Secure and Efficient Federated Learning in Edge Computing

Submitted by Rui Jin
to the University of Exeter as a thesis for the degree of
Doctor of Philosophy in Computer Science, November 2023.

I certify that all material in this thesis which is not my own work has been identified and that any material that has previously been submitted and approved for the award of a degree by this or any other University has been acknowledged.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university.

Acknowledgements

First of all, I would like to express my deepest gratitude to my supervisors Prof. Jia Hu and Prof. Geyong Min, for their continuous support and guidance throughout my PhD. I have been continuously inspired by their passion, knowledge, kindness, creativity, and hard working attitude. Working with them has been one of the best experiences in my life.

I would like to thank my labmates over the years, Dr. Zhengxin Yu, Dr. Zi Wang, Dr. Jin Wang, Dr. Wang Miao, Dr. Haozhe Wang, Dr. Jed Mills, Zhe Wang, Yuhong Jiang, Songyuan Li, Jialin Tian, Jie Gao, and Jiazhen Zhang, for their help during my PhD study. I want to thank my friends Joey, Min, Jiahui, Ying, and Yalan for being there during both joyful and difficult times.

Last but not least, I would like to express my love and gratitude to Yongchao, my dad, and my mum for their encouragement and support in my life. They always support my dreams and ambitions, academically and otherwise, while keeping me grounded. I am so grateful that I have them in my life.

List of Publications

1 **R. Jin**, J. Hu, G. Min and J. Mills, "Lightweight Blockchain-empowered Secure and Efficient Federated Edge Learning," in *IEEE Transactions on Computers*, vol. 72, no. 11, pp. 3314-3325, 2023.

2 **R. Jin**, J. Hu, G. Min and H. Lin, "Byzantine-Robust and Efficient Federated Learning for the Internet of Things," in *IEEE Internet of Things Magazine*, vol. 5, no. 1, pp. 114-118, 2022.

3 **R. Jin**, J. Hu, and G. Min, Robust and Fair Federated Learning based on Shapley Value, (to be submitted to *IEEE/ACM Transactions on Networking*)

4 J. Mills, J. Hu, G. Min, **R. Jin**, S. Zheng and J. Wang, "Accelerating Federated Learning With a Global Biased Optimiser," in *IEEE Transactions on Computers*, vol. 72, no. 6, pp. 1804-1814, 2023.

5 Y. Hou, S. Garg, L. Hui, D. N. K. Jayakody, **R. Jin** and M. S. Hossain, "A Data Security Enhanced Access Control Mechanism in Mobile Edge Computing," in *IEEE Access*, vol. 8, pp. 136119-136130, 2020.

Abstract

Federated Learning (FL) has emerged as a promising paradigm for privacy-preserving Machine Learning (ML). It enables distributed end devices (clients) to collaboratively train a shared global model without exposing their local data. However, FL typically assumes that all clients are benign and trust the coordinating central server, which is unrealistic for many real-world scenarios. In practice, clients can harm the FL process by sharing poisonous model updates (known as *poisoning attack*) or sending counterfeit yet harmless parameters to the central server to obtain the trained global model without actual contribution (known as *free-riding attack*), while the central server could malfunction or misbehave. Moreover, the deployment of FL for real-world applications is hindered by the high communication overhead between the server and clients that are often at the network edge with limited bandwidth.

This thesis aims to develop novel FL approaches toward secure and efficient FL in edge computing. First, a novel lightweight blockchain-based FL framework is devised to mitigate the single point of failure of traditional FL. This is achieved by removing the centralized model aggregation to the distributed blockchain nodes. Incorporating the Inter-Planetary File System and Verifiable Random Function, the proposed framework is energy-efficient and scalable with the blockchain network size. Next, a secure and efficient federated edge learning system is proposed, based on the developed blockchain-based FL framework, with a communication-efficient training scheme to reduce the communication cost of clients and a secure model aggregation protocol to build defense against poisoning attacks. Then, an original Shapley value-based defense mechanism is designed to further enhance the robustness of FL, not only against adversarial poisoning attack but also the stealthy free-riding attack. Extensive experiments show that the proposed approach can detect typical free-riding attacks with high precision and is resistant to poisoning attacks launched by adversarial clients.

Table of contents

List of figures

List of tables

Nomenclature

BSs	Base Stations
ECDSA	Elliptic Curve Digital Signature Algorithm
FedAvg	Federated Averaging
FL	Federated Learning
GDPR	General Data Protection Regulation
GHOST	Greedy Heaviest-Observed SubTree
IPFS	Inter-Planetary File System
IoT	Internet of Things
LAG	Lazily Aggregated Gradient
LBFL	Lightweight Blockchain-based FL
ALIE	A Little is Enough
BEFT	Blockchain-Empowered Federated Learning
LAN	Local Area Network
ML	Machine Learning
MEC	Multi-access Edge Computing
MI	Mutual Information
PoA	Proof-of-Authority
PoS	proof-of-stake

PoW	Proof-of-Work
SVRFL	Shapley Value-based Robust Federated Learning
SGD	Stochastic Gradient Descent
VRF	Verifiable Random Function
BFT	Byzantine Fault Tolerant
CS	Central Server
DL	Deep Learning
DPoS	Delegate PoS
DP	Differentially Private
DAG	Directed Acyclic Graph
DHT	Distributed Hash Table

Chapter 1

Introduction

The last few decades have witnessed the rapid development of Machine Learning (ML), which has revolutionized nearly all walks of our lives, including entertainment, transportation, healthcare, and education [106, 84]. This progression in ML is intrinsically tied to the surge in available data and enhanced computational capabilities. ML models, particularly those within the realm of Deep Learning (DL), often demand a large volume of data to train. However, the increasing concern regarding ownership and privacy of personal data, and regulations like the General Data Protection Regulation (GDPR) [103], have spurred the pursuit of privacy-preserving ML. Federated Learning (FL), a distributed ML training paradigm, has emerged as a promising solution. It enables multiple data owners to collaboratively train a shared ML model without disclosing their local data. Furthermore, the advances in edge computing, which facilitate the ease of computation at the network edge, open up numerous opportunities for FL-based applications. This thesis focuses on developing secure and efficient FL approaches at the network edge.

This chapter is organized as follows. Section 1.1 introduces edge computing paradigm. Section 1.2 describes FL in detail. Section 1.3 points out the research challenges and objectives of this thesis. In Section 1.4, I present the outline of this thesis and the contributions.

1.1 Edge computing

With the exponential growth of the Internet of Things (IoT) and the widespread adoption of 4G/5G networks, the public's habit of accessing and processing data has gradually changed, which brings fast-growing demand for low-latency applications[82]. However, traditional cloud computing, characterized by centralized data processing in remote data centers, faces limitations in meeting these demands. The sheer volume and speed of data generated by IoT devices, ranging from smartphones, smart wearables, and home automation systems to

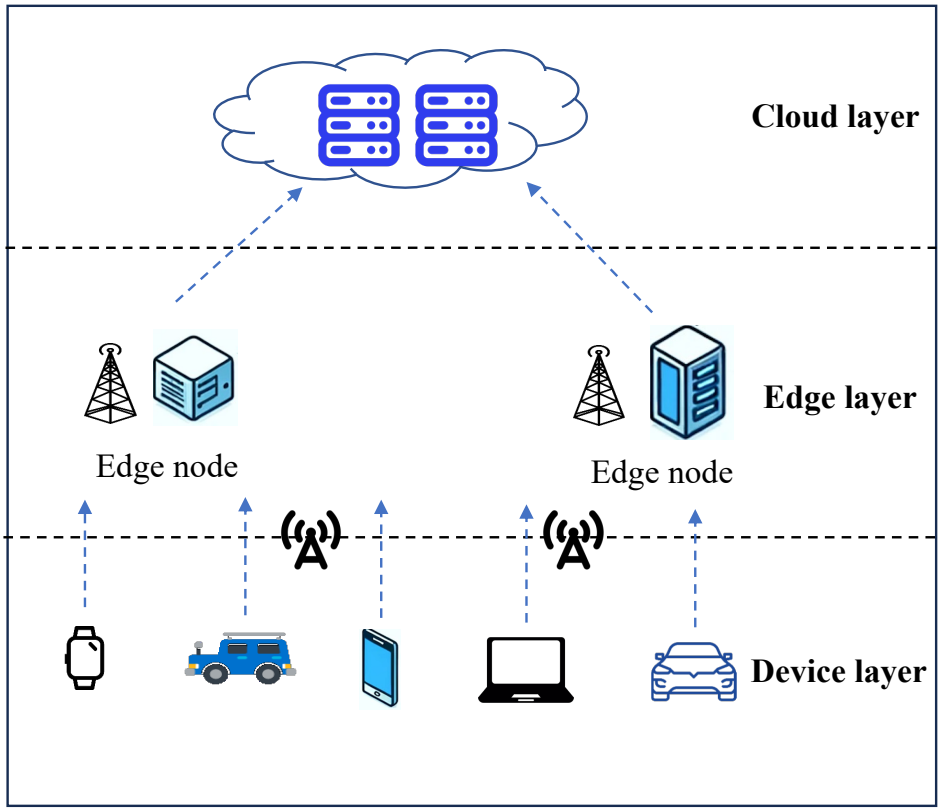

Fig. 1.1 Edge Computing Architecture

industrial sensors and autonomous vehicles, strain the linear scalability of cloud infrastructure, leading to latency issues and bandwidth constraints.

In response to these challenges, a groundbreaking computing paradigm, edge computing, has emerged. Edge computing redistributes computational resources to the edge of the network, closer to where data is generated [111]. The primary goal of edge computing is to reduce latency in data processing and transmission, save bandwidth, and provide a more efficient system for producing insights from data. As shown in Fig 1.1, edge computing could be structured into three hierarchical layers: cloud layer, edge layer, and device layer. The cloud layer is composed of cloud servers equipped with extensive computational resources and storage capabilities [72]. It serves as the backbone of the computing infrastructure, capable of handling large-scale data processing tasks, complex analytics, and long-term data storage. The cloud layer is ideal for non-latency-sensitive applications that require significant computational power and where data locality is not a primary concern. The edge layer contains edge servers that are distributed geographically closer to the data sources. These servers provide localized computational power, allowing for quicker data processing than the cloud layer due to their proximity to the end-users. There are also specialized nodes

for particular tasks like data aggregation, security functions, and network switching, which facilitate the efficient transfer of data between the cloud layer and the device layer. The device layer includes a wide range of IoT devices, smartphones, sensors, and actuators – essentially, any endpoint that generates or consumes data. The devices in this layer can perform basic data processing tasks, such as data collection and preliminary analysis. They are often limited by their computational power and battery life, which makes the edge layer's support crucial. The hierarchical architecture of edge computing offers a flexible, scalable, and efficient solution to cater to the diverse demands of modern computing.

Various edge computing paradigms have been developed, such as cloudlet, fog computing, and multi-access edge computing.

- **Cloudlets**, initially proposed by Satyanarayanan et al. [110], serve as a middle layer between end devices and the cloud. They are lightweight, resource-rich servers positioned at the edge, offering a proximal computing environment for resource-intensive tasks. Meanwhile, cloudlet can also serve users as an independent cloud, making it a "mini-cloud" or a small-scale data center with adequate computing and storage resources. By processing data locally at the edge, rather than in a distant centralized cloud, cloudlets can significantly reduce latency, which is particularly beneficial for real-time applications.

- **Fog Computing** was first coined by Cisco [16], aiming to bring the benefits of cloud closer to end devices. It is a distributed computing paradigm and highly integrated with the cloud, but with the processing tasks also being carried out at the network edge [31]. Fog nodes can encompass various devices, ranging from switches and routers to base stations and resource-rich data centers, with fewer resources than the cloud, but could provide higher bandwidth with less latency for IoT devices as they can be accessed via Local Area Network (LAN) [45, 138, 115].

- **Multi-access Edge Computing (MEC)** was formally introduced by the European Telecommunication Standards Institute to deploy cloud-like computing in the proximity of mobile users [87, 105]. Closely related to fog computing, emphasizes on placing computing capabilities and service environments at the edge of cellular networks[51, 87]. Typically, MEC servers are deployed at macro or micro base stations [105]. By pushing data-intensive tasks toward the edge and locally processing data in proximity to the users, MEC can reduce traffic bottlenecks in the core and backhaul network [115].

With edge computing, data can be processed locally without always needing to be transferred to a centralized cloud, potentially reducing exposure to breaches and yielding enhanced privacy advantages over centralized cloud computing.

1.2 Federated Learning

The vast amounts of data generated by modern devices have propelled the growth of ML models and algorithms that are designed to extract knowledge and derive valuable insights. However, the traditional centralized learning paradigm of ML which requires data to be collected and trained in a single location, often a large-scale data center, becomes harder to deploy, as people are increasingly concerned about data privacy and the inevitable communication cost and security threat in transmitting data to the central server. Federated Learning, first introduced by McMahan *et. al.*[88], becomes a promising solution, which pushes model training back to the data source, with only model updates communicated and aggregated centrally, thereby mitigating the privacy risks and reducing the communication overhead associated with data communication.

The applications of FL roughly fall into two categories: "cross-device" and "cross-silo" [56]. In the "cross-device" setting, FL primarily involves a large number of devices, typically owned by individual users, which contribute to the training of a shared ML model. The "cross-device" aspect refers to the participation of multiple, often heterogeneous devices like smartphones, tablets, wearables, or IoT devices that are distributed across different locations. Several giant companies have applied the cross-device FL to enhance their services, for example, Google has applied FL in android Gboard for next word prediction [1], and Apple leverages FL for applications like the QuickType keyboard and the vocal classifier for “Hey Siri” [10]. In the "cross-silo" setting, a number of organizations or institutions, often referred to as "silos", collaboratively train an ML model while keeping their data localized within their own infrastructure. This is particularly important for industries where data sharing is restricted due to regulatory, competitive, or privacy concerns. Unlike cross-device settings, where there can be significant variability in device capabilities, cross-silo FL usually occurs between participants with more comparable computational resources. The cross-silo FL has attracted substantial attention and has been employed in several industries. For example, WeBank utilized FL for credit risk prediction [3], and ten leading pharmaceutical companies leveraged FL for drug discovery without disclosing proprietary information [89]. This thesis focuses on cross-device FL, where edge computing plays an important role, as real-world FL deployments with a large number of collaborating user devices rely heavily on computing resources available at their devices and inter-device communications [96]. The intermediate

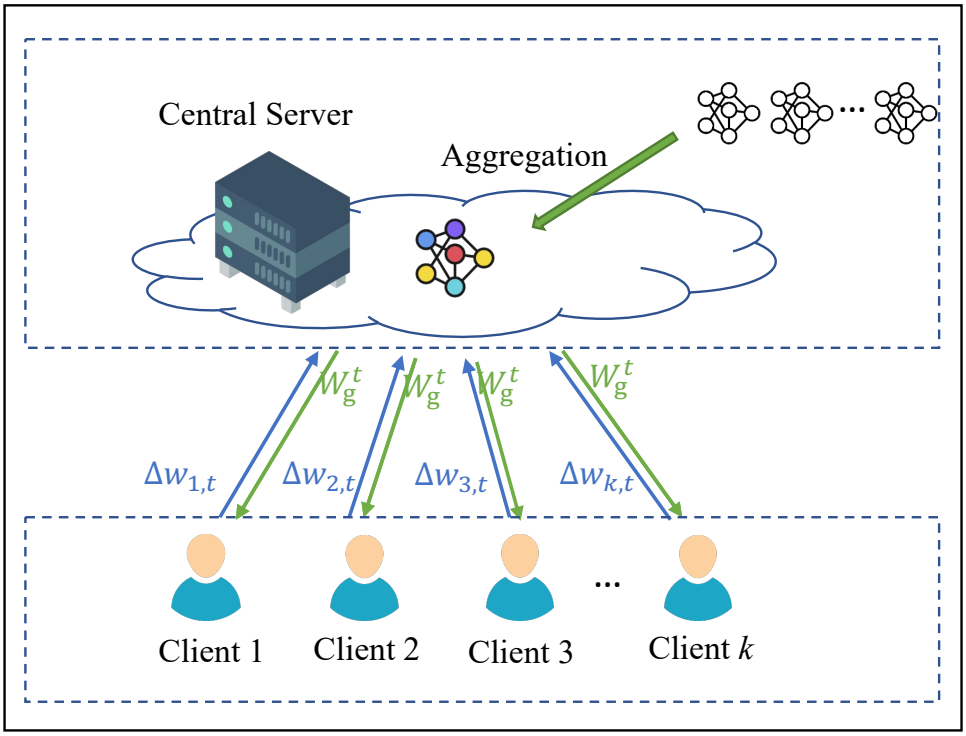

Fig. 1.2 Federated learning process

edge nodes (i.e., Base Stations) equipped with computation, communication, and storage capacity work together with the remote central server to perform large-scale FL tasks [123].

In FL, a central server (CS) orchestrates the collaborative training process. As shown in Fig. 1.2, the general process of FL could be described as follows:

Step 1 **Initialization of task and global model:** In the initial phase, the CS setups the targeted task with its requirements (i.e., network condition and computation capacity of participating devices) and broadcasts the initialized global model to the selected clients.

Step 2 **Local model update:** Given the received global model, each selected client runs local training steps over the private training dataset to update the local model by minimizing the objective loss function and sends the model update back to the CS in parallel.

Step 3 **Global model aggregation:** After receiving model updates from selected clients, the CS performs the aggregation process according to a certain aggregation rule (i.e., the widely adopted *FedAvg* [88]) and sends the updated global model to newly selected clients.

The step 2 and step 3 iterate until certain criteria have been met (i.e., the global model converges or the maximum training round is reached). Through this distributed learning process, FL manages to achieve collective intelligence while preserving participants' data privacy.

1.3 Research Challenges and Objectives

FL has been shown as a promising approach for efficiently harnessing user data while upholding a strong privacy guarantee. Alongside, edge computing has undergone rapid development, which opens up numerous possibilities for FL applications. However, the real-world deployment of FL encounters significant challenges, in terms of efficiency, security, and reliability.

- **Efficiency:** One of the primary efficiency challenges in FL is the communication overhead incurred when clients send their model updates to a central server for aggregation, and the high communication rounds. Transmitting these updates, especially over wireless networks, can be bandwidth-intensive and thus is non-negligible.
- **Security:** FL is vulnerable to byzantine clients who behave arbitrarily [133]. These can include adversarial clients with the aim of impairing global model performance (a.k.a poisoning attack) and free riders who intend to obtain the global model without actually contributing any data or training efforts (a.k.a free-riding attack).
- **Reliability:** The centralized orchestration and aggregation of FL put the CS at a dominant position, rendering the whole learning process to be highly vulnerable to malfunction or misbehaviour of the server. This single point of failure greatly impacts the reliability of the collaborative learning process.

To address these challenges, this thesis aims to develop novel techniques and algorithms to improve the efficiency, security, and reliability of FL, making it more practicable for real-world deployments. In particular, this research aims to:

1. Propose a lightweight blockchain-based FL framework to mitigate the single point of failure of traditional FL.
2. Develop a blockchain-empowered secure and efficient FL system that is robust against poisoning attacks, and cost-efficient to resource-constrained clients.
3. Devise a robust FL mechanism that is not only robust to poisoning attack but also the stealthy free-riding attack.

