibility and verification-based security of PIRATE, we conducted an experiment on the prototype. As shown in Fig. 4 we experimented on 6 instances, four of which composes Committee1, and the rests are Neighbor1 and Neighbor2. In a committee-wise Ring Allreduce manner, Committee1 and its neighbors communicate with verifiable aggregations. The training process is omitted in the experiment for simplicity.

In Fig. 4 variables like `hqc`, `b_lock` and `vheight`, we refer our readers to [8]. Green colored line are the execution logs of intra-committee consensus phases, i.e., PREPARE, PRE-COMMIT, COMMIT and DECIDE. Other colored lines correspond to the verifiable decisions on aggregations of Committee1, Neighbor1 and Neighbor2. Contaminated decisions would be detected with the given information.

B. Performance Evaluation

We conducted a simulation for performance evaluation on pipelined PIRATE in a large scale scenario. On one machine, we ran 50 to 100 instances to simulate a single committee. According the selection ratio n/c^2, we can speculate the performance of 625 to 2500 nodes in total due to the concurrent nature of PIRATE. We assume devices spend a same amount of time for computing gradients. We simulate this process by having instances wait for a same amount of time to generate an equal-sized chunk of data (28 MB). Then, instances transmit chunks of data to simulate the decentralized D-SGD process of PIRATE. The machine is a mini PC (model serial number: NUC8i5BEK), with a quad-core i5-8259U processor (3.80 GHz). To simulate the network condition of 5G, we assume every message has a 10 ms latency. And the uplink bandwidth is uniformly-distributed ranging from 80 Mbps to 240 Mbps, while the downlink bandwidth is set to 1 Gbps.