1.4 Thesis Organisation and Contributions

To achieve the objectives listed in Section 1.3, this thesis makes the following original contributions:

- A lightweight blockchain-based federated learning framework which moves the model aggregation work from the FL server to the blockchain nodes, thus eliminating the single point of failure in traditional FL. This framework also incorporates the Inter-Planetary File System (IPFS) to store the global model with its address recorded on the chain instead of the whole model, to reduce the communication cost of block propagation and the storage cost of the ever-growing blockchain. Moreover, instead of the conventional computation-intensive PoW mechanisms to achieve the consensus in the blockchain, an energy-efficient committee-based consensus protocol using Verifiable Random Function (VRF) is integrated. This new protocol selects committee members with probabilities proportional to their stakes to protect against malicious blockchain nodes and to validate a candidate block that contains the IPFS address of the global model.

- A blockchain-empowered secure and efficient federated learning system based on the proposed framework with a a novel byzantine-resilient communication-efficient training scheme leveraging the Mutual Information (MI) between clients and the state-of-the-art compression method, to protect against poisonous model updates whilst reducing the communication overhead. This new training scheme is embedded in the operating process of the blockchain system.

- A Shapley Value-based Robust Federated Learning (SVRFL) method to build defense against not only poisoning attacks but also the stealthy free-riding attack in FL, with theoretical convergence guarantee. SVRFL leverages Shapley value and cosine similarity to detect free riders in the collaborative learning process and excludes them from the FL system, ensuring the basic fairness of FL. It further employs Shapley value to calculate utility scores of local model updates to filter out potential poisonous ones.

The rest of the thesis is organised as follows:

- **Chapter 2** first introduces the background of this thesis, including blockchain technology and the objective, optimization, and vulnerabilities of FL. Next, a comprehensive review of blockchain-based FL, defense mechanisms that are designed to secure FL, and strategies for communication-efficient FL are presented.

- **Chapter 3** develops an origin blockchain-based federated learning framework to reduce the computation and communication cost of conventional Proof-of-Work (PoW)-based blockchain and mitigate the single point of failure of traditional FL.
- **Chapter 4** presents a novel blockchain-empowered federated learning system based on the framework developed in Chapter 3, which is robust to poisoning attacks and communication-efficient for participating clients in FL.
- **Chapter 5** proposes a new mechanism that can defend against both free riders and adversarial clients without imposing any additional cost on clients.
- **Chapter 6** concludes the thesis and outlines the future works.

Chapter 2

Backgrounds and Literature Review

This chapter provides a thorough background on blockchain technology and FL, encompassing blockchain consensus and topology, along with the objective, optimization, and vulnerabilities of FL. It then presents related work focusing on blockchain-based FL, robust FL approaches, and strategies for communication-efficient FL.

2.1 Blockchain technology

Since the inception of Bitcoin, the underlying blockchain technology has received considerable attention from both academia and industry for its distributed, transparent, and tampering-resistant features [95]. At its core, blockchain is a decentralized ledger or database that maintains a growing list of records, termed as blocks, which are linked using cryptographic hashes. As shown in Fig 2.1, the block is composed of block header and block body. Previous hash value, index, timestamp, difficulty value, nonce, and root hash constitute the block header, while the block body contains transactions. The root hash is the hash value of all transactions which guarantees that the transactions contained in the block cannot be tampered. The unique structure and the underlying consensus mechanism offered blockchain system decentralization, traceability, anonymity, and immutability.

- **Decentralization:** The decentralized peer-to-peer blockchain system is achieved by the joint efforts of nodes in the network. Every node in the blockchain network maintains a copy of the chain locally and tries to get the right to add a new block to the blockchain to win the reward.
- **Traceability:** The blockchain maintained and stored in every blockchain node provides traceable historical transactions as every historical block is kept with the chain.

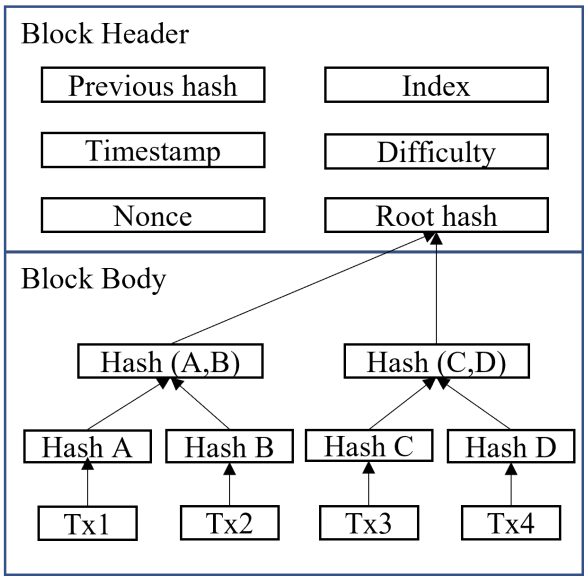

Fig. 2.1 The structure of a typical block

- **Anonymity:** Nodes in the blockchain system have their own private key and public key that conceal their identity in the real world.
- **Immutability:** The consensus mechanism deployed in the blockchain system makes the transactions hard to tamper, as the majority of nodes in the system are independent and honest. Thus, the adversary has to pay a lot in a proof-of-stake (PoS) based consensus network or compromise half of the nodes in a proof-of-work (PoW) based consensus network.

Owing to these features, blockchain technology has been extended from public to private, permissioned blockchain (private blockchain and consortium blockchain) emerged. The private blockchain has access control and is usually managed by a single authority or organization, while the consortium blockchain is controlled by multiple predefined authorities and organizations.

2.1.1 Consensus protocol

Consensus protocol, as the defining technology behind the security and performance of blockchain, has attracted increasing interest from both academia and industry. PoW, the underlying consensus of bitcoin, raised the following concerns: 1) unsustainable energy consumption, 2) low transaction capacity and long confirmation latency. In response to the above limitations, blockchain researchers have been investigating new blockchain protocols such as PoS, Proof-of-Authority (PoA), Casper FFG [18], Hyperledger Fabric [7], Algorand

[44], and Tendermint [17] etc. All blockchain consensus can be generally treated as PoW-based and BFT-based. PoW-based protocols scale well with network size and are suited for permissionless blockchains, but yield very limited throughput and long confirmation latency while Byzantine Fault Tolerant (BFT) protocols have built-in consensus finality and achieve much higher transaction capacity, but incur high messaging complexity per block ($O(N^2)$ versus PoW's $O(N)$, N is network size) and need a permissioned network for identity management, thus they do not scale with network size [131].

- **PoW-based protocols:** PoW, as the security guarantee of bitcoin, has proved its capability and capacity in past years. Without changing the way that miners mine a block, protocols like the greedy heaviest-observed subtree (GHOST) [113] and bitcoin-NG have been proposed [36]. The longest-chain rule is the rule adopted in the bitcoin network. Every node (i.e., miner) in the network always admits the longest chain. According to this rule, all unconfirmed blocks due to the fork would be orphaned, resulting in a waste of honest mining power which could otherwise contribute to the efficiency and security of the chain. GHOST is an alternative to the longest-chain rule, it allows orphan blocks to be accepted, thus more transactions in the same time interval would appear in the network, which effectively saves the honest mining effort. GHOST requires that given a tree of blocks with the genesis block being the root, the longest chain within the heaviest subtree shall be used as the main chain. Inspired by this, bitcoin-NG was proposed. It decouples block generation into two planes: leader-election and transaction serialization, which respectively correspond to two types of blocks: key blocks and micro blocks. Once a key block is mined, all subsequent microblocks shall be generated by the current key block miner until the generation of the next key block. The longest-chain rule is still applied to key blocks, while the heaviest-chain rule is adopted to microblocks.

- **BFT-based protocols:** Instead of calculating a meaningless hash of the block to meet the specific target, researchers have been investigating other energy-efficient ways to generate blocks. PoS is the one commonly adopted to partly or entirely replace PoW for block proposing. A stake refers to the coins or network tokens owned by a participant that can be invested in the blockchain consensus process. From the security point of view, PoS leverages token ownership for Sybil attack mitigation. Compared to a PoW miner whose chance to propose a block is proportional to its brute-force computation power, the chance to propose a block for a PoS miner is proportional to its stake value. Along with PoS, BFT is usually utilized for block finalization to achieve the security guarantee of the chain. Tendermint, Algorand, Casper FFG, and EOSIO [2] are well-adopted cases. Tendermint is the first public blockchain

project to incorporate a BFT consensus layer. It works in consensus cycles, each cycle involves a multi-round BFT consensus process to finalize one block [131]. Each round consists of three phases: Propose, Prevote, and Precommit. In the proposed phase, a validator is designated by a deterministic algorithm as block proposer in a round-robin fashion such that validators are chosen with a frequency proportional to the value of their deposited stakes (tockens). A validator continuously runs the three phases until more than 2/3 Precommits are received in one block. The validator would then broadcast Commit votes for the block and listen for other validators' Commit votes. When a block receives more than 2/3 of Commit votes, it will be finalized in the blockchain. Unlike Tendermint, the validator in Algorand is selected using MPC, a special verifiable random function (VRF) scheme has been applied, ensuring that only the stakeholder himself/herself knows if him/her gets elected into the committee. Casper FFG is a lightweight PoS consensus layer built on top of Ethereum's current PoW-based block proposing mechanism (Ethash). EOSIO is an example of delegate PoS (DPoS) protocol, it is the democratic form of Algorand without VRF, with the top 21 joining the consensus group, among whom the right of block proposal is equally shared. Although PoS is heralded as the most promising mechanism to replace PoW, it still owns vulnerabilities for its costless simulation and centralization risk.

2.1.2 Blockchain topology

In order to make blockchain more deployable in real world, especially for resource-constrained devices, researchers proposed a new structure of blockchain – DAG, which enables multiple blocks to be appended in the network concurrently, thus achieving a balance between asynchronization and speed. There are two types of DAG ledger, block-based DAG and transaction-based DAG. In a block-based DAG, every vertex contains a collection of transactions which is similar to the block concept in bitcoin, the only difference is that one block can be hash-pointed to multiple parent blocks. SPECTRE [112] is one of the block-based DAG protocols, requiring any node who wants to mine a new block to find all blocks of zero in-degree (i.e., "tips") in the DAG and hash-point the new block header to these tips before starting the PoW mining for the new block. A recursive voting procedure is employed to determine the order of any two blocks in concurrent DAG, but may not extend to fully linear ordering. To address this issue, PHANTOM [114] was proposed. Every node in PHANTOM searches for the largest k-cluster of blocks. k denotes the node degree in the cluster and is a predefined security number that rarely k or more honest blocks are created simultaneously. The k-cluster is regarded as honest and all blocks within are linearly ordered. The transactions covered by the cluster are then validated in the new order. Differentiating from the two,

Conflux [71] incorporates a specific blockchain as a main chain, reflecting a similar idea as the GHOST rule. It introduces the parent edge and reference edge and maintains the pivot chain to keep the full order of the blockchain. This scheme yields high transaction throughput but also higher confirmation latency [131]. Though the transaction capacity of block-based DAG is much higher than that of traditional chain-structured blockchain, it can still hold overlapped transactions and takes extra bandwidth and processing power to solve conflicts.

Unlike block-based DAG, every transaction in transaction-based DAG forms a vertex and is hash-pointed to previous transactions. IOTA Tangle is one of the transaction-based DAG schemes, which applies a two-tip rule to add transactions to the chain. That is every user needs to validate two unapproved transactions (tips) of the DAG to append a new transaction. Markov-chain Monte Carlo is employed to solve the conflicting tips. Though Tangle is a transaction-based DAG, its transaction capacity is still capped by link-/physical-layer communication bandwidth. Differentiate with Tangle, an arbitrary tip number is set in Byteball [29]. A witness group scheme (12 reputable entities) is deployed for determining the MC and transaction finalization. The TPS of Byteball increased but with decentralization compromised. Nano [69] is another transaction-based DAG proposal. It is built upon a parallel blockchain with a block-lattice structure. The key insight of Nano is that forks and other faulty transactions only affect the accounts referenced in the said transactions, which allows Nano to provide stable and fast transaction confirmation.

2.2 Federated learning objective and optimization

FL is a privacy-preserving ML training paradigm. In FL tasks, the goal is to learn a single model that minimizes the empirical risk function over a union of clients' training data that is kept locally within their devices. Consider a typical FL setting with n clients and a CS, the objective function $F(w)$ (w is the shared model) could be written as:

$$F(w) = \mathbb{E}_{D\sim\mathscr{X}} f(D,w) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}_{\xi_i\sim D_i}[F(\xi_i,w)] \tag{2.1}$$

where ξ_i is the training data sampled from its local training dataset D_i, $D = \bigcup_1^n D_i$, $\mathscr{X}$ is the training data distribution, and f is the loss function that clients use to train w.

To learn the optimal model, various optimization approaches have been proposed. McMahan *et. al.* [88] introduced Federated Averaging (FedAvg) which is the most popular one and has been widely deployed. As shown in algorithm 1, in each communication round, selected clients perform a number of Stochastic Gradient Descent (SGD) steps locally and send updated local models to the CS. Through averaging the received model updates, the CS

creates the global model for the next round of training. Typically, FedAvg is executed over a predefined number of communication rounds or a predetermined time allocation [15]. The convergence of FedAvg over independent and identically distributed (i.i.d) [123, 144, 62] and non-i.i.d [76, 77] data have been theoretically analyzed in the literature. While FedAvg has shown empirical success, it does not fully address the underlying challenges associated with heterogeneity that arises from data across different devices and the computation and communication capabilities of participating devices [75]. To tackle the heterogeneity issue and accelerate convergence, many FedAvg variants have been proposed. For example, Li *et. al.*[75] introduced FedProx which adds a regularization term to client's local objective function, restricting the local update to be closer to the global model. Sai Praneeth *et. al.* [60] devised SCAFFOLD which reduces the variance between clients to reduce client drift and accelerate convergence. Adaptive optimization methods in traditional centralized ML, such as SGD with Momentum, Adam, and AdaGrad, have been adapted to FL for faster learning [134, 91, 102]. The optimization of FL remains an active area of research, seeking to enhance the effectiveness and applicability of FL in diverse real-world scenarios.

Algorithm 1 FedAvg

Sever executes:
Initialize w_0
for each round $t = 1, 2, \ldots, T$ **do**
  $S_t \leftarrow$ randomly select client set
  for each client $i \in S_t$ **in parallel do**
    $w_{t+1}^i \leftarrow ClientUpdate$
  end for
  $w_{t+1} \leftarrow \sum_i^{|S_t|} \frac{1}{|S_t|} w_{t+1}^i$
end for

ClientUpdate(i, w):
for local step $j = 1, 2, \ldots, K$ **do**
  $w \leftarrow w - \eta \nabla f(\xi_i, w)$ for xi_i D_i
end for
return w

2.3 Vulnerabilities of federated learning

The privacy-preserving principle and the centralized orchestration inherent in FL render it susceptible to the threats posed by unreliable clients and potential malfunction or misbehavior

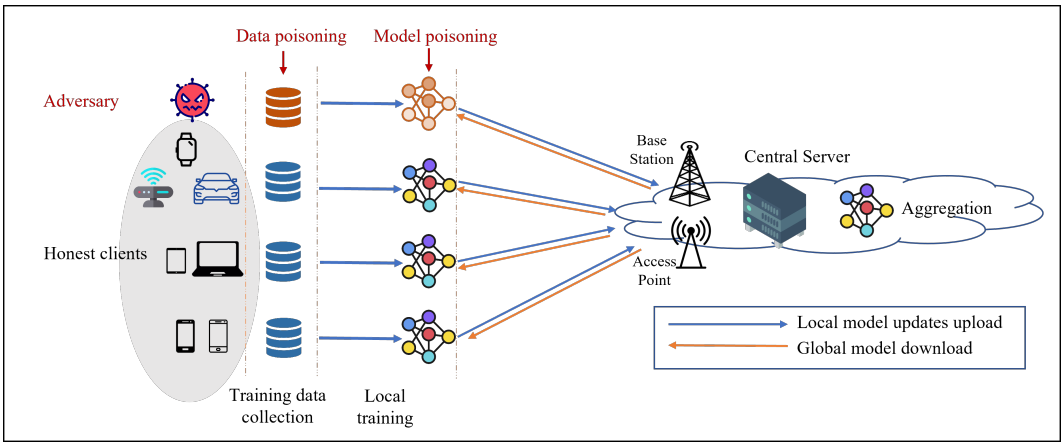

Fig. 2.2 Poisoning attacks in FL

of the central server. These vulnerabilities give rise to a diverse array of attack vectors and security concerns.

2.3.1 Threats from clients

FL is particularly vulnerable to byzantine clients, named after the Byzantine Generals Problem in distributed system, referring to clients who don't act in the system's best interest. Recent study shows that even a single byzantine client can degrade the global model significantly [40]. Within the context of FL, such clients can exploit the system's trust in a multitude of ways:

- **Poisoning attacks:** These attacks involve byzantine clients sending manipulated or falsified model updates to the server. By doing so, these clients aim to adversely affect the aggregated global model. Such detrimental updates can skew the model's learning trajectory, leading to suboptimal or entirely incorrect predictions. Poisoning attacks could be roughly classified as data poisoning and model poisoning, as shown in Fig 2.2.

  - Data poisoning attack: Data poisoning happens during the training data collection phase. Adversaries can inject poisonous data samples into the training dataset. The ways of generating poisonous data could roughly be classified as feature-altering and label-altering. In feature-altering, the adversary is assumed to have no control over the labeling process but can craft the specific feature of training examples. While in label-altering, adversaries replace the targeted label with the desired one without touching the training examples. One common data poisoning attack in FL scenario is *label-flipping* [40]. During a label-flipping

attack, an adversary deliberately alters the labels of their local data prior to training. For instance, in an image classification task, to force the global model to misclassify images labeled "1" as "7", the adversary would flip the labels from "1" to "7" in their training data. Data poisoning attack could be carried out by any participating clients in a FL task, the strength of the attack depends on the number of adversaries in a training round and the proportion of the poisoned data samples.

- Model poisoning attack: Instead of compromising the integrity of training data collection, model poisoning attacks compromise the integrity of the learning process during the local training phase and are considered to be stronger than the data poisoning attacks for their direct influence on the shared global model. As the CS is unable to inspect the individual training, an adversary can behave completely arbitrarily and tailor its outputs to have a similar distribution with the correct model updates. Typical model poisoning attacks include: 1) *Sign-flipping* [73] A sign-flipping attack involves the inversion of the sign of a local model update to disrupt the convergence of the global model. Specifically, an adversarial client i trains his/her local model using the received global model and computes the local model update ∇g_i. Rather than submitting ∇g_i to the central server, the client uploads a manipulated version $u\nabla g_i$, where u is a negative constant that inverses the gradient direction. When $u = -1$, this attack is also known as *bit-flipping* attack. 2) *A Little is Enough (ALIE)* [11]: ALIE represents a time-coupled attack capable of deceiving robust aggregation rules. In such an attack, the adversary subtly adds small perturbations to the average of model updates from benign clients in a single training round. These minute perturbations accumulate over time, leading to model divergence. Specifically, the adversary calculates the benign model updates' average μ and standard deviation δ, and then constructs the corrupted model update as $\nabla g' = \mu - z\delta$. Here, z is a coefficient determined by the number of participating clients and adversaries. 3) *Gaussian attack* [13]: In this attack, adversaries send the random model updates following the Gaussian distribution to the CS. The larger the isotropic covariance of the Gaussian distribution, the stronger the attack.

- **Free-riding attack:** A free-riding attack is explicitly conceived to stay undetected while not disturbing the FL process. Fraboni *et. al.* [39] proposed plain free-riding and disguised free-riding attacks with theoretical analysis of not compromising the convergence of the global model. In plain free-riding attack, the free rider simply returns the zero gradient, which is easy to detect. While the disguised free-riding attack

adds small additive noise to mimic SGD updates. Specifically, the free rider submits a local model update $g' = \varphi(t)\varepsilon_t$ to the central server, where ε_t is Gaussian white noise, and $\varphi(t)$ is the time-varying perturbation coefficient approximated as $\varphi(t) = \sigma t^{-\gamma}$. Here, t represents the current training round. The parameter σ corresponds to practical assumptions on the parameter evolution during federated learning, and can be estimated using the update distribution $\Delta w_g = w_g^1 - w_g^0$ after random initialization of the global model. The parameter γ serves as the decay factor. This work assumes free riders to conduct disguised free-riding attacks.

2.3.2 Threats from central server

Being the orchestrator of the entire process, CS aggregates model updates from clients, ensuring that the global model benefits from the collective intelligence of all participants. While FL preservers privacy by moving model training back to where data is generated, the CS in FL systems still plays a pivotal role, making it a potential target and a source of vulnerabilities.

One of the primary threats associated with the CS is the risk of a single point of failure [79, 125]. Since the central server is responsible for aggregating model updates from multiple clients and disseminating the updated global model back to them, any compromise in its integrity or availability can have widespread repercussions. A malicious or compromised central server could manipulate the aggregation process, leading to the distribution of a corrupted global model. This could be particularly damaging in scenarios where the model influences critical decisions, such as in healthcare or autonomous driving applications.

The central server also poses a risk in terms of privacy [98, 14]. Even though raw data does not leave the client devices, sophisticated analysis of the aggregated updates by a compromised server could potentially lead to privacy breaches. For instance, model inversion techniques could be employed to infer sensitive information about the clients' data.

In summary, while FL offers numerous advantages in terms of decentralized learning and privacy preservation, it also brings forth unique challenges that need to be addressed. The threats posed by byzantine clients and the vulnerabilities associated with a centralized CS emphasize the need for robust security measures to make FL truly resilient and trustworthy.

2.4 Related work

This subsection will introduce the related work in blockchain-based FL and the robust mechanisms designed to defend against byzantine clients in FL.

2.4.1 Blockchain-based federated learning

To handle the issue brought about by an unreliable server, blockchain-based FL becomes a promising solution. The peer-to-peer distributed nature of blockchain makes the decentralized aggregation of global model available, shedding new light on handling the single point of failure in conventional FL. Research works [109, 78, 145, 80, 83, 24, 101, 38, 147, 128, 37] move the aggregation step from the server to the blockchain nodes, while Qu *et al.*[100] and Mugunthan *et al.*[93] eliminate the role of the aggregator by letting clients themselves aggregate model updates obtained from the blockchain network. To ensure the reliability of the global model, most blockchained FL frameworks embed local model update verification into their design. Under different assumptions on blockchain nodes, the verification methods vary. Several blockchain-integrated FL frameworks assume that clients join the blockchain network and participate in FL task simultaneously, so that their training data could be used as the validation datasets [78, 83, 24], thus the validation accuracy of local model updates (recorded in the transaction) could be further utilized in the aggregation process. Li *et al.* [78] introduced K-fold cross-validation where K committee members test the model update on their training data and take the median of the accuracy values as the score of the update. Model updates with qualified scores would be selected for global model aggregation. Lu *et al.*[83] applied a similar strategy to select model updates with accuracy lying within a certain range to update the global model. This accuracy-based validation mechanism has also been utilized in blockchain-based FL systems where nodes have no access to the training data. Mugunthan *et al.*[93] implemented smart contract with a cross-verification procedure that clients cross-verify others' model updates by testing them over their local training datasets, and send the accuracy values back for the calculation of contribution scores, which would be used in the weighted aggregation stage.

Instead of evaluating model updates with the help of clients, which would inevitably increase the communication burden and be restricted with clients' active status, recent blockchain-based FL frameworks reaped recent advances in Robust FL to protect against malicious clients. Biscotti [109], SPDL [137] and Omnilytics [80] applied the Multi-krum as the validation mechanism and give passes to model updates with closer Euclidean distances. Biscotti introduces noiser, verifier and aggregator nodes, where noisers produce differentially private (DP) Gaussian noise to be added to the model updates, verifiers run Multi-krum to sign commitments for passed updates, and aggregators aggregate unmasked passed model updates via a secure protocol. Similar to Biscotti, SPDL leverages DP Gaussian noise to local model update and Multi-krum in model updates aggregation to provide private and secure FL with theoretical convergence guarantee. Omnilytics was implemented using smart contract with incentives and punishments to honest and malicious clients respectively. However,

the proposed designs have only been tested using i.i.d datasets, so their performance under non-i.i.d FL setting has not been explored.

The deployments of proposed blockchain-based FL systems in the research community roughly fall into three categories: to client devices [109, 78, 24, 101], to edge nodes [145, 57, 48], and to the existing blockchain platform Ethereum with smart contracts [80, 93]. Considering the limited resources of client devices, running and maintaining blockchain locally is costly in terms of communication and storage for the large propagated block in the network and the ever-growing blockchain. For PoW-based blockchain systems [101], the additional computation cost of running the blockchain is also non-negligible. The existing mature platform, Ethereum, that enables smart applications running on the blockchain provides secure execution of FL tasks but is monetary costly in terms of the gas fee induced by every interaction with the blockchain. The deployment to edge nodes is promising for their broad capacity in terms of computation, communication and storage, but existing blockchain-based FL works targeting this scenario fail to address the security issue brought by malicious clients and an unreliable server, and the high communication overhead for resource-constrained clients concurrently.

2.4.2 Robust federated learning

In FL, the FedAvg algorithm [88] is widely used for federated model training which takes either the model average or the gradient average of model updates from clients for its compelling performance under honest settings [65]. However, this mean aggregation rule is not robust in adversarial situations. Specifically, an attacker can arbitrarily manipulate the parameters of the shared global model since this simplified average operation with only one worker device being compromised [13, 142]. To mitigate the adverse impact of byzantine clients in FL, various research endeavors have been devoted to byzantine-robust FL.

However, most studies focus on the adversarial setting of FL where adversaries launch poisoning attacks. A subset of these works aims to mitigate the impact of poisoning attacks by comparing each client's local model update and discarding statistical outliers before incorporating them into the global model update [13, 142, 46, 99, 41]. For example, Blanchard *et al.* [13] proposed the Multi-Krum, a Euclidean-distance-based aggregation rule, that is resilient to f Byzantine workers. Following the Krum, the CS calculates the sum of Euclidean distance of every model update to others as scores and chooses the smallest $n-f-2$ ones for final averaging aggregation, where n is the total number of local model updates in a single communication round. Theoretical convergence of the global model has been approved when $f < \frac{n-2}{2}$. Differentiate from Krum, TM, and CM [142] perform element-wise aggregation. In TM, for each i^{th} model parameter, CS sorts the i^{th} parameter of

n local model updates (i.e., $g_{1,i}, g_{2,i}, g_{3,i}, \ldots, g_{n,i}$) and removes the largest and the smallest β of them before computing the average value of the remaining $n-2\beta$ ones. This aggregation rule achieves an order-optimal error rate (the optimal growth rate of the distance between the calculated solution and the theoretical optimal solution) when $f \leq \beta < \frac{n}{2}$ and the objective function is strongly convex. Instead of calculating the trimmed mean of the model updates, CM takes the median of the i^{th} model parameter of all local model updates as the relative i^{th} gradient for updating the global model. Similar to the CM, Pillutla *et al.* [99] applied a "middle-seeking" strategy and proposed the RFA by approximating the geometric median of local model updates vectors for global model update. However, additional twice communication overhead is required to solve the approximation of the geometric median. Another type of works seeks to identify adversarial clients in poisoning attacks [74, 146]. For example, FLDetector [146] uses Cauchy mean value theorem to predict clients' model updates and check their consistency across rounds for the detection of malicious clients. Another line of work aims to provide certifiable robustness under specific constraints, such as when the number of adversarial clients does not exceed a certain threshold [21], or when the change in training data from malicious clients is bounded [132].

Research on defense against free riders in FL is still in a nascent stage. A limited number of works have been done[81, 52, 139, 121]. Lin *et al.* [81] and Huang *et al.* [52] utilize DAGMM for free riding attack detection based on the statistics of evolving model updates. However, they may not be effective, as there is no further punishment to free riders, enabling their continuous access to the global model throughout the learning process. Wang *et al.* [121] move further towards building the defense mechanism, based on contribution scores. It calculates the contribution score of each client with the help from all other participating clients through cross-validating model updates. Nevertheless, it induces extra computational and communication overhead for clients, and its functionality depends on clients' active participation. Despite these efforts. few works can defend against both the poisoning and free-riding attacks simultaneously due to their distinctive attacking patterns. Recently, Xu *et al.* [139] proposed RFFL with reputation scores to build the defense, which are calculated based on the cosine similarity between client's local model update and the global model update. However, it has a restricted scenario of full client participation and is sensitive to the diversity of the local model updates.

As FL continues to evolve, ensuring the security, authenticity, and reliability of the collaborative training process is of utmost importance. Defense mechanisms in FL are not just ancillary tools but are foundational to ensuring the viability and trustworthiness of FL in real-world applications.

2.4.3 Communication-efficient federated learning

One of the major bottlenecks hindering the real-world FL application is the high communication cost to clients that arises from sending around model updates. The participating clients are often edge devices with limited bandwidth. Addressing this challenge is crucial to harness the valuable user data for FL, enabling the realization of privacy-preserving collective intelligence across diverse applications. Previous works targeting communication-efficient FL can be broadly categorized into two approaches: 1) reducing the communication overhead for clients in stable FL environments and 2) optimizing resources for FL deployment in practical, unreliable environments.

To reduce the communication overhead, various gradient compression methods have been proposed and shown significant reductions in communication cost with minimal impact on training accuracy [64, 5, 104, 4, 118, 107]. This compression can take various forms, such as quantization, where the model weights are represented using fewer bits, effectively reducing the size of the updates with minimal impact on the model performance. Another strategy involves sparsification of the updates, where only a subset of the model parameters, typically those with significant changes, are transmitted [65]. This selective transmission ensures that only the most impactful updates are communicated, thereby conservatively using the available bandwidth. Another line of work focuses on reducing the number of communication rounds [91, 129, 26, 23]. Mills *et al.* [91] proposed to use a distributed form of Adam optimization to accelerate the convergence speed and reduce the number of communication rounds. Wu *et al.* [129] used client selection where adverse local model updates are excluded in the aggregation stage to propel the convergence of the global model. Chen *et al.* incorporated meta-learning with MAML and META-SGD to speed up the model training. Apart from these methods, Chen *el al.* proposed Lazily Aggregated Gradient (LAG), where the reuse of outdated gradients is applied, thus reducing the communication rounds to achieve a target accuracy.

To facilitate the practical deployment of FL in edge environments, where clients are often wirelessly connected to the central server and the network conditions are dynamic and unreliable, various approaches have been proposed. One line of work focuses on mitigating the impact of noise in the model update transmissions, aiming to reduce the number of communication rounds required to achieve the desired model performance. For example, Yang *et. al.* [141] proposed a noise-aware scheduling policy to prioritize model updates with better channel conditions. Ang *et. al.* [8] proposed an alternative approach to tackle noise by adjusting the loss function of each client through the addition of a regularization term. Some other works introduce efficient client scheduling strategies. Amiri *et. al.* [6] investigated the trade-off between the number of participating clients in each round and

the allocated resources per client, and proposed a client scheduling approach that jointly considers the channel condition and the importance of local model update. However, this work assumes perfect channel state information, which may not hold true in real-world deployments. To address this, Wadu *et. al.*[119] developed a client scheduling approach according to the predicted channel conditions by utilizing Gaussian process regression under imperfect channel state information.

In summary, these strategies enhance the feasibility of FL, particularly in bandwidth-constrained environments, and pave the way for its broader adoption across various domains where data privacy is paramount.

2.5 Summary

This chapter presents an overview of the background knowledge related to this thesis. It begins with an introduction to blockchain technology and FL with a focus on its objective, optimization, and vulnerabilities. It then presents state-of-the-art research work towards secure FL using blockchain, robust FL with defense mechanisms, and communication-efficient FL with different strategies.

Chapter 3

Lightweight Blockchain-based Federated Learning Framework

The fusion of blockchain and FL appears a promising solution to realize the reliability and integrity of the shared global model in the collaborative learning process. There have been a number of research works dedicated to blockchain-based FL. However, as described in Section 2.4, they are either costly in terms of communication and storage due to the large block size (within the ever-growing blockchain) or costly to operate, as every interaction with the blockchain (i.e., Ethereum) consumes a monetary gas fee. Thus, there is a lack of a cost-effective and reliable blockchain-based FL system that runs at vulnerable network edge with resource-constrained devices. To fill this gap, this chapter proposes a novel lightweight blockchain-based FL (LBFL) framework, tailored for FL applications in edge computing enviroment. Incorporating Inter-Planetary File System (IPFS) and Verifiable Random Function (VRF), LBFL is scalable to a large number of blockchain nodes, storage and communication-efficient at the network edge.

3.1 Prelimilary

3.1.1 IPFS

Inter-Planetary File System is a peer-to-peer distributed file system that provides a high throughput content-addressed block storage model with content-addressed hyperlinks (the unique hash values) [12]. Its Distributed Hash Table (DHT) and Merkle Directed Acyclic Graph (DAG) make efficient data storage and file retrieval. Any modification of a file would result in a different hash value, thus the integrity of the stored file is ensured. Although the alternative peer-to-peer file system exists (*i.e.,* Maidsafe [97] which is also a decentralized

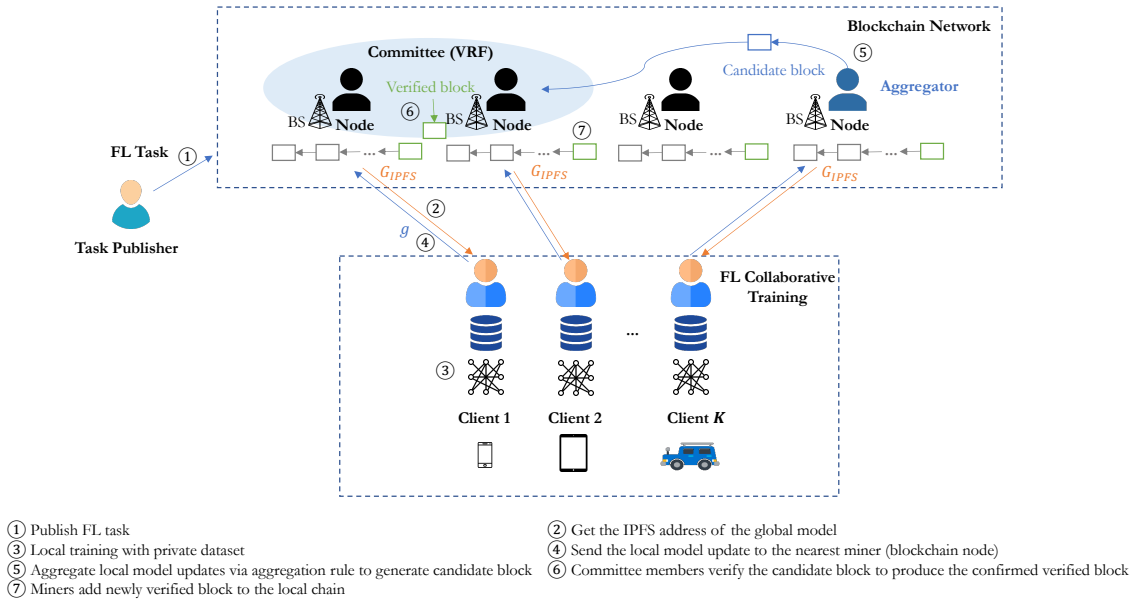

Fig. 3.1 Overview of LBFL framework

and autonomous network for secure data storage and retrieval with content-address hash values stored in the DHT), IPFS has been widely applied in blockchain system development [148, 58, 63, 66]. In this regard, LBFL incorporates IPFS to store the global model on the blockchain nodes, thereby reducing the storage cost of the blockchain and ensuring the integrity of the global model.

3.1.2 VRF

Verifiable Random Function [90] is a public-key pseudorandom function that provides proof for its random outputs with certain inputs. VRF provides blockchain nodes a non-interactive way to independently determine if they were chosen to be the committee members [44]. These unique characteristics of VRF allow us to use it to allocate the blockchain committee in LBFL.

3.2 LBFL Design

As shown in Fig. 3.1, LBFL is assumed to be deployed and run at the network edge with edge nodes (*i.e.,* Base Stations (BSs)) to be the blockchain nodes. The task publisher first publishes the FL task with the initial global model stored in the IPFS to the blockchain network (Step ①). The blockchain nodes retrieve the task information from the network. Clients then could get the IPFS address of the global model via querying the nearest active node (Step

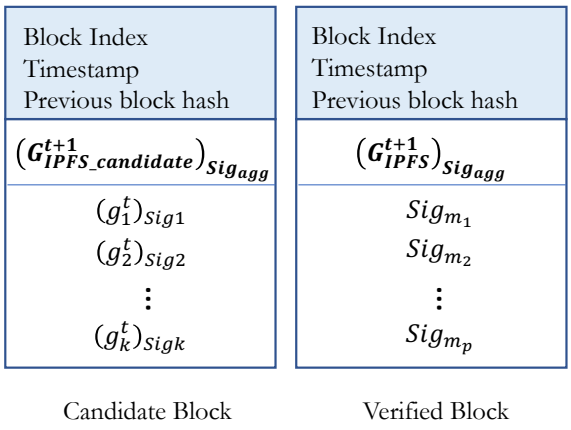

Fig. 3.2 Block structure

②). Utilising the content-addressed IPFS, the global model could be easily downloaded. Starting with this global model, clients run local training steps using their private datasets. Once the local training is complete, clients send the model updates (gradients) together with their signatures as transactions to the nearest active node (Step ③ and ④). Receiving a model update from a client, the blockchain node first verifies whether it is issued by the client via checking the signature. If the requirement is met, the node disseminates this transaction to the network. Once enough pending transactions (local model updates) are received, nodes competitively calculate the global model for the next round following the aggregation protocol specified, generate the candidate block, and send it to the committee that was allocated with VRF (Step ⑤). Committee members then verify if the global model is aggregated correctly with their "Yes" or "No" votes. If the candidate block receives more than $\frac{2}{3}$ agreements from the committee, it is regarded as a verified block and propagated to the network. After receiving the newly verified block, nodes check the signatures of the committee members and add it to the local chain (Step ⑥).

3.2.1 Transaction and Block Design

A blockchain is a list of blocks linked with hash pointers. Each block has a block header, consisting of the index of the block, timestamp, and the hash string of the previous block, and a block body that contains transactions. As shown in Fig. 3.2, each model update with the signature signed by the client forms the transaction. Model updates utilized for global model aggregation together with the IPFS address of the aggregated model W_g^{t+1} signed by the aggregator would be packed into the body of the candidate block. Once the candidate block has been confirmed, local model updates contained in this block will be removed for their uselessness to future rounds of the learning process and the benefit of storage saving. Instead,

signatures of the block header and IPFS address of the confirmed global model issued by committee members who vote "Yes" to the candidate block would be recorded.

3.2.2 Consensus Design

Stake-based Committee Constitution

Using VRF, LBFL selects committee members in a private and non-interactive manner. Based on VRF, a novel sortition algorithm for choosing a random subset of nodes to constitute the committee is proposed. Each node runs the sortition algorithm independently with a public seed (the latest block of blockchain) as its input to see if itself is selected as a candidate committee member. It is non-trivial that the sortition selects nodes in proportion to their

Algorithm 2 Sortition

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

Output: $select, hash, proof$

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

owned stakes; otherwise, it would be vulnerable to Sybil attack [34], where a single faulty entity presents multiple entities, thus controlling a substantial fraction of the system. As shown in the *Sortition* algorithm, if the generated unique random *hash*, determined by the secret key SK_i and the input *seed*, satisfies the condition, the node i becomes a candidate committee member and sends the eligible information (*hash* and $proof$) to the network. The r generated via VRF is a unique uniform random value that lies within the range of $[0,1]$, thus its probability density function is $f(r) = 1$. The probability of blockchain node i being chosen to be a committee member could be computed as follows:

$$p_i = Pr(\frac{r}{s_i} < \frac{\alpha K}{S}) = Pr(r < \frac{\alpha K s_i}{S}) = \frac{\alpha K s_i}{S}, \tag{3.1}$$

it can be seen from the above equation that the more stake the node owns, the higher probability of it being chosen for the committee. The committee constitution phase ends when the correct size of the committee is achieved, first K committee members become the

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

$$E(X) = \sum_{i=1}^{n} p_i = \alpha K. \tag{3.2}$$

According to the Chernoff bound on the sum of independent Bernoulli trials, the following can be obtained

$$Pr(X \leq (1-\delta)\alpha K) \leq e^{-\alpha K \delta^2/2}, \tag{3.3}$$

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

$$Pr(X \leq K) \leq e^{-K(\alpha-1)^2/2\alpha}. \tag{3.4}$$

To ensure enough candidate committee members exist in each committee constitution phase, the probability equation (3.4), indicating not having $K+1$ candidate committee members should be small to 0. The choice of K and α will be discussed in the Evaluation section.

Consensus Achievement

As the defining technology behind the security of blockchain, the consensus protocol is of significant importance. The VRF-enabled committee constitution guarantees resistance against Sybil attacks as no blockchain node can predict the next generated block in advance, and there are limited stakes that malicious nodes could hold. As shown in Algorithm 3, when a new candidate block is distributed to the network, committee members that are in charge of verifying candidate blocks first check whether enough pending transactions are collected for global model aggregation and the signature of the IPFS address is issued by the aggregator using its public key pk_{agg}. If the aforementioned condition is met, committee members calculate the global model following the predefined aggregation protocol (provided by the task publisher) and compare it with the one obtained from the IPFS. If the results are the same as the aggregator's, committee members vote "Yes" to the candidate block with their signatures regarding the block header and the signed IPFS address. We use the Elliptic Curve Digital Signature Algorithm (ECDSA) to make the signature commitment which is utilized

Algorithm 3 Consensus

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

in Bitcoin. If more than $\frac{2}{3}$ agreements from committee members are received, the candidate block is regarded as the valid verified block with transactions replaced by signatures from supported committee members. This $\frac{2}{3}$ consensus threshold is based on the concept of the Byzantine Fault Tolerance (BFT) [68]. In a distributed system like a blockchain, some of the nodes (or validators) may be faulty and behave maliciously or erroneously, leading to inconsistencies in the system. The BFT consensus algorithm is designed to tolerate up to one-third of the nodes being faulty. Setting the consensus threshold to $\frac{2}{3}$ ensures that LBFL can tolerate up to one-third of the committee members being faulty (either due to malicious intent or technical issues), while still achieving consensus on the verified block. In LBFL, the voting process for each candidate block occurs only for a specified duration. Any candidate block that fails to receive enough agreements during the time duration will be dropped and the committee will verify the next candidate block. For the committee constituted each time, LBFL specifies its maximum voting step. If there is no candidate block being confirmed during these voting rounds, a new committee will be constituted to verify the new candidate block.

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

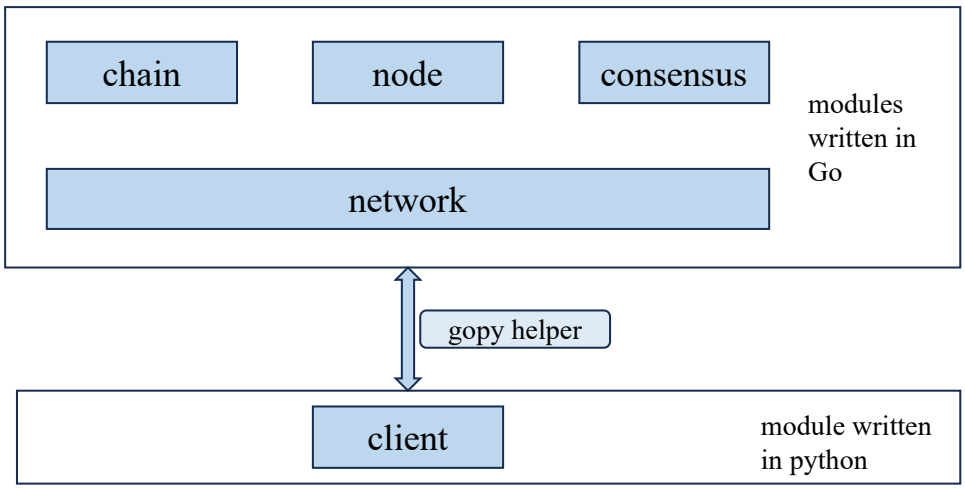

Fig. 3.3 Modules in LBFL

3.3 Implementation

The implementation of LBFL can be divided into two parts: blockchain and FL, consisting of chain, node, consensus, and network modules written in go, and client module written in python, as shown in Fig 3.3. The FL part (client) interfaces the blockchain part with a gopy helper package based on *go-python3* library[33]. Client operations, including global model retrieving and model training using local data, are implemented in the client module. *Go-ipfs-api* [53] was used to connect with the IPFS. The cryptography part of the design is implemented with *Coniks* [30] library which is used for VRF implementation and the built-in crypto package of Go that contains ECDSA implementation.

3.3.1 Chain module

The block and transaction structures are defined in this module, together with their calling functions including: block generation, block validation, block accessing and retrieving, transaction generation, transaction validation, and chain validation, *etc*.

3.3.2 Node module

The actions of a blockchain node are defined in this module, such as: voting if being selected as a committee member, getting the chain information from a peer, getting transactions in the transaction pool, sending transactions in the network, and sending the latest global model to the client who requested, *etc*.

3.3.3 Consensus module

The proposed VRF-based consensus mechanism is implemented in this module. Through calling the role function implemented in this module, a blockchain node is able to know whether it becomes a committee member in a new block generation round. Once a candidate block is published in the network, the voting function will be triggered and active committee members will vote for the candidate block upon receiving the notification.

3.3.4 Network module

This module enables the communication between nodes over TCP layer. A super node is introduced to mimic the broadcasting channel. Through implementing remote procedure calls, FL task publisher is able to publish the FL task and clients can interact with the blockchain network.

3.4 Evaluation

LBFL was deployed to a machine with one NVIDIA Getforce GTX 1080 Ti GPU, one Intel (R) Xeon (R) CPU E5-2630 v3 @ 2.40GHz and 64GB of RAM. All experiments were executed over non-i.i.d datasets: FEMNIST and CIFAR10. FEMNIST was preprocessed using the Leaf federated benchmark tool [19] consisting of 62 different classes (numbers and letters). The non-i.i.d CIFAR10 partition constructed from LotteryFL [70] was used, where clients can have different classes of unbalanced data with different degrees. This evaluation assigned each CIFAR10 client 10 classes of data with an unbalanced degree of 0.75. All FL tasks were loaded with 50 workers and 20 of them were selected randomly to participate in each FL training round. The parameter values of LBFL are presented in Table 1 unless stated otherwise. To ensure enough candidate committee members exist in each committee constitution phase, this evaluation set committee size K and control parameter α with 15 and 3 respectively, so that the probability of not having $K+1$ candidate committee members is smaller than 4.54×10^{-5} according to the equation (3.4), indicating it would barely happen. This evaluation used the secure aggregation rule proposed in Chapter 4 as the aggregation protocol. A custom CNN was implemented for the FMNIST task, which comprises a convolutional layer with relu, a max-pooling layer, a convolutional layer with relu, a max-pooling layer, a fully connected layer with relu, and the output layer with softmax. For the CIFAR10 task, this evaluation used ResNet14[49]. Each experiment was repeated 3 times.

Table 3.1 The default parameters of LBFL

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

3.4.1 Security Analysis

The safety of LBFL is guarded by the committee-based consensus protocol, but proving this experimentally requires testing all possible attack strategies which is infeasible. This subsection measures the resistance of LBFL to byzantine blockchain nodes with the specified attacking strategy: the byzantine blockchain nodes actively participate in both committee constitution and candidate block generation. If the byzantine blockchain node gets the chance to be a committee member, it votes "No" to the correct candidate block and votes "Yes" to the incorrect one. For generating the candidate block, instead of aggregating local model updates following the predefined aggregation protocol, byzantine blockchain nodes perform a Gaussian attack by setting random values following the standard Gaussian distribution as global model parameters. $\frac{1}{3}$ blockchain nodes are assigned to be malicious and colluded together to mislead the learning process, which is 33 out of 100. This is the maximum number of byzantine faulty nodes in the distributed system to make sure all non-faulty nodes can reach a consensus [68]. As shown in Fig. 3.4 and 3.5, the performance of BEFL was unharmed by these byzantine nodes, demonstrating that BEFL is robust to this kind of attack scenario.

3.4.2 Scalability Analysis

To measure the scalability of LBFL, this evaluation varied committee size with a fixed 100 blockchain nodes, and varied the number of blockchain nodes with a fixed committee size

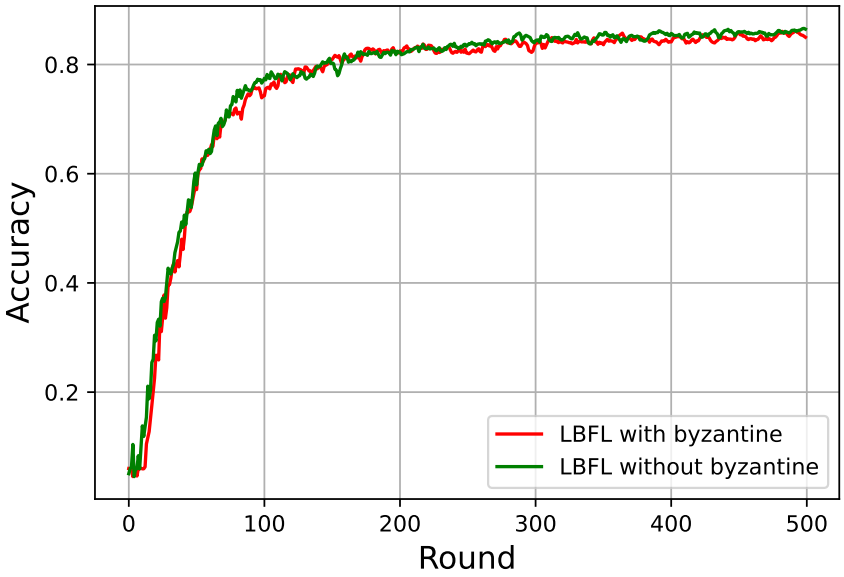

Fig. 3.4 Test accuracy of LBFL under honest clients setting for FEMNIST task

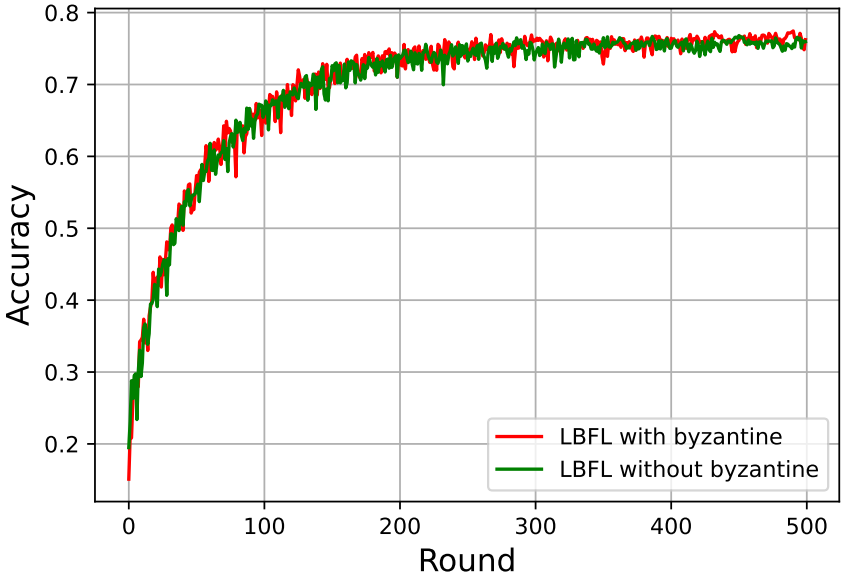

Fig. 3.5 Test accuracy of LBFL under honest clients setting for CIFAR10 task

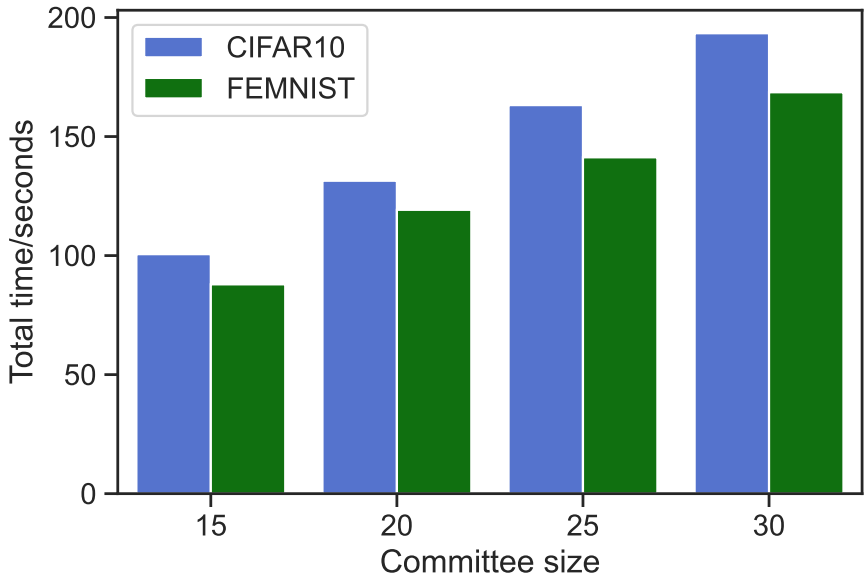

Fig. 3.6 The averaged total time of per verified block generation and distribution with varied committee sizes for CIFAR10 and FEMNIST tasks

of 15. The CIFAR10 and FEMNIST tasks were rerun under different LBFL system settings for 50 training rounds. As shown in Fig. 3.6, the average total time per verified block generation and distribution grows almost linearly with the committee size as the dominant voting time increases. The overall time for generating and distributing a valid block for the FEMNIST task is shorter than that of the CIFAR10 as the smaller transmitted model updates in the network. Although the larger the committee size, the more certainty of having enough candidate committee members in each committee constitution phase, and the higher credibility of the generated block as more agreements obtained, the additional communication overhead for reaching consensus on candidate block is non-negligible. Fig. 3.7 shows the performance of LBFL as the network size grows. It can be seen that the total time of block finalization and distribution barely increases in CIFAR10 task and the increment is almost negligible in FEMNIST task when the number of blockchain nodes increases from 50 to 800. Instead of saving the full shared global model and local model updates in a block, the recorded IPFS address and the removal of unnecessary stale model updates reduces the block size significantly, thus enabling fast block transmission in the network.

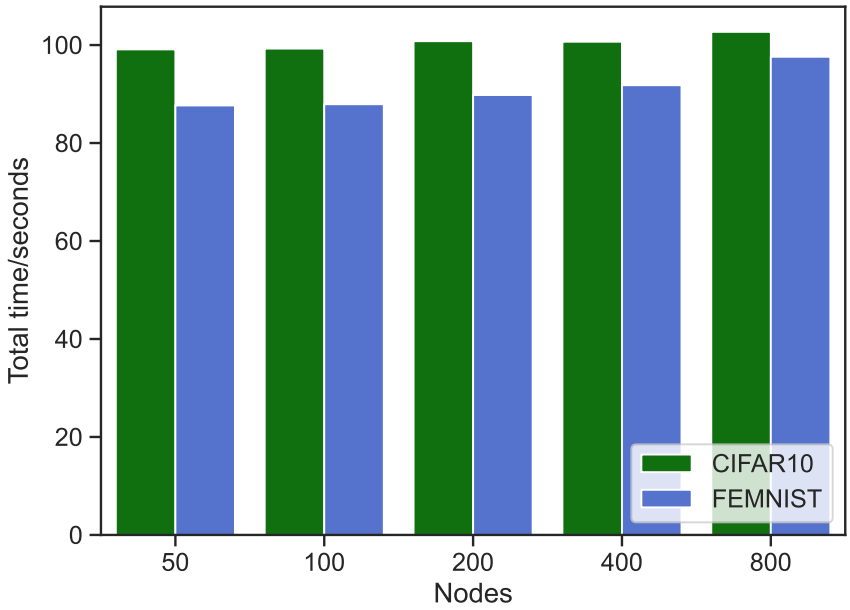

Fig. 3.7 The averaged total time of per verified block generation and distribution with varied blockchain network size for CIFAR10 and FEMNIST tasks

3.5 Summary

This chapter introduces LBFL, a novel lightweight blockchain framework tailored for FL. To reduce the communication cost of block propagation, and the storage cost of maintaining the blockchain, LBFL incorporates the IPFS to store the global model with its address recorded in the block. Instead of the conventional computation-intensive PoW mechanisms to achieve the consensus in the blockchain, an energy-efficient committee-based consensus protocol using VRF was proposed. Experiment results show that LBFL is secure and scalable.

Chapter 4

Blockchain-empowered Secure and Efficient Federated Edge Learning

Blockchain-based FL system is able to mitigate the single point of failure of conventional FL and guarantee the integrity of the shared global model during the learning process owing to its secure and tamper-resistant feature. However, it is unable to promise the reliability of the global model if adversarial clients exist, who may send poisonous model updates to degrade the performance of the shared model, which is possible in real-world scenarios. Moreover, the deployment of FL for real-world applications is hindered by the high communication overhead between the server and clients that are often at the network edge with limited bandwidth. To address these issues, this chapter proposes a secure and efficient Blockchain-Empowered Federated Learning (BEFL) system based on LBFL, leveraging the state-of-the-art compression mechanism PowerSGD [118] and Mutual Information (MI) between clients' models to capture their inherent correlation and choose reliable updates according to their MI values. Extensive experiments were conducted with benchmark non-independent identically distributed (non-i.i.d) datasets (FEMNIST and CIFAR10) for FL under both honest and adversarial settings. Experimental results show that BEFL is resistant to clients acting maliciously and launching data poisoning and model poisoning attacks. BEFL also achieves better performance than baselines under the benign setting with reduced communication overhead.

4.1 System Design

Based on LBFL, BEFL aims to reduce the communication cost of clients without degrading the global model performance and mitigate the influence of potential malicious clients. As

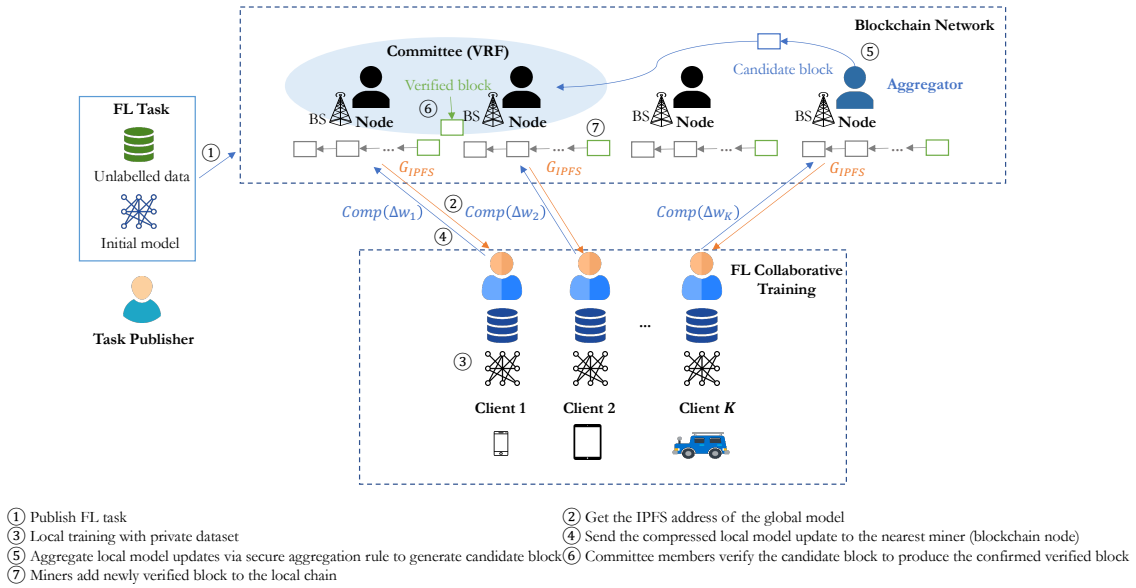

Fig. 4.1 Overview of BEFL framework

shown in Fig. 4.1, BEFL adopts the overall architecture of LBFL. Instead of sending original model updates, clients in BEFL compressed their model updates and send the compressed ones to the blockchain network. BEFL assumes that the task publisher has access to a set of unlabeled data containing all the classes of training data on clients, which is used to calculate the MI between clients. This assumption is easily satisfied, as in FL, the task publisher (or the server) usually collects its own data for model validation, which contains all the classes and is i.i.d, while collecting unlabeled data is simpler than labeled ones [25].

4.1.1 Communication-efficient Distributed Training

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

To save the communication cost of uploading the model update $\Delta w^t = w^t - W_g^t$ for global aggregation, BEFL applies the state-of-the-art gradient compression mechanism PowerSGD, which incorporates error-feedback [118], to compress the transmitted update. As shown

in Algorithm 4, the low-rank PowerSGD decomposes the gradient matrix $M \in R^{n\times m}$ into $\widehat{P} \in R^{n\times r}$ and $Q \in R^{m\times r}$. Compared to traditional FedAvg [88], the additional computation cost brought by compression is relatively low, which involves one left multiplication, one right multiplication, and an orthogonalization (achieved by Gram-Schmidt procedure with $O(mn^2)$, where m and n are the dimensions of the matrix) on each matrix in gradient vector Δw.

Algorithm 4 Distributed model update with PowerSGD

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

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

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

4.1.2 Secure Aggregation Protocol

The privacy-preserving principle of FL makes it susceptible to malicious clients who may send poisonous model updates to corrupt the joint learning process. The low-rank approximated model update inevitably introduces biases to the raw model update, thus making the identification and mitigation of potential malicious model updates more difficult. As shown in [130], the MI between honest clients shows an increasing trend with training round, instead of the Euclidean distance. We propose a novel robust aggregation mechanism leveraging the underlying MI between clients to capture the differences between honest and malicious clients.

MI is a powerful statistic for measuring the degree of dependence [32], it captures the amount of shared information between two random variables. BEFL treats the output of client's local model as a random vector due to the stochastic property of SGD. BEFL assumes that the output F_i of local model w_i from client i and output F_j of local model w_j from client j follow Gaussian functions with respective variances σ_i^2 and σ_j^2. The MI between client i and j, denoted by $MI_{i,j}$ could be calculated as [116]:

$$MI_{i,j} = MI(F_i, F_j) = H(F_i) + H(F_j) - H(F_i, F_j), \tag{4.1}$$

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

$$\left.\begin{aligned} H(F_i) &= \tfrac{1}{2}[1 + \log\left(2\pi\sigma_i^2\right)] \\ H\left(F_j\right) &= \tfrac{1}{2}[1 + \log\left(2\pi\sigma_j^2\right)] \\ H\left(F_i, F_j\right) &= 1 + \log(2\pi) + \tfrac{1}{2}\log[\sigma_i^2\sigma_j^2\left(1 - \rho_{ij}^2\right)] \end{aligned}\right\}. \tag{4.2}$$

In Eq. (4.2) the correlation coefficient ρ_{ij} is defined as:

$$\rho_{ij} = \frac{E[(F_i - E[F_i])]E[(F_j - E[F_j])]}{\sigma_i^2\sigma_j^2}, \tag{4.3}$$

where $E[\cdot]$ denotes the mathematical expectation. From Eq. (4.1) and (4.2), we could get

$$MI_{i,j} = -\frac{1}{2}\log(1 - \rho_{ij}^2). \tag{4.4}$$

It can be seen from Eq. (4.4) that, 1) when the outputs F_i and F_j are highly correlated, the correlation coefficient ρ_{ij} is close to 1 and thus the MI value would be noticeably large; 2) when F_i and F_j are barely correlated, the correlation coefficient ρ_{ij} is close to 0 and their MI

becomes very small. The outputs of clients' local model are obtained from decompressing their model updates and the unlabeled dataset provided by the FL task publisher.

Algorithm 5 BEFL aggregation

The global model W_g^t and momentum value $\mathbf{m^t}$ of last round, unlabeled dataset of FL task X, and the received valid model updates set $S = \{Comp(\Delta w_1), Comp(\Delta w_2), \ldots, Comp(\Delta w_k)\}$

for each compressed model update $Comp(\Delta w_i)$ in S **do**
  $\Delta w_i \leftarrow DECOMPRESS(Comp(\Delta w_i))$
  $w_i = \Delta w_i + W_g$
end for
for each w_i **do**
  Compute $MI_{i,j}$, $(j = (1,\ldots,k), j \neq i)$
  $MI_i = \mathbf{mean}\left\{MI_{i,j} : j = (1,\ldots,k), j \neq i\right\}$
end for
$MI_{mad} \leftarrow MAD(\mathbf{MI})$
$MI_{madn} = \frac{MI_{mad}}{0.6745}$
$\Delta\mathbf{w} \leftarrow 0$
for $i = 1,\ldots,K$ **do**
  $t_i = \frac{|MI_i - Median(\mathbf{MI})|}{MI_{madn}}$
  $count = 0$
  if $t_i \leq 2$ **then**
    $\Delta\mathbf{w} = \Delta\mathbf{w} + \Delta w_i$
    $count++$
  end if
end for
$\mathbf{m^{t+1}} = \beta\mathbf{m^t} - \eta\frac{\Delta\mathbf{w}}{count}$
$W_g^{t+1} = W_g^t - \beta\mathbf{m^t} + (1+\beta)\mathbf{m^{t+1}}$

As shown in Algorithm 5, the MI values between different clients calculated via Eq. (4.4) are leveraged to the secure aggregation process. The correlation coefficient ρ_{ij} is:

$$\rho_{ij} = \frac{\sum_{q=1}^{N}\left(F_i\left(x_q\right) - E\left[F_i\left(x_q\right)\right]\right)\left(F_j\left(x_q\right) - E\left[F_j\left(x_q\right)\right]\right)}{\sqrt{\sum_{q=1}^{N}\left(F_i\left(x_q\right) - E\left[F_i\left(x_q\right)\right]\right)^2\left(F_j\left(x_q\right) - E\left[F_j\left(x_q\right)\right]\right)^2}}, \tag{4.5}$$

where x_q denotes the unlabeled data sample and N is the number of data samples. To quantify the correlation of each model update to others, the MI values between each other model update are averaged to produce the MI score. Similar to the Multi-krum [13] robust aggregation rule, our method selects model updates "close to the barycenter". Model updates with excessively low MI scores are suspected to be malicious for the intrinsic large difference, while the one with a very high score tends to increase the redundancy in model aggregation

which may slow down the convergence speed. To capture the distance of each MI score to the barycenter, BEFL utilises the standard deviation (SD) robust alternative, median absolute deviation about the median (MAD) which is computed as:

$$MI_{mad} = MAD(\mathbf{MI}) = Median(|\mathbf{MI} - Median(\mathbf{MI})|), \tag{4.6}$$

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

$$MI_{madn} = \frac{MI_{mad}}{0.6745}, \tag{4.7}$$

where 0.6745 is the MAD value of a standard normal distribution. To filter out potential malicious model updates as well as less contributive ones, BEFL uses the standard heuristic of taking all values lying within two SDs of the mean (95% probability) as empirically useful. BEFL replaces mean and SD here to median and normalized MAD (MADN) as robust location and dispersion measures [24]. Model updates with MI scores lie within the 95% confidence level are selected for aggregation. The global model is updated with Nesterov's momentum using the selected gradients.

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

4.2 Performance Evaluation

Experiment setup: All experiments were executed over non-i.i.d datasets: FEMNIST and CIFAR10. FEMNIST was preprocessed using the Leaf federated benchmark tool [19] consisting of 62 different classes (numbers and letters). The non-i.i.d CIFAR10 partition constructed from LotteryFL [70] was used, where clients can have different classes of unbalanced data with different degrees. This evaluation assigned each CIFAR10 client 10 classes of data with an unbalanced degree of 0.75. All FL tasks were loaded with 50 workers and 20 of them were selected randomly to participate in each FL training round. The parameter values of BEFL are presented in Table 4.1 unless stated otherwise. The number of blockchain nodes is set to 100, mirroring the default setting of LBFL. The rank values are set to 2 for FEMNIST task and 4 for CIFAR10 task, respectively, as these values yield the best performance among the tested values of 2, 4, 6, and 8. The β parameter is set to 0.9, which is optimal among the tested values of 0.3, 0.6, 0.9, and 0.99. The learning rate is set to 0.1,

Table 4.1 The default parameters of BEFL

Parameter	Value
Number of blockchain nodes	100
Rank of FEMNIST SGD update	2
Rank of CIFAR10 SGD update	4
Beta β	0.9
Eta η	1
Learning rate of client's SGD	0.1
Batch size of client's model training	64 samples
Local steps of client's model training	5

Table 4.2 Communication cost per round

	Raw size (Kb)	Compressed size (Kb)
FEMNIST	3399.742	50.727 $(67\times')$
CIFAR10	764.602	151.070 $(5\times')$

as this value exhibits the best performance among the tested learning rates of 0.001, 0.01, 0.1, and 0.3. The batch size is set to 64 samples, and the local step is set to 5. The unlabeled dataset was constituted of 1000 random samples without their labels in the test dataset. A custom CNN was implemented for the FMNIST task, which comprises a convolutional layer with relu, a max-pooling layer, a convolutional layer with relu, a max-pooling layer, a fully connected layer with relu, and the output layer with softmax. For the CIFAR10 task, this evaluation used ResNet14[49]. Each experiment was repeated 3 times, with the mean and 95% confidence interval (CI) plotted in the relevant figures.

This evaluation compares BEFL with FedAvg and Biscotti. Biscotti shares the same objective of defending against malicious clients and solving the centralized server bottleneck issue in federated learning using blockchain technology.

4.2.1 Communication efficiency

This subsection evaluates the communication cost of each client for participating in a training round. As shown in Table 4.2, the sizes of transmitted model updates for both FEMNIST and CIFAR10 tasks are reduced significantly, from 3399.742kb to 50.727kb in FEMNIST, and 764.602kb to 151.070kb in CIFAR10. By compressing the model updates into low-rank

matrices, the uploading message became $67\times'$ smaller of FEMNIST task and $5\times'$ smaller of CIFAR10 task. To measure the difference between the decompressed model update and the original model update, this subsection utilizes average absolute error per parameter and Euclidean distance which can be computed with the following formula:

$$Euclidean(\Delta w_o, \Delta w_d) = \sqrt{\sum_{k=1}^{n} (\Delta w_{o,k} - \Delta w_{d,k})^2}, \tag{4.8}$$

where Δw_o is the original model update, Δw_d is the decompressed model update, and n is the total number of model parameters. The experiments were rerun for FEMNIST and CIFAR10 tasks with 20 clients participating in each training round. The mean of average absolute error per parameter of decompressed model update for CIFAR10 is 0.00103 while the maximum is 0.00146. For FEMNIST, the mean is 0.00162, while the maximum is 0.02085. The mean Euclidean distance between the decompressed model update and the original model update for CIFAR10 is 1.496404, while the maximum is 2.429850. For FEMNIST, the mean Euclidean distance is 1.771408, while the maximum is 21.806949. From the calculated numerical results, we can observe that the average values are much smaller than the maximum ones, indicating that there is not a significant difference between the decompressed data and the original data from an overall perspective.

As shown in Fig. 4.2, BEFL achieves equivalent performance compared to conventional FedAvg [88] under the honest setting. Due to non-i.i.d data, Biscotti filtered out partial contributive model updates in terms of their long Euclidean distances to others resulting in 3.55% and 3.92% accuracy dropping down for the FEMNIST and the CIFAR10 task respectively. Moreover, BEFL converges faster than traditional FedAvg and Biscotti due to the intrinsic model update selection and momentum update of the global model. Model updates with excessively high MI scores tend to increase the redundancy in the aggregation process and ones with very low MI scores are suspected to induce high variance. Using momentum of SGD updates has proven to accelerate the network training [50] and the oscillations brought by compression could be dampened.

4.2.2 Resistance to poisoning attacks

This subsection evaluates BEFL's performance against both data poisoning and model poisoning attacks. Data poisoning happens during the data collection phase. Malicious clients inject data samples into the training dataset. One common data poisoning attack in the FL scenario is the Label-Flipping (LF) [40] attack in which adversaries replace the targeted label with the desired one. In model poisoning attacks, adversaries adjust the training model

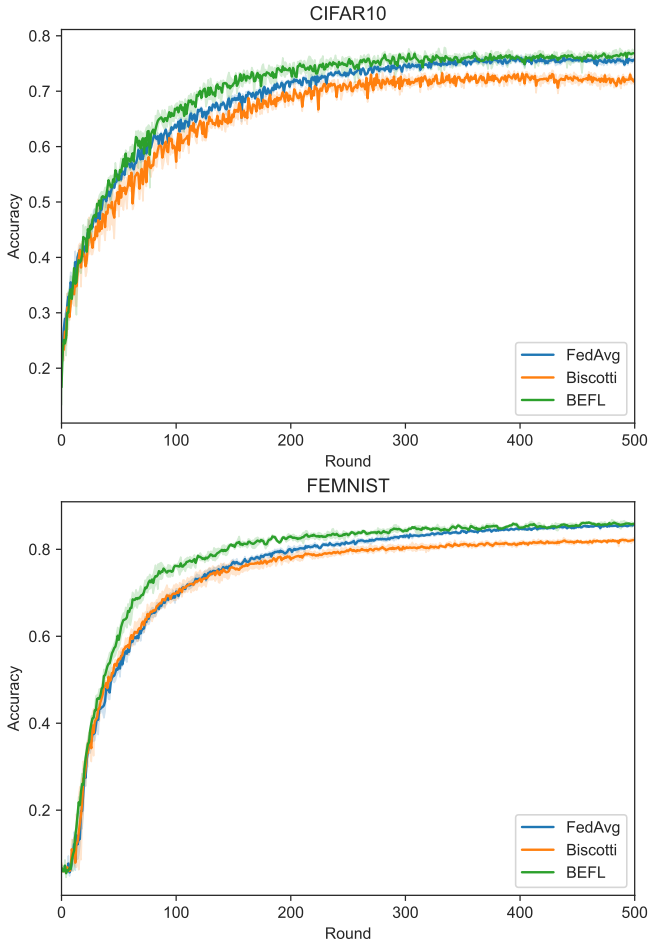

Fig. 4.2 Test accuracy of FedAvg, Biscotti and BEFL over CIFAR10 and FEMNIST datasets when there is no attack. Curves are averages over 3 random trials, shaded regions represent 95% CIs

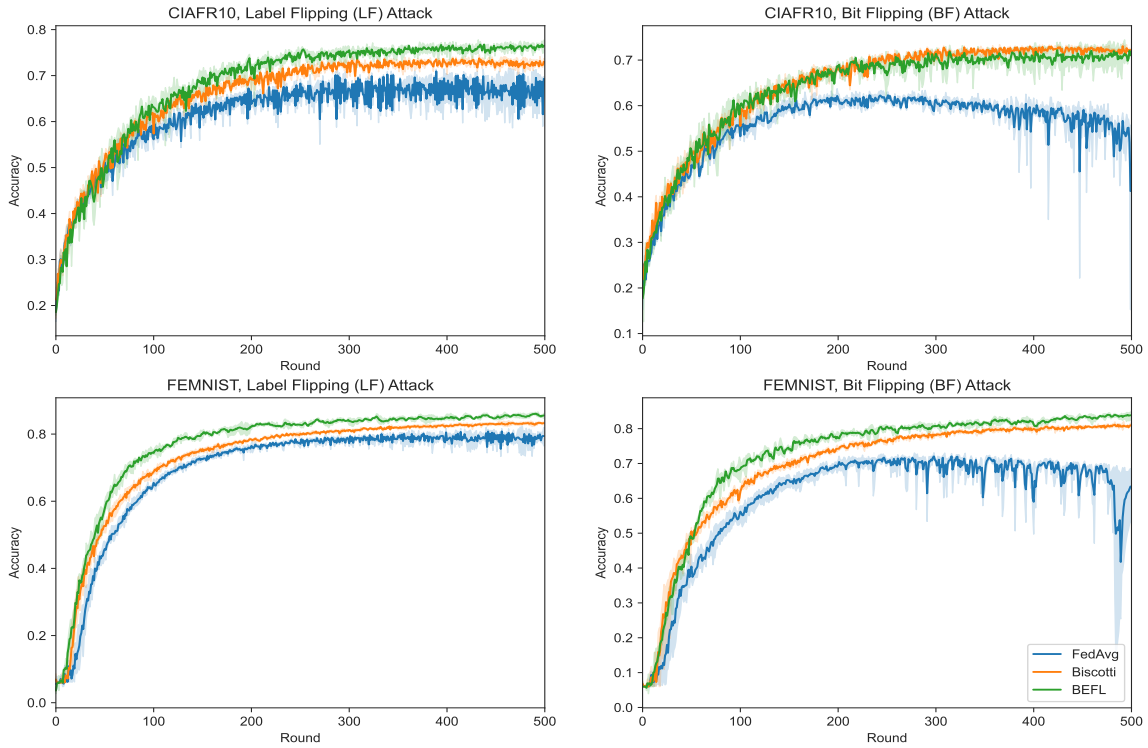

Fig. 4.3 Test accuracy of FedAvg, Biscotti and BEFL over CIFAR10 and FEMNIST datasets when adversaries perform LF and BF attacks. Curves are averages over 3 random trials, shaded regions represent 95% CIs

directly and tailor its output to have a similar distribution with the correct model updates. Bit-Flipping (BF) attack [59] (also known as sign-flipping) is one of the common model poisoning attacks in which adversaries send the negative of the gradients to the master. 20% malicious clients launching LF and BF attacks are simulated in CIFAR10 AND FEMNIST tasks. In particular, malicious clients flipped the label l_c to $9-l_c$ in CIFAR10 and l_f to $61-l_f$ in FEMNIST for the LF attack and multiplied by -1 their gradients before compression in the BF attack. Uncompressed model updates were uploaded when performing FedAvg and Biscotti aggregation. As presented in Fig. 4, BEFL is resilient to both LF and BF attacks as conventional FedAvg struggles to converge. BEFL shows superior performance than Biscotti in FEMNIST task under both attacks. The accuracy achieved by BEFL is higher than that of Biscotti. For CIFAR10 task, it also performed better when adversaries launching LF attack. The performance of BEFL and Biscotti under BF attack is comparable as they achieved nearly the same accuracy.

Table 4.3 Breakdown of time in different phases of BEFL under different settings when processing FEMNIST task

(”number of blockchain nodes”, ”committee size”)	(100, 15)	(100, 30)	(200,15)	(200,30)
committee constitution	0.115	0.185	0.154	0.257
candidate block generation	7.233	7.221	7.445	7.492
voting	**80.272**	**160.619**	**81.568**	**162.968**
verified block propagation	0.298	0.570	0.632	1.139
total time	87.918	168.595	89.800	171.857

Table 4.4 Breakdown of time in different phases of BEFL under different settings when processing CIFAR10 task

(”number of blockchain nodes”, ”committee size”)	(100, 15)	(100, 30)	(200,15)	(200,30)
committee constitution	0.115	0.207	0.170	0.256
candidate block generation	7.719	7.592	7.449	7.562
voting	**92.448**	**185.005**	**93.899**	**186.360**
verified block propagation	0.312	0.587	0.609	1.180
total time	100.595	193.391	102.128	195.358

4.2.3 Blockchain performance

This section evaluates the overhead of each stage in BEFL and measures the performance of BEFL when scaling up committee size and joining of more nodes.

Overhead breakdown: To measure the overhead of each main stage of our design, we simulated BEFL over a varying number of blockchain nodes with varied committee sizes. We captured the amount of time spent in major stages: 1) committee constitution: blockchain nodes run the Sortition algorithm to constitute the committee; 2) candidate block generation: blockchain nodes (except committee members) collect model updates and aggregate them following the BEFL aggregation rule to generate a block; 3) voting: committee members vote for the candidate block and 4) verified block propagation: the distribution of the verified block to each node. As shown in Table 4.3 and 4.4, BEFL was deployed with 100 and 200 nodes with committee sizes of 15 and 30. When $K = 30$, the probability of not having enough candidate committee members is smaller than 2.07×10^{-9}, according to equation (3.4). The values recorded in the table are the averaged time (seconds) of 50 training rounds. Voting takes the longest time among the four stages and it doubles with the committee size since there are twice as many "yes" votes required to confirm the legitimacy of candidate block. The voting time needed in FEMNIST task is lower than that of CIFAR10 task since

the compressed model size of FEMNIST update is smaller than CIFAR10's, thus it takes less time to transmit the candidate block to each committee member. The time needed for committee constitution and verified block propagation is almost the same in the two tasks, both remain very low. The committee constitution time doubles with the committee size, while propagation time doubles with the number of blockchain nodes.

4.3 Summary

This chapter introduces BEFL, a novel lightweight blockchain system for secure, efficient and practical FL at the wireless edge. To reduce the communication cost for participating clients and defend against malicious ones that send poisonous model updates, the communication-efficient MI-guarded training scheme exploiting PowerSGD is proposed. The reliability of the global model is ensured by LBFL which guarantees the secure aggregation execution. The performance results show that BEFL achieves communication efficiency with byzantine robustness, and the reliability of the global model is ensured by the blockchain system.

Chapter 5

Shapley Value-based Robust Federated Learning

The privacy preservation principle makes FL a promising approach in achieving privacy-preserving collaborative intelligence. However, it also brings FL vulnerability to free-riding and poisoning attacks, where free riders dissimulate their participation of training by sending counterfeit yet harmless parameters to the central server, while adversarial clients send poisonous model updates to the central server to degrade the global model performance. These distinctive attacking patterns make the simultaneous defense against both attacks very challenging. To tackle this issue, this chapter introduces a novel Shapley value-based robust federated learning method (SVRFL), which models FL as a cooperative game and leverages Shapley values of model updates to defend against free riders and adversarial clients. Specifically, a client reputation score and a model utility score are computed using the Shapley value. Based on these scores, SVRFL ensures the basic fairness within FL by identifying and eliminating free riders, and realizes adversarial robustness through discarding poisonous model updates, with theoretical convergence guaranteed. Extensive experimental results show that SVRFL can detect typical free-riding attacks with up to 100% precision and is resistant to poisoning attacks launched by adversarial clients.

5.1 Preliminaries

5.1.1 Shapley Value

Shapley value is a classic game theory solution for fairly distributing rewards to players in a cooperative game, which has been widely applied in economics [47, 85] and management science [35]. It averages the marginal contribution of one player to all possible coalitions.

Consider a set of N players I and a value function v that assigns a real value to each coalition, representing the total reward achieved by that coalition. The Shapley value of player i is calculated as:

$$\phi_i(v) = \sum_{S \subseteq I \setminus \{i\}} \frac{|S|!(N-|S|-1)!}{N!} [v(S \cup \{i\}) - v(S)]. \tag{5.1}$$

As originally shown by Shapley [108], the Shapley value enjoys several desirable properties:

- Efficiency: for any value function v, $\sum_{i \subseteq I} \phi_i(v) = v(I)$.
- Symmetry: for player i and j, if $v(S \cup i) = v(S \cup j)$ for every coalition S that contains neither i or j, then $\phi_i(v) = \phi_j(v)$.
- Null player: for any v, if i is a null player, then for all $S \subseteq I \setminus \{i\}$: $v(S \cup i) = v(S)$, such that $\phi_i(v) = 0$.
- Linearity: if value function v could be write as $v = v_1 + v_2$, then $\phi_i(v_1 + v2) = \phi_i(v_1) + \phi_i(v_2)$. For $a \in \mathbb{R}$, $\phi_i(av) = a\phi_i(v)$.

Unique in natural properties of fairness in cooperative game theory, Shapley value has been increasingly applied in machine learning for data valuation [43, 67, 54]. It has also attracted increasing attention in FL as a fair contribution evaluation measurement [120, 122].

5.2 Threat model

Grounded in the principle of privacy preservation, FL inherently opens the door to vulnerabilities to free riders and adversarial clients. This work investigates the defense strategy against recently emerging free-riding attack [39], and poisoning attacks, including untargeted attack which is designed to degrade the overall performance of the global model (*i.e.*, sign-flipping [73] attack and a little is enough (ALIE) attack[11]) and targeted attack that is designed to manipulate the global model to serve a specific detrimental objective (*i.e.*, label-flipping attack [40]).

Free-riding: A free-riding attack is explicitly conceived to stay undetected while not disturbing the FL process. Fraboni *et. al.* [39] proposed plain free-riding and disguised free-riding attacks with theoretical analysis of not compromising the convergence of the global model. In plain free-riding attack, the free rider simply returns the zero gradient, which is easy to detect. While the disguised free-riding attack adds small additive noise to mimic SGD updates. Specifically, the free rider submits a local model update $g' = \varphi(t)\varepsilon_t$ to the central server, where ε_t is Gaussian white noise, and $\varphi(t)$ is the time-varying perturbation

coefficient approximated as $\varphi(t) = \sigma t^{-\gamma}$. Here, t represents the current training round. The parameter σ corresponds to practical assumptions on the parameter evolution during federated learning, and can be estimated using the update distribution $\Delta w_g = w_g^1 - w_g^0$ after random initialization of the global model. The parameter γ serves as the decay factor. In this work, we assume free riders to conduct disguised free-riding attacks.

Sign-flipping: A sign-flipping attack involves the inversion of the sign of a local model update to disrupt the convergence of the global model. Specifically, an adversarial client i trains their local model using the received global model and computes the gradient g_i. Rather than submitting g_i to the central server, the client uploads a manipulated version ug_i, where u is a negative constant that inverses the gradient direction.

A Little is Enough (ALIE): ALIE represents a time-coupled attack capable of deceiving robust aggregation rules such as Krum, TM, and Bulyan. In such an attack, the adversary subtly adds small perturbations to the average of model updates from benign clients in a single training round. These minute perturbations accumulate over time, leading to model divergence. Specifically, the adversary calculates the benign model updates' average μ and standard deviation δ, and then constructs the corrupted model update as $g' = \mu - z\delta$. Here, z is a coefficient determined by the number of participating clients and adversaries.

Label-flipping: During a label-flipping attack, an adversary deliberately alters the labels of their local data prior to training. For instance, in an image classification task, to force the global model to misclassify images labeled "1" as "7", the adversary would flip the labels from "1" to "7" in their training data.

5.3 SVRFL design

In response to these free riders and adversarial clients, SVRFL is proposed with the following goals:

- Accurate detection of free-riding attacks. SVRFL should be able to identify free riders with low false positive rate and exclude them in the FL process to ensure the basic fairness in FL.
- Resilience against poisoning attacks. SVRFL should withstand both untargeted and targeted attacks from adversarial clients. It is able to mitigate the impact of malicious model updates.
- Effectiveness in non-i.i.d FL setting. SVRFL should be effective in non-i.i.d FL environment as clients' data are heterogenous in real-world scenarios.

Design overview: SVRFL follows the general steps of FL discussed in Section 1.2. In step 3, instead of performing the widely used FedAvg [88] aggregation rule, SVRFL introduces the reputation score and model utility score to defend against free riders and adversarial clients leveraging Shapley value and cosine similarity. This work assumes that the central server could collect a small validation dataset D_v (*i.e.,* 100 validation samples) for Shapley value calculation, which is reasonable as the required validation dataset is quite small (*i.e.,* 100 validation samples), manual collection and labeling are usually feasible for the server [20]. For example, Google enlists its staff to generate validation dataset for its federated next word prediction via typing with Gboard [1]. This work also assumes the byzantine clients (free riders and adversarial clients) consistently perform attacks during the whole global model training period.

5.3.1 Free-riding attack detection:

Aiming to obtain the aggregated model of benign clients, free riders engage in FL by disseminating deceptive parameters. These parameters, which are small in magnitude, adapt and evolve with the progress of the global model training. This strategic mode of attack cleverly imitates the general form of benign model updates, posing a huge challenge for accurate attack detection. In order to tackle this issue, Shapley value and cosine similarity are leveraged to design an effective detection mechanism.

Shapley value has been applied in data valuation [124], contribution evaluation [120, 122], and relevant client selection [94] in FL. The main idea is that a client's model update with a higher Shapley value often suggests higher quality of data, thus contributing more to the global model performance improvement. Inspired by this, this work investigated the Shapley value of free rider and benign client. FL can be regarded as a cooperative game, wherein participating clients in the collaborative training process are players of the game. They work together to bring the best global model. Suppose there are n clients in the FL task, and k of them would be randomly selected in each training round which constitutes the set I_t. The value function for Shapley value calculation is defined as the validation loss decrease,

$$v(S, D_v) = F(w_g^t, D_v) - F(w_S^t, D_v), \tag{5.2}$$

where w_g^t is the global model of current round t and the aggregated model parameter w_S^t is computed as:

$$w_S^t = w_g^t - \frac{\eta}{|S|} \sum_{i \in S} g_i^t. \tag{5.3}$$

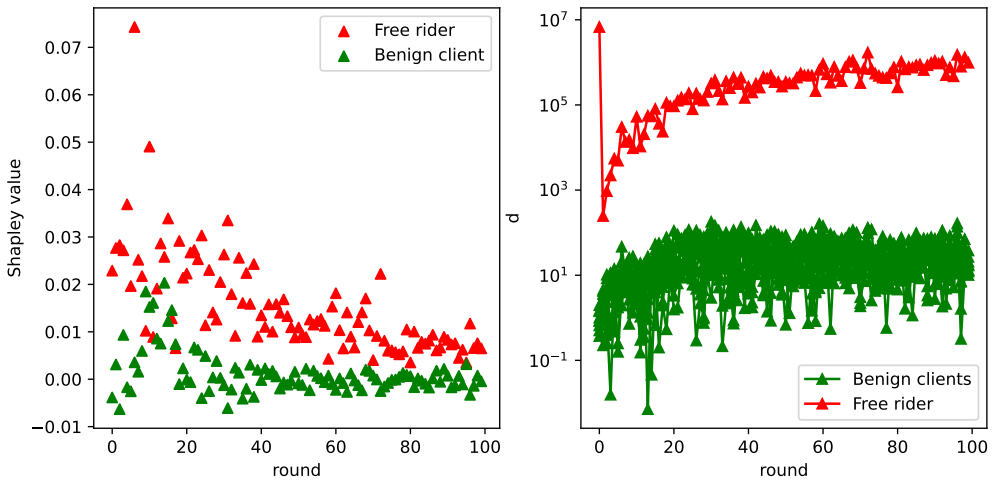

Fig. 5.1 Left: the Shapley value of free rider and the averaged Shapley value of benign clients. Right: the d values of 10 clients (1 free rider and 9 benign clients). Experiment run on non-i.i.d MNIST dataset

The set $S \subseteq I_t$ and η is the server-side learning rate. Given the value function, the Shapley value of client i at round t, denoted as sv_i^t, could be calculated according to equation (5.1).

Intriguingly, as shown in Figure 1, the empirical findings suggest that the Shapley value of a free rider is generally larger than that of benign clients. This may seem counterintuitive at first glance, but it makes sense when considering the strategies employed by free riders. Their disguised model update g_f is deliberately small in magnitude to avoid impairing the aggregated model performance of benign clients. Thus, for a coalition $S \in I_t$ and a free rider f, we can approximate the aggregated model

$$w_{S+f}^t = w_g^t - \frac{\eta}{|S|+1}(\sum_{i \in S} g_i^t + g_f) \approx w_g^t - \frac{\eta}{|S|+1}\sum_{i \in S} g_i^t. \tag{5.4}$$

The model update from the free rider essentially acts as a counterbalance, pulling the aggregated model, built from updates from S, closer to the global model. This subtle influence helps to limit the impact of diverse model updates that arise due to heterogeneity in local data across clients. This "balancing act" performed by the free rider consequently increases their Shapley value, since they seemingly contribute to maintaining the stability of the global model, and hence are perceived as "valuable" in the coalition. Given this observation, the cosine similarity between client's local model and the global model to is employed to discern the nuances between a free rider and a benign client. Notably, a free rider's local model tends to exhibit higher cosine similarity with the global model compared

to a benign client. Thus, SVRFL calculates a feature value d_i of client i as:

$$d_i = \frac{|sv_i|}{1 - \frac{<w_i, w_g>}{||w_i||||w_g||}}, \tag{5.5}$$

where w_i is the local model of client i, and w_g is the global model. As shown in Figure 5.1, the d value of the free rider is significantly larger than that of the benign client. SVRFL applies the classic clustering method, K-means, to cluster d values into 2 clusters, c_1 and c_2, and denote their corresponding center positions as P_{c1} and P_{c2}. In each training round, if the positions of the two cluster centers have an order-of-magnitude difference, which is $max(P_{c1}, P_{c2}) > h \cdot min(P_{c1}, P_{c2})$, where h is the threshold, the client with a d value lying in the cluster with a numerically larger average d value would be regarded as a free-rider.

5.3.2 Poisonous model update mitigation

Unlike free riders, adversarial clients work towards deteriorating the performance of global model by sending poisonous model updates to the central server. To counter these harmful influences, SVRFL further exploits Shapley value to measure the utility of client's local model update, as it represents the averaged marginal contribution to model improvement on all coalitions of participating clients.

Let $U = (u_1, u_2, \cdots, u_n)$ be the utility vector that is private to the central server, wherein u_i represents the utility score of the model update sent from client i. These utility scores are initialized to 0 at the onset of the FL task. In each training round, the utility score of the received model update from client i is computed as:

$$u_i = \alpha u_i + (1 - \alpha) sv_i; \forall i \in I_t, \tag{5.6}$$

where $\alpha \in (0, 1]$. SVRFL incorporates the past Shapley values in utility measurement to 1) prioritize model update from client who consistently has a high Shapley value across multiple rounds, 2) be less responsive to sudden changes in round contributions, and 3) learn from history to develop a defense against time-coupled poisoning attacks (*i.e.*, ALIE). Model update with a positive utility score will be selected for global model aggregation. The global model is updated as follows:

$$w_g^{t+1} = w_g^t - \frac{\eta}{\sum_{i \in I_t}^{|I_t|} s(u_i)} \sum_{i \in I_t}^{|I_t|} s(u_i) g_i^t, \tag{5.7}$$

where

$$s(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{otherwise.} \end{cases} \tag{5.8}$$

5.3.3 Complete SVRFL

Algorithm 6 shows the complete SVRFL method. All clients will be initialized with zero reputation score and utility score. In each training round, the server will randomly choose $m = \min(|I|, k)$ eligible clients whose reputation score is above the reputation threshold γ, and send the latest global model to them. Each selected client will train his/her local model using the received global model via performing stochastic gradient descent, and send the model update back to the server. After receiving the gradients from selected clients, the server first calculates the Shapley values of model updates. As the exact computation of Shapley value requires an exponential number of model inferences, this work applies Monte Carlo simulations to approximate the Shapley value as shown in algorithm 7 [22, 86]. The server then performs free rider detection via calculating d values and employing K-means algorithm. If there is a free rider detected, its reputation score will be deducted by 1 and its Shapley value will be assigned with 0, as its zero contribution in this round. Next, the server will compute the utility scores of model updates according to their Shapley values and update the global model accordingly. Clients that contribute model updates with positive utility scores are rewarded with an increment in reputation by $\frac{1}{\sum_{i \in I_t}^{|I_t|} s(u_i)}$. Only clients with positive reputation scores could obtain the well-trained global model at the end of FL training process.

5.3.4 Theoretical analysis

This section theoretically analyzes the difference of the global model learned by SVRFL and the optimal global model $w^* = \arg\min_{w \in \Theta} F(w)$ under no attack is bounded, where Θ is the model parameter space. As discussed in Section 2.2, $F(w) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_{\xi_i \sim D_i}[F(\xi_i, w)]$, where ξ_i is the training samples sampled from client i's training data D_i. The following assumptions which are made, which are commonly used in the literature [143, 61, 92, 20].