[2]https://github.com/hot-stuff/libhotstuff

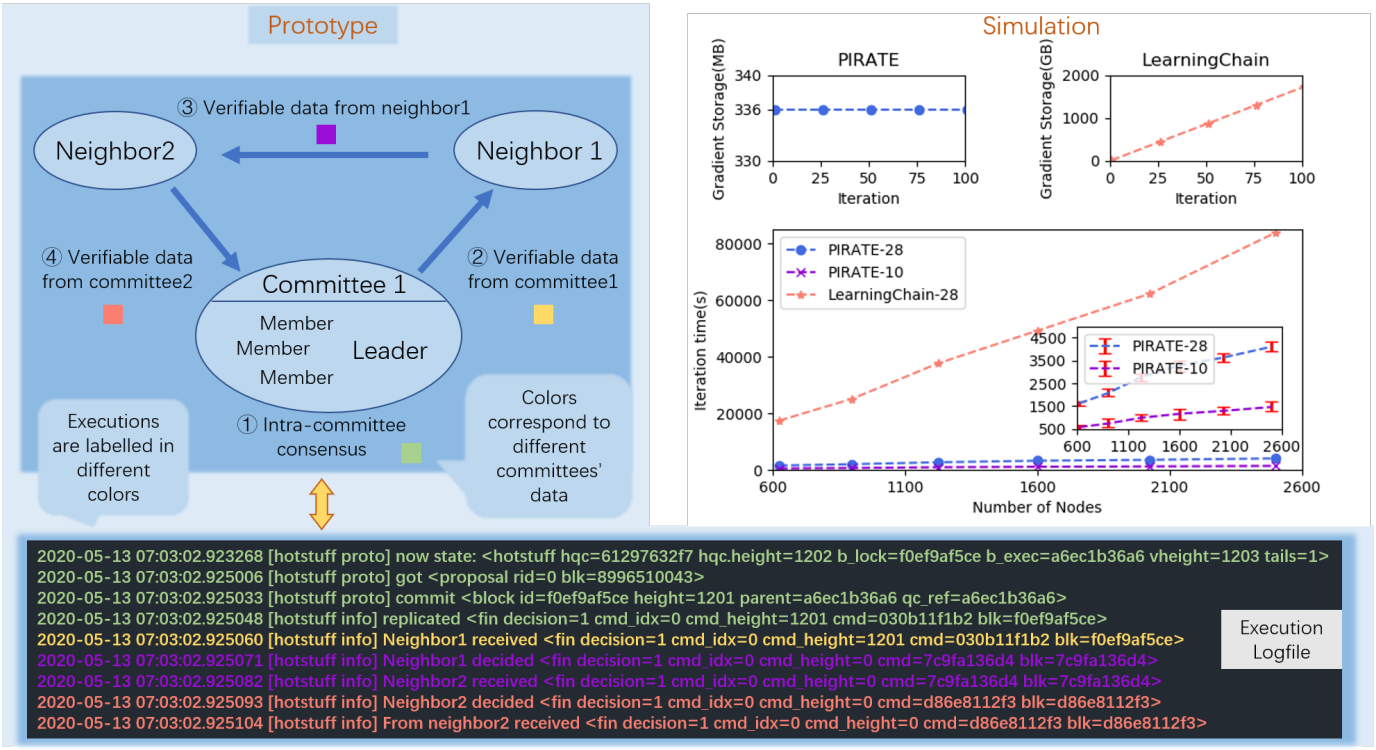

Fig. 4. Efficiency as shown in the right-top section, is evaluated with a simulation. The rest of the figure demonstrates how the prototype works. In the simulation section, the left-top figure: Gradient Storage of PIRATE vs. iteration with single gradient size of 28 MB. The right-top figure: Gradient Storage of LearningChain vs. iteration with the single-gradient size of 28 MB. The Bottom figure: Iteration time of PIRATE and LearningChain vs. number of nodes with the single-gradient sizes of 28 MB and 10 MB. In the prototype section, the upper part is the abstraction of the whole process. The lower part is the execution results recorded in a logfile.

We compare PIRATE with another blockchain-based D-SGD framework LearningChain without the presence of malicious node. We first compare the gradient storage overhead of the two frameworks. We then compare the iteration time measured by the time used to broadcast a block.

As Fig. 4 shows, gradient storage overhead for PIRATE are constant as iteration progresses, while LearningChain's storage has a linear growth. In each iteration, PIRATE stores only the leader's gradient, the neighbor committee's gradient and the local gradient itself has computed. In LearningChain, nodes are required to store the history of all leader-announced gradients, and the local gradients broadcast by all nodes.

Fig. 4 shows that PIRATE outperforms LearningChain on iteration time. The major cost in each iteration is the broadcast of gradients. PIRATE shows a superior performance in terms of iteration time for each committee. This is because nodes are required to broadcast to only c members, meanwhile, consensus decisions are reached concurrently.

VI. Open Issues

The open issues are envisioned as follows.

- **Decentralized Permission Control**. Candidates having inferior computation ability, bad historical credit scores and unstable network conditions can undermine the efficiency of distributed learning. Without a centralized permission control, reliability assessment is challenging, especially for realtime attributes. Latency induced by decentralized communication and verification inevitably affects timeliness.
- **Protection Against Model Poisoning Attack**. When applied to FL, PIRATE faces a challenging issue of model-poisoning attack. The attack can be successful even for a *highly constrained* byzantine node [15]. By exploiting the non-i.i.d property of data shards, byzantine attackers can send poisoned local updates that do not hurt convergence. Such harmful local updates can still affect the global model that triggers misclassifying. The attack is also "sneaky" enough to bypass accuracy checks from central servers. In a decentralized environment, where nodes are less constrained in terms of communicating, computing and validating, model poisoning attack can be even more threatening.
- **Privacy Protection**. When computing nodes train their own data and upload training models for aggregations (like FL), it is possible for attackers to reconstruct the private data using gradient information. For privacy protection, differential privacy mechanism is widely used [11]. However, inevitably there is a trade off between a privacy budget and training accuracy. A well-balanced protection mechanism in both privacy and training accuracy is tempting.

VII. Conclusion and Future Work

To guarantee the high availability of distributed learning in 5G era, a distributed-learning framework with high efficiency,

decentralization and byzantine-resiliency is in urgent need. To fill this gap, we propose PIRATE, a byzantine-resilient D-SGD framework under the decentralized settings. Utilizing a sharding-based blockchain protocol, learning convergence can be well protected. A prototype is implemented to show the feasibility of the proposed PIRATE. The simulation results show that PIRATE scales better than the existing solution LearningChain. As future work, we will further analyze the robustness of PIRATE with extensive experiments.

VIII. Acknowledgement

This work is partially supported by National Natural Science Foundation of China (61902445, 61872310), partially by Fundamental Research Funds for the Central Universities of China under grant No. 19lgpy222, and partially by Guangdong Basic and Applied Basic Research Foundation under Grant 2019A1515011798.

References

[1] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and distributed computation: numerical methods*. Prentice hall Englewood Cliffs, NJ, 1989, vol. 23.

[2] L. Lamport, "Time, clocks, and the ordering of events in a distributed system," *Commun. ACM*, vol. 21, no. 7, pp. 558–565, Jul. 1978. [Online]. Available: http://doi.acm.org/10.1145/359545.359563

[3] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu, "Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent," in *Advances in Neural Information Processing Systems*, 2017, pp. 5330–5340.

[4] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 119–129.

[5] Y. Chen, L. Su, and J. Xu, "Distributed statistical machine learning in adversarial settings: Byzantine gradient descent," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 1, no. 2, pp. 44:1–44:25, Dec. 2017. [Online]. Available: http://doi.acm.org/10.1145/3154503

[6] S. Li, Y. Cheng, Y. Liu, W. Wang, and T. Chen, "Abnormal client behavior detection in federated learning," *arXiv preprint arXiv:1910.09933*, 2019.