Assumption 1. *The expected loss function $F(w)$ is μ-strongly convex and differentiable over Θ with L-Lipschitz gradient. That is, for all $w, w' \in \Theta$,*

$$F(w') \geq F(w) + \langle \nabla F(w), w' - w \rangle + \frac{\mu}{2} \|w' - w\| \tag{5.9}$$

$$\|\nabla F(w) - \nabla F(w')\| \leq L \|w - w'\| \tag{5.10}$$

Algorithm 6 SVRFL

Input: The client set I consists of n clients, the number of selected clients in each training round k, a small validation dataset Dv, the server-side learning rate η, the number of training round T, reputation threshold γ, cluster distance threshold h, the number of Monte Carlo simulations R, and utility update parameter α.

Output: The trained global model w_g.

$w_g \leftarrow$ random initialization
$r_i \leftarrow 0, \forall i \in I$, reputation score initialization
$u_i \leftarrow 0, \forall i \in I$, utility score initialization
for round $t = 0, 1, 2, \cdots, T$ **do**
  server executes:
  for each client i in I **do**
    if $r_i < \gamma$ **then**
      $I \backslash \{i\}$; //remove client i from FL task
    end if
  end for
  $m = min(|I|, k)$
  $I_t \leftarrow$ random sample m clients from I

  client executes:
  for each client i in I_t **do**
    compute gradient g_i^t using received global model w_g^t
    communicate g_i to the server
  end for

  server executes:
  $g^t \leftarrow \{g_i^t\}$; //collect clients' updates
  $sv^t = SVEstimate(g^t, w_g^t, D_v, \eta, R)$
  $w_i^t = w_g^t - g_i^t, \forall i \in I_t$, //reconstruct clients' local models
  $d_i = \frac{|sv_i^t|}{1 - cos(w_i^t, w_g^t)}, \forall i \in I_t$
  cluster $c_1, c_2 \leftarrow$ K-means$(d = (d_1, d_2, \cdots, d_m))$
  cluster center position $P_{c1}, P_{c2} \leftarrow c_1, c_2$
  if $max(P_{c1}, P_{c2}) > h \cdot min(P_{c1}, P_{c2})$ **then**
    for client i whose d_i in the cluster of which center position is $max(P_{c1}, P_{c2})$ **do**
      $r_i = r_i - 1$
      $sv_i = 0$
    end for
  end if
  $u_i = \alpha u_i + (1 - \alpha) sv_i, \forall i \in I_t$, //update utility score
  if $max(\{u_i, \forall i \in I_t\}) > 0$ **then**
    $w_g^{t+1} = w_g^t - \frac{\eta}{\sum_{i \in I_t}^{|I_t|} s(u_i)} \sum_{i \in I_t}^{|I_t|} s(u_i) g_i^t$
    $r_i = r_i + s(u_i) \frac{1}{\sum_{i \in I_t}^{|I_t|} s(u_i)}, \forall i \in I_t$, //update reputation.
  end if
end for

Algorithm 7 SVEstimate

Input: Model updates from clients g^t, global model w_g^t, validation dataset D_v, server-side learning rate η, and the number of Monte Carlo simulations R.

Output: The Shapley values of the received model updates sv.

$M \leftarrow$ set of clients in g^t
$P \leftarrow$ set of R permutations of g^t
$sv_i = 0, \forall i \in M$
for each permutation $p \in P$ **do**
  $S_{p,i} = \{j | j \in M \bigwedge p(j) \leq i\}$
  $sv_i = sv_i + \frac{1}{R}(v(S_{p,i} \bigcup i, D_v) - v(S_{p,i}, D_v))$
end for
$sv \leftarrow \{sv_i\}_{i \in M}$

Assumption 2. *For each client, the stochastic gradient is locally unbiased:*

$$\mathbb{E}_{\xi_i \sim D_i}[F(\xi_i, w)] = F_i(w). \tag{5.11}$$

Assumption 3. *The stochastic gradient of each client has a bounded variance uniformly, and the deviation between local and global gradient is bounded:*

$$\mathbb{E}_{\xi_i \sim D_i}[\|\nabla F(\xi_i, w) - \nabla F_i(w)\|^2] \leq \sigma^2 \tag{5.12}$$

$$\|\nabla F_i(w) - \nabla F(w)\| \leq \tau^2 \tag{5.13}$$

Since free riders will not impair the convergence of the global model, this subsection here investigates the scenario where a β fraction of clients are adversarial in the FL system. In SVRFL, there may exist a certain part of malicious gradients that have positive utility scores in a training round, similar to [136], this subsection here makes another assumption about the defense capability of the aggregation:

Assumption 4. *For problem (2.1) with βn adversarial clients and $(1-\beta)n$ benign clients, suppose that at most δm ($0 \leq \delta \leq \beta \leq 0.5$) adversarial clients can circumvent SVRFL at each training round, where m is the number of selected clients in each round. Let $\mathscr{G}$ denote the set of benign clients in a training round, $\mathbb{E}|\mathscr{G}| = (1-\beta)m$. The averaged benign gradients is $\bar{g}^t = \frac{1}{|\mathscr{G}|}\sum_{i \in \mathscr{G}} g_i^t$. We assume that there exist positive constant c and b, such that $\hat{g}^t = \frac{1}{\sum_{i \in I_t}^{|I_t|} s(u_i)} \sum_{i \in I_t}^{|I_t|} s(u_i) g_i^t$ satisfies:*

$$[\mathbb{E}\|\hat{g}^t - \bar{g}^t\|]^2 \leq c\delta \sup_{i,j \in \mathscr{G}} \mathbb{E}[\|g_i^t - g_j^t\|^2] \tag{5.14}$$

$$\text{var}[\|\hat{g}^t\|] \leq b^2 \tag{5.15}$$

Theorem 1. *Suppose Assumption 1-4 hold and SVRFL uses learning rate η that $\sqrt{\frac{1}{2L^2}+\frac{\mu^2}{4L^4}}+\frac{\mu}{2L^2} < \eta < \sqrt{\frac{1}{L^2}+\frac{\mu^2}{4L^4}}+\frac{\mu}{2L^2}$. For any number of adversarial clients, the difference between the global model learnt by SVRFL and the optimal global model w^* under no attacks is bounded. We have the following:*

$$\mathbb{E}[\|w^t - w^*\|^2] \leq (1-q)^t \mathbb{E}[\|w^0 - w^*\|^2] + \frac{\eta^2 \Delta_1}{q}, \tag{5.16}$$

where $\Delta_1 = 8c\delta(\sigma^2+\tau^2)+4b^2+\frac{2\beta^2\tau^2}{(1-\beta)^2}+\frac{2\sigma^2}{(1-\beta)m}$, and $q = 2\eta\mu - 2\eta^2L^2 - 1$. By choosing $\eta \in (\sqrt{\frac{1}{2L^2}+\frac{\mu^2}{4L^4}}+\frac{\mu}{2L^2}, \sqrt{\frac{1}{L^2}+\frac{\mu^2}{4L^4}}+\frac{\mu}{2L^2})$, $0 < 1-q < 1$, we have $\lim_{t\to\infty}\mathbb{E}[\|w^t - w^*\|^2] \leq \frac{\eta^2\Delta_1}{q}$.

Proof. The detailed proof is in the A.1

This section also investigates the convergence of SVRFL in the honest setting, where all clients are benign. With the following assumptions, it can be proved that the global model learnt by SVRFL and the optimal global model w^* is bounded.

Assumption 5. *For any subset of clients' gradients, and any $\delta \in (1,0)$, there exists a positive L_1 such that,*

$$\mathbb{P}\left\{ \sup_{w,w' \in \Theta; w \neq w'; S \in I} \frac{\|\frac{1}{|S|}\sum_{i=1}^{|S|}(\nabla f(D_i,w) - \nabla f(D_i,w'))\|}{\|w-w'\|} \leq L_1 \right\} \geq 1 - \frac{\delta}{3}, \tag{5.17}$$

where I is the set of clients.

Definition 1. (Sub-exponential). *Random variable X with mean μ is sub-exponential if $\exists v > 0$ and $\alpha > 0$ such that,*

$$\mathbb{E}[exp(\lambda(X-\mu))] \leq e^{\frac{v^2\lambda^2}{2}}, \forall|\lambda| \leq \frac{1}{\alpha_1} \tag{5.18}$$

Assumption 6. *The gradient of empirical loss function is bounded at the optimal global model w^*. Specifically, there exist positive σ_1 and α_1 such that for any unit vector $v \in B$, $\langle \nabla f(D,w^*), v\rangle$ is sub-exponential with scaling parameters (σ_1, α_1),*

$$\sup_{v\in B} \mathbb{E}[exp(\lambda\langle \nabla f(D,w^*), v\rangle)] \leq e^{\frac{\sigma_1^2\lambda^2}{2}}, \forall|\lambda| \leq \frac{1}{\alpha_1}, \tag{5.19}$$

where B denotes the unit sphere $\{v : \|v\|_2 = 1\}$.

Assumption 7. *There exist positive constants σ_2 and α_2 such that for any $w \in \Theta$ with $w \neq w^*$*

and unit vector $v \in B$, the gradient difference $h(D,w) \triangleq \nabla f(D,w) - \nabla f(D,w^*)$ normalized by $\|w - w^*\|$ is sub-exponential with scaling parameters (σ_2, α_2),

$$\sup_{v \in B; \theta \in \Theta} \mathbb{E}\left[exp\left(\frac{\lambda \langle h(D,w) - \mathbb{E}[h(D,w)], v\rangle}{\|w - w^*\|}\right)\right] \leq e^{\frac{\sigma_2^2 \lambda^2}{2}}, \forall |\lambda| \leq \frac{1}{\alpha_2}. \tag{5.20}$$

Assumption 8. *Each client's local training dataset $D_i (i = 1, 2, \cdots, n)$ is sampled independently from the distribution $\mathscr{X}$.*

Theorem 2. *Suppose Assumption 1, 5-8 hold. When there is no attacker, the difference between the global model learnt by SVRFL and the optimal global model is bounded. Formally, we have the following with probability at least $1 - \delta$:*

$$\|w^t - w^*\| \leq (1-p)^t \|w^0 - w^*\|^2 + \frac{2\Delta_1 \eta + 4\Delta_3 \tau \eta}{p} \tag{5.21}$$

where $p = 1 - (\sqrt{1 - \frac{\mu^2}{4L^2}} + 8\Delta_3 \eta)$,

$$\Delta_1 = \begin{cases} \sqrt{2}\sigma_1 \sqrt{d \log 6 + \log \frac{3}{\delta}}, & \text{if } K \leq \frac{\sigma_1}{\alpha_1} \\ 2\alpha_1 (d \log 6 + \log \frac{3}{\delta}), & \text{otherwise} \end{cases}, \tag{5.22}$$

$$\Delta_3 = \begin{cases} \sqrt{2}\sigma_2 \sqrt{d \log \frac{18K_1}{K_2} + \frac{1}{2} d \log \frac{n}{d} + \log \frac{6\sigma_2^2 r\sqrt{n}}{\alpha_2 \sigma_1 \delta}}, & \text{if } K_3 \leq K_4 \\ 2\alpha_2 (d \log \frac{18K_1}{K_2} + \frac{1}{2} d \log \frac{n}{d} + \log \frac{6\sigma_2^2 r\sqrt{n}}{\alpha_2 \sigma_1 \delta}), & \text{otherwise} \end{cases}, \tag{5.23}$$

$K_1 = \max(L, L_1)$, $K_2 = \min(\alpha_2, \sigma_2)$, $K_3 = d \log(18K_1) + \frac{1}{2} d \log \frac{n}{d} + \log \frac{6r\sqrt{n}}{\sigma_1, \delta}$, *and* $K_4 = d \log K_2 - \log \frac{\delta_2^2}{\alpha_2} + \frac{\sigma_2^2}{2\alpha_2^2}$, $\tau = \frac{\alpha_2 \sigma_1}{2\sigma_2^2} \sqrt{\frac{d}{n}}$, *$d$ is the dimension of w.*

Proof. The detailed proof is in the A.2

5.4 Evaluation

Datasets and Models: Two widely used benchmark datasets, MNIST and CIFAR10, were used. Both consist of images in 10 classes to evaluate SVRFL. MNIST contains 60000 training samples and 10000 test samples. CIFAR10 has 50000 training samples and 10000 test samples. This evaluation considered $n = 20$ clients and distributed the training samples to clients following the previous work [21]. Specifically, these clients will be uniformly split into 10 groups. A training sample with label l will be assigned to the l^{th} group with probability p, to each other group with probability $\frac{1-p}{9}$. Data are uniformly distributed to

Table 5.1 Hyperparameters

	MNIST	CIFAR10
Client's learning rate	0.1	0.12
Server learning rate	$\eta = 1$	
Local training step	5	10
Batch size	64	
RFFL	$\alpha = 0.98$, $\beta = \frac{1}{5n}, \gamma = 0.5$	$\alpha = 0.975$, $\beta = \frac{1}{500n}, \gamma = 1$
Multi-Krum	$f = 6$	
TM	$\beta = 0.3$	
SVRFL	$\gamma = -2, h = 50, R = 2m, \alpha = 0.2$	

each client within the same group. The parameter p is the controlling factor of the distribution difference among clients' training data, $p = 0.1$ means clients' data are i.i.d. This evaluation sets $p = 0.5$ by default. 100 randomly selected samples from test data of a dataset is the validation dataset. A convolutional neural network (CNN) architecture is used for MNIST task, which comprises a convolutional layer with relu, a max-pooling layer, a convolutional layer with relu, a max-pooling layer, a fully connected layer with relu, and the output layer with softmax. For CIFAR10 task, the ResNet20 [49] is used.

Baselines: SVRFL is compared with classic FedAvg[88], free-riding defense mechanism (RFFL), and popular robust aggregation rules as follows:

RFFL: RFFL leverages the cosine similarity between local model update and the aggregated global model update to calculate the reputation score of a client, and removes client with reputation score lower than the threshold β. The reputation of client i in round t is computed as $r_i^t = \alpha r_i^{t-1} + (1-\alpha) cos(g_i^t, \overline{g_t})$, where α is the moving average coefficient, and $\overline{g_t} = \frac{1}{\sum_{i \in I} r_i^{t-1}} \sum_{i \in I} r_i^{t-1} g_i^t \times \frac{\gamma}{||g_i^t||}$ is the aggregated gradient, where γ is the normalization coefficient to prevent gradient explosion. Instead of getting the full aggregated gradient, clients obtain the sparsified version of the aggregated gradient and update the local model for the next round of training. The degree of sparsification is determined by their reputation, the higher their reputation, the less sparsified gradient downloaded.

FLTrust[20]: FLTrust uses a small clean validation dataset on the server side to train the model simultaneously with clients. Leveraging cosine similarity between client's model update and server model update, FLTrust computes a trust score $TS_i = Relu(cos(g_i, g_0))$, where g_i is the model update from client i and g_0 is the server model update, and performs trust score-weighted aggregation to update the global model $w_g = w_g - \eta \frac{1}{\sum_{i=1}^{k} TS_i} \sum_{i=1}^{k} TS_i \frac{||g_0||}{||g_i||} \cdot g_i$.

Multi-Krum[13]: Instead of using the cosine similarity, Multi-Krum utilizes the Euclidean distance between local model updates to filter out potential malicious ones. Specifi-

cally, Multi-Krum computes the score of each model update. For model update g_i from client i, its $score_i = \sum_{i \to j}^{j} ||g_i - g_j||^2$, where sum runs over $k - f - 2$ closest model updates to g_i and f is the number of malicious model updates. Multi-Krum sorts these scores and selects smallest $k - f$ ones for final averaging aggregation.

TM[142]: TM performs element-wise aggregation. For each coordinates d, TM sorts the coordinate values of all received model updates $\prod_d$ and computes the trimmed average of them, which could be expressed as: $x_d = \frac{1}{(1-2\beta)k} \sum_{i=\beta k}^{k-\beta k} \prod_d(i)$.

Coordinate Median (CM)[142]: Similar to TM, CM performs coordinate-wise operation as well. It takes the median of coordinate values of all received model updates. The d^{th} parameter value of aggregated gradient is computed as $x_d = median(\prod_d)$.

FL settings: For MNIST task, the training round $T = 100$ was set for all algorithms, except FLTrust which requires $T = 1000$ to achieve its best performance. For CIFAR10 task, $T = 2000$ was set for FLTrust, and $T = 300$ was set for the other algorithms. 6 clients out of 20 are malicious and 10 clients are randomly selected to participate in each training round. The hyperparameters of algorithms are presented in Table 5.1 unless stated otherwise. We randomly sample 1000 samples from the training data of a dataset as the validation dataset for FLTrust as suggested. For attacking strategies, we set the decaying factor $\gamma = 1$ for free-riding attack, the negative constant $u = -1$ for sign-flipping attack, and change the "1" labels to "7" in malicious clients' training data in MNIST task to forge the global model to misclassify handwritten digit "1" as "7", and flip "1" labels to "9" in CIFAR10 task to lead the global model to misclassify "automobile" as "truck".

Evaluation metrics: *Precision* is used to measure the accuracy of free rider detection, which is computed as:

$$precision = \frac{TP}{TP + FP}, \tag{5.24}$$

where TP stands for True Positive, indicating the free rider is correctly identified, and FP stands for False Positive, indicating a benign client is misidentified as a free rider. We use *free rider detection rate (FRDR)* to represent the fraction of free riders that are successfully detected and excluded from the FL task. We use *test accuracy* to measure the performance of the learnt global model. For the label-flipping attack, this evaluation uses *attack success rate (ASR)*, which is the fraction of test samples with source label that are misclassified as the targeted ones, i.e., the sample with label "1" (source label) is predicted as "7" (target label) in MNIST task, and *target accuracy (TACC)*, which is the target label accuracy, to measure the defense performance of the algorithms.

Table 5.2 The performance of free rider detection when all clients participate in each training round

	MNIST		CIFAR10	
	Precision	FRDR	Precision	FRDR
SVRFL	100%	100%	100%	100%
RFFL	93.80%	100%	32.60%	100%

Table 5.3 The performance of free rider detection when a subset of clients participates in each training round

(a) MNIST

Participation rate	0.3	0.5	0.7	0.9
FRDR	100%	100%	100%	100%
Precision	100%	100%	100%	100%

(b) CIFAR10

Participation rate	0.3	0.5	0.7	0.9
FRDR	100%	100%	100%	100%
Precision	87.5%	100%	100%	100%

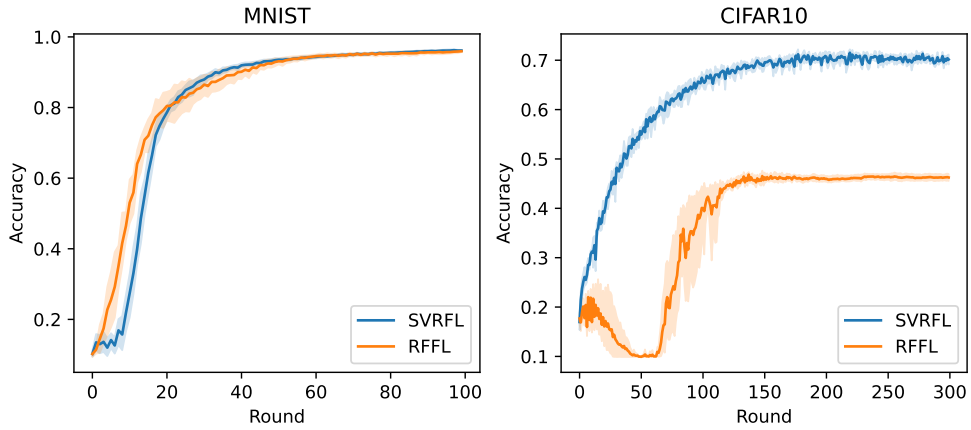

Fig. 5.2 Test accuracy of SVRFL and RFFL when free riders perform free-riding attack. Curves are averages over 5 random trials, shaded regions represent 95% confidence intervals.

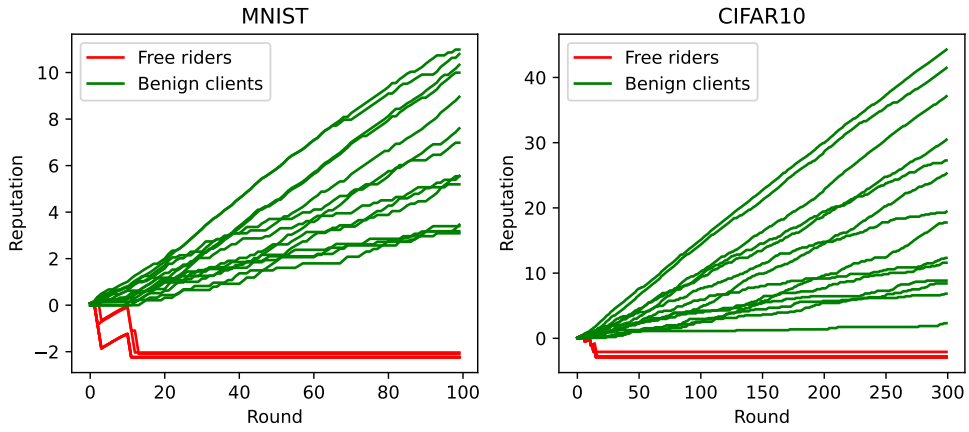

Fig. 5.3 Reputation values of free riders and benign clients in MNIST and CIFAR10 task.

5.4.1 Defense against free-riding attack

This subsection compares SVRFL with RFFL. In this experiment, the number of participating clients in each training round was set to 20, mirroring the operational condition of RFFL, which presupposes the participation of clients in every round. As shown in Table 5.2, both defense mechanisms accomplish 100% FRDR in two tasks. SVRFL outperforms RFFL in terms of precision, achieving 100% in both tasks. While RFFL performs commendably on MNIST, it struggles to accurately distinguish between free riders and benign clients in the CIFAR10 task. This is because the non-i.i.d data distribution together with the more complex model architecture of CIFAR10, resulting in diverse model updates and low reputations of benign clients due to their negative cosine similarity values with the aggregated gradient. As reflected in Figure 5.2, RFFL filtered out too many benign clients with free riders, leading to significantly degraded global model performance. In contrast, SVRFL successfully identifies and removes free riders at an early stage of the FL task, as depicted in Figure 5.3. The reputations of free riders rapidly plummet beyond the established threshold. For this experiment, we set the threshold value γ to -2, implying that if a free rider attempts to obtain the global model without real contribution twice, they will be promptly barred from the FL system. Moreover, as shown in Table 5.3, SVRFL also performs well in the scenario where partial clients participate in each training round – a condition that more accurately reflects real-world circumstances.

5.4.2 Defense against poisoning attacks

This subsection evaluates the performance of SVRFL under untargeted and targeted attacks. SVRFL is compared with byzantine-robust FLTrust, Multi-Krum, TM, and CM, and conven-

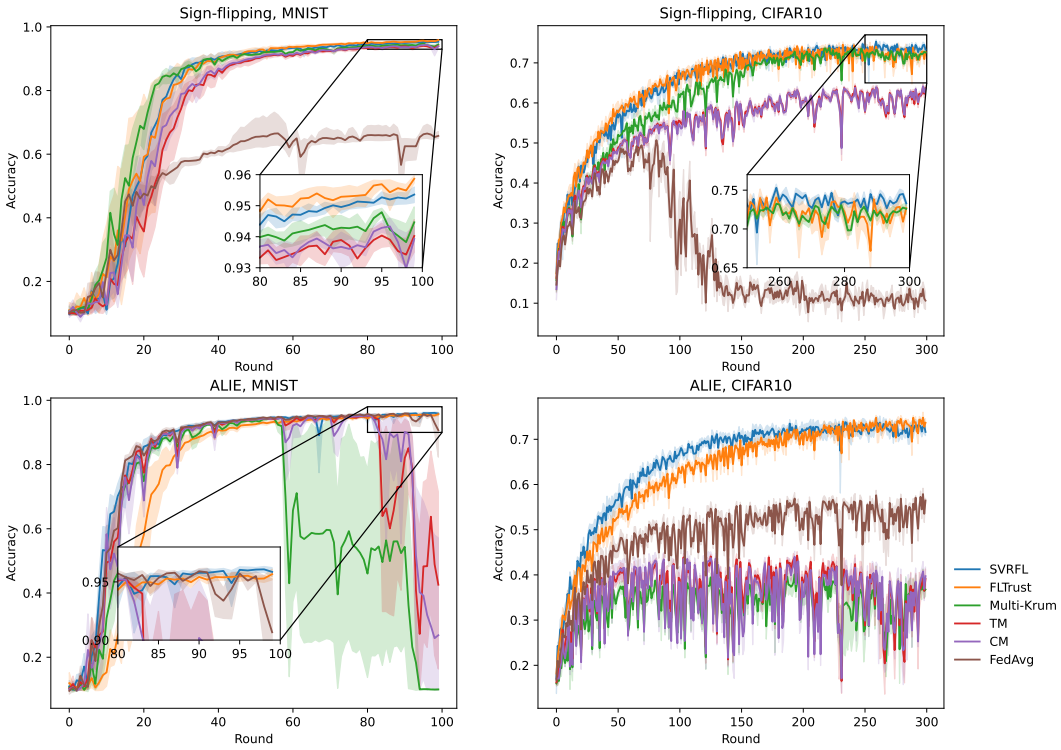

Fig. 5.4 Test accuracy of algorithms in first 100 rounds in MNIST and first 300 rounds in CIFAR10 when adversaries perform sign-flipping and ALIE attack. Curves are averages over 5 random trials, shaded regions represent 95% confidence intervals.

Table 5.4 The average test accuracy[%] (95% confidence intervals in brackets) of the learnt global model with different algorithms under sign-flipping and ALIE attack, and the average ASR[%] and TACC[%] (95% confidence intervals in brackets) achieved by different algorithms under label-flipping attack.

(a) MNIST

	Sign-flipping	ALIE	Label Flipping	
	Test accuracy	Test accuracy	ASR	TACC
FedAvg	65.7(±1.9)	90.7(±11.6)	28.19(±22.14)	69.57(±22.21)
SVRFL	95.4(±0.4)	**95.8(±0.6)**	0.21(±0.59)	97.59(±0.90)
FLTrust	**95.9(±0.3)**	95.7(±0.7)	0.65(±0.87)	97.06(±1.55)
Multi-Krum	94.5(±0.9)	10.0(±0.2)	**0.22(±0.36)**	**97.92(±0.72)**
TM	93.9(±0.9)	42.6(±43.4)	0.99(±0.40)	96.55(±0.64)
CM	94.0(±0.8)	26.9(±41.1)	0.88(±0.59)	96.56(±1.00)

(b) CIFAR10

	Sign-flipping	ALIE	Label Flipping	
	Test accuracy	Test accuracy	ASR	TACC
FedAvg	10.6(±3.4)	56.5(±0.9)	25.0(±5.3)	69.6(±5.3)
SVRFL	**73.2(±0.5)**	71.6(±1.5)	**13.6(±5.3)**	76.8(±5.6)
FLTrust	72.7(±0.7)	**73.7(±1.5)**	14.2(±5.9)	**76.9(±3.9)**
Multi-Krum	72.6(±1.1)	36.8(±2.6)	37.0(±14.7)	59.0(±14.8)
TM	63.6(±0.8)	39.1(±3.6)	19.5(±3.0)	75.5(±3.1)
CM	63.6(±1.0)	39.7(±3.1)	19.7(±1.0)	75.1(±0.5)

tional FedAvg in the partial client participation setting, which is more commonly encountered in practical implementations of FL, thus the comparison with RFFL is eliminated. As shown in Figure 5.4, SVRFL achieves the best performance under the ALIE attack in MNIST and the Sign-flipping attack in CIFAR10. The test accuracy of the finally learnt global model with different algorithms is compared and shown in Table 5.4. As presented, FLTrust achieves the highest test accuracy under the sign-flipping attack in MNIST and the ALIE attack in CIFAR10. Multi-krum achieves the best performance under the label flipping attack in MNIST, while SVRFL and FLTrust show better performance than others in CIFAR10. As the second-best performing algorithm, SVRFL achieves a 95.4% average test accuracy under the sign-flipping attack in MNIST, with a 0.5% reduction compared to the best. Under the label flipping attack in MNIST, the differences in average ASR and TACC between SVRFL and Multi-krum are 0.01% and 0.33%, respectively.

5.5 Summary

This chapter proposes a new byzantine-robust mechanism, SVRFL, which could not only defend against adversarial clients but also stealthy free riders. By exploiting the Shapley value and cosine similarity, SVRFL accurately identifies free riders and expels them from FL system, thereby ensuring fundamental fairness within FL. In the presence of adversarial clients executing poisoning attacks, SVRFL also ensures the reliability of the global model by measuring the model update utility according to its Shapley value. Extensive experiments over MNIST and CIFAR10 datasets demonstrate the effectiveness of SVRFL.

Chapter 6

Conclusion and Future Work

FL has been envisioned to enable collaborative intelligence at the network edge. However, its privacy-preserving design makes it vulnerable to unreliable clients and the Central Server (CS) and poses a communication burden to resourced-constrained edge devices. To mitigate these issues, this thesis focuses on designing defensive mechanisms with communication efficiency to realize secure and efficient federated edge learning. In the following, a short summary of the research and several directions for future work are presented.

6.1 Thesis summary

FL is a recent distributed ML model training paradigm and has shown huge potential in realizing privacy-preserving ML. This privacy preservation feature makes it particularly valuable in fields like healthcare, finance, and e-commerce, where data privacy is paramount. With the broader aim of enhancing the security, reliability, and efficiency of FL, this thesis presented novel mechanisms and theoretical contributions to FL. The major achievements of this thesis are presented as follows:

- In traditional FL, a CS orchestrates the whole learning process, rendering it subject to any malfunction and misbehavior of the CS. To mitigate this issue, Chapter 3 presented an origin blockchain-based FL framework which moves the centralized model aggregation and distribution to the blockchain nodes, who communicate with the participating clients in FL and competitively aggregate pending model updates to generate the updated global model. The correctness of the generated global model is ensured by the dedicated VRF-based consensus protocol which is more energy-efficient than traditional PoW-based consensuses (*i.e.*, the consensus of Bitcoin and Ethereum). To make the blockchain scalable with the blockchain network size, IPFS

has been leveraged into the block design, in which only the IPFS address of the global model is recorded. By doing this, the block size has been greatly reduced, making the communication and storage costs of blockchain blocks affordable to edge nodes. Simulation results have demonstrated the security and the scalability of the proposed blockchain framework.

- Apart from the vulnerabilities posed by the CS, FL is also vulnerable to adversarial clients, who send poisonous model updates to the CS to degrade the global model performance. Considering the communication burden for resource-constrained participating clients, Chapter 4 further proposed a novel blockchain-empowered FL system with a communication-efficient distributed training scheme and a secure aggregation protocol which are embedded in the operating process of the blockchain system. Specifically, the state-of-the-art gradient compression method PowerSGD is utilized to reduce the communication cost of clients. Given the compressed model updates, the mutual information (MI) between clients' model updates is leveraged to build the defense against poisonous ones. In particular, a model update with very low MI and extremely high MI are discarded as a low MI value indicates the intrinsic large difference between this local model and others, and the one with high MI value tends to contribute little to the global model due to little new information it adds to the process. Simulation results show that the proposed system is able to defend against 20% data poisoning and model poisoning attacks with greatly reduced communication cost.

- In real-world FL deployment, aside from adversarial clients and benign clients, free riders who wish to obtain the well-trained global model without any contribution may exist as well. Chapter 5 further explored the defense not only against adversarial clients but also free riders. To distinguish between a free rider and a benign client, Shapley value which has been widely applied in economics and management science for fair evaluation of contribution is leveraged for measuring the contribution of clients. Specifically, a reputation score and a model utility score are computed using the Shapley values of model updates. Based on these scores, the proposed method, SVRFL, ensures the basic fairness within FL by identifying and eliminating free riders, and realizes adversarial robustness through discarding poisonous model updates, with theoretical convergence guaranteed. Extensive experiments show that SVRFL can detect typical free-riding attacks with up to 100% precision and is resistant to poisoning attacks launched by adversarial clients.

6.2 Future work

FL is an evolving research area, it offers default privacy preservation to clients by keeping their data localized. However, recent studies show that even the shared model update parameters, which are essential for the collaborative learning process in FL, can pose potential privacy risks. These risks arise because the model updates, while not directly containing raw data, can still carry implicit information about the underlying data used for training. Sophisticated attackers can employ gradient inversion attacks, to extract sensitive information from these updates [42, 55, 140]. This possibility challenges the fundamental privacy-preserving premise of FL. Most research works try to mitigate this issue with differential privacy by adding carefully crafted noise to model updates [127, 126] or homomorphic encryption to protect the communicated model updates and the aggregated model [9, 28]. These strategies add different amounts of obfuscation to the shared model updates, thus minimizing the success rate for attackers targeting private data recovery. However, it also introduces challenges in distinguishing between malicious or free-riding model updates and those that are benign. Exploring new FL mechanisms to defend against poisoning, free-riding, and gradient inversion attacks can be a promising and important research direction.

In addition to ensuring the reliability of FL amidst the presence of byzantine clients, maintaining the efficiency of the collaborative training process is also crucial. In practical FL deployments, the majority adopts a synchronous setting, where the duration of each training round is as fast as that of the slowest devices. This approach, while straightforward, can lead to inefficiencies, especially in heterogeneous environments where device capabilities and network connections vary significantly. To mitigate this issue, asynchronous FL was proposed, where the aggregation is performed without receiving all the updates from selected clients. However, this lack of synchronization raises concerns about the stability of the global model due to the integration of the outdated model updates [135]. Blockchain might be a promising approach to control the version of the asynchronously aggregated global model since all the aggregated global models could be recorded in the blockchain. Further exploration of blockchain-empowered asynchronous FL could be a promising and interesting direction to realize reliable asynchronous FL.

References

[1] (2017). Federated learning: Collaborative machine learning without centralized training data. https://research.googleblog.com/2017/04/federated-learning-collaborative.html. (Accessed: June 20, 2023).

[2] (2018). Eos.io technical white paper. https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md. (Accessed: September 11, 2020).

[3] (2020). Utilization of fate in risk management of credit in small and micro enterprises. https://www.fedai.org/cases/utilization-of-fate-in-risk-management-of-credit-in-small-and-micro-enterprises/. (Accessed: July 2, 2023).

[4] Albasyoni, A., Safaryan, M., Condat, L., and Richtárik, P. (2020). Optimal gradient compression for distributed and federated learning. *arXiv preprint arXiv:2010.03246*.

[5] Alistarh, D., Grubic, D., Li, J., Tomioka, R., and Vojnovic, M. (2017). Qsgd: Communication-efficient sgd via gradient quantization and encoding. *Advances in neural information processing systems*, 30.

[6] Amiri, M. M., Gündüz, D., Kulkarni, S. R., and Poor, H. V. (2021). Convergence of update aware device scheduling for federated learning at the wireless edge. *IEEE Transactions on Wireless Communications*, 20(6):3643–3658.

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

[8] Ang, F., Chen, L., Zhao, N., Chen, Y., Wang, W., and Yu, F. R. (2020). Robust federated learning with noisy communication. *IEEE Transactions on Communications*, 68(6):3452–3464.

[9] Aono, Y., Hayashi, T., Wang, L., Moriai, S., et al. (2017). Privacy-preserving deep learning via additively homomorphic encryption. *IEEE transactions on information forensics and security*, 13(5):1333–1345.

[10] Apple (2019). Designing for Privacy - WWDC19 - Videos - Apple Developer — developer.apple.com. https://developer.apple.com/videos/play/wwdc2019/708. (Accessed: August 5, 2023).

[11] Baruch, G., Baruch, M., and Goldberg, Y. (2019). A little is enough: Circumventing defenses for distributed learning. *Advances in Neural Information Processing Systems*, 32.

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

[13] Blanchard, P., El Mhamdi, E. M., Guerraoui, R., and Stainer, J. (2017). Machine learning with adversaries: Byzantine tolerant gradient descent. *Advances in neural information processing systems*, 30.

[14] Boenisch, F., Dziedzic, A., Schuster, R., Shamsabadi, A. S., Shumailov, I., and Papernot, N. (2023). When the curious abandon honesty: Federated learning is not private. In *2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P)*, pages 175–199. IEEE.

[15] Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., Kiddon, C., Konečnỳ, J., Mazzocchi, S., McMahan, B., et al. (2019). Towards federated learning at scale: System design. *Proceedings of machine learning and systems*, 1:374–388.

[16] Bonomi, F., Milito, R., Zhu, J., and Addepalli, S. (2012). Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pages 13–16.

[17] Buchman, E. (2016). *Tendermint: Byzantine fault tolerance in the age of blockchains*. PhD thesis, University of Guelph.

[18] Buterin, V. and Griffith, V. (2017). Casper the friendly finality gadget. *arXiv preprint arXiv:1710.09437*.

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

[20] Cao, X., Fang, M., Liu, J., and Gong, N. (2021a). Fltrust: Byzantine-robust federated learning via trust bootstrapping. In *Proceedings of NDSS*.

[21] Cao, X., Jia, J., and Gong, N. Z. (2021b). Provably secure federated learning against malicious clients. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 6885–6893.

[22] Castro, J., Gómez, D., and Tejada, J. (2009). Polynomial calculation of the shapley value based on sampling. *Computers & Operations Research*, 36(5):1726–1730.

[23] Chen, F., Luo, M., Dong, Z., Li, Z., and He, X. (2018a). Federated meta-learning with fast convergence and efficient communication. *arXiv preprint arXiv:1802.07876*.

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

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

[26] Chen, T., Giannakis, G., Sun, T., and Yin, W. (2018b). Lag: Lazily aggregated gradient for communication-efficient distributed learning. *Advances in neural information processing systems*, 31.

[27] Chen, Y., Su, L., and Xu, J. (2017). Distributed statistical machine learning in adversarial settings: Byzantine gradient descent. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 1(2):1–25.

[28] Cheng, K., Fan, T., Jin, Y., Liu, Y., Chen, T., Papadopoulos, D., and Yang, Q. (2021). Secureboost: A lossless federated learning framework. *IEEE