[7] "Bitcoin: A peer-to-peer electronic cash system," 2009. [Online]. Available: bitcoin.org

[8] M. Yin, D. Malkhi, M. K. Reiter, G. G. Gueta, and I. Abraham, "Hotstuff: Bft consensus in the lens of blockchain," 2018.

[9] M. Zamani, M. Movahedi, and M. Raykova, "Rapidchain: Scaling blockchain via full sharding," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2018, pp. 931–948.

[10] M. Zinkevich, M. Weimer, L. Li, and A. J. Smola, "Parallelized stochastic gradient descent," in *Advances in neural information processing systems*, 2010, pp. 2595–2603.

[11] X. Chen, J. Ji, C. Luo, W. Liao, and P. Li, "When machine learning meets blockchain: A decentralized, privacy-preserving and secure design," 12 2018, pp. 1178–1187.

[12] B. T. Polyak and A. B. Juditsky, "Acceleration of stochastic approximation by averaging," *SIAM Journal on Control and Optimization*, vol. 30, no. 4, pp. 838–855, 1992.

[13] J. Ji, X. Chen, Q. Wang, L. Yu, and P. Li, "Learning to learn gradient aggregation by gradient descent," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, 7 2019, pp. 2614–2620. [Online]. Available: https://doi.org/10.24963/ijcai.2019/363

[14] D. Koo, Y. Shin, J. Yun, and J. Hur, "Improving security and reliability in merkle tree-based online data authentication with leakage resilience," *Applied Sciences*, vol. 8, no. 12, p. 2532, 2018.

[15] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," *arXiv preprint arXiv:1811.12470*, 2018.

Sicong Zhou (chowsch2@mail2.sysu.edu.cn) is currently with the School of Data and Computer Science, Sun Yat-Sen University, China. His research interests mainly include distributed learning and blockchain performance optimization.

Huawei Huang (M'16) (corresponding author, huanghw28@mail.sysu.edu.cn) received his Ph.D in Computer Science and Engineering from the University of Aizu, Japan. He is currently an associate professor with the School of Data and Computer Science, Sun Yat-Sen University, China. His research interests mainly include distributed learning and blockchains. He has serves as a visiting scholar with the Hong Kong Polytechnic University (2017-2018); a post-doctoral research fellow of JSPS (2016-2018); an assistant professor with Kyoto University, Japan (2018-2019).

Wuhui Chen (chenwuh@mail.sysu.edu.cn) is an associate professor in Sun Yat-sen University, China. He received his bachelor's degree from Northeast University, China, in 2008. He received his master's and Ph.D. degrees from University of Aizu, Japan, in 2011 and 2014, respectively. From 2014 to 2016, he was a JSPS research fellow in Japan. From 2016 to 2017, he was a researcher in University of Aizu, Japan. His research interests include Edge/Cloud Computing, Cloud Robotics, and Blockchain.

Pan Zhou (M'14) (panzhou@hust.edu.cn) is currently an associate professor with School of Electronic Information and Communications, Wuhan, P.R. China. He received his Ph.D. in the School of Electrical and Computer Engineering at the Georgia Institute of Technology (Georgia Tech) in 2011, Atlanta, USA. He received his B.S. degree in the Advanced Class of HUST, and a M.S. degree in the Department of Electronics and Information Engineering from HUST, Wuhan, China, in 2006 and 2008, respectively. He held honorary degree in his bachelor and merit research award of HUST in his master study. He was a senior technical member at Oracle Inc., America, during 2011 to 2013, and worked on Hadoop and distributed storage system for big data analytics at Oracle Cloud Platform. He received the "Rising Star in Science and Technology of HUST" in 2017. His current research interest includes: security and privacy, big data analytics and machine learning, and information networks.

Zibin Zheng (SM'16) (zhzibin@mail.sysu.edu.cn) received the Ph.D. degree from the Chinese University of Hong Kong, Hong Kong, in 2012.,He is a Professor with the School of Data and Computer Science, Sun Yat-sen University, Guangzhou, China. His current research interests include service computing and cloud computing.,Prof. Zheng was a recipient of the Outstanding Ph.D. Dissertation Award of the Chinese University of Hong Kong in 2012, the ACM SIGSOFT Distinguished Paper Award at ICSE in 2010, the Best Student Paper Award at ICWS2010, and the IBM Ph.D. Fellowship Award in 2010. He served as a PC member for IEEE CLOUD, ICWS, SCC, ICSOC, and SOSE.

Song Guo (M'02-SM'11-F'20) (song.guo@polyu.edu.hk) received his Ph.D. degree in computer science from the University of Ottawa, Canada. He is currently a full professor at Department of Computing, The Hong Kong Polytechnic University. His research interests mainly include cloud and green computing, big data, and cyber-physical systems. He serves as an Editor of several journals, including IEEE TPDS, TETC, TGCN, and IEEE Communications Magazine. He is a senior member of IEEE and ACM, and an IEEE Communications Society Distinguished Lecturer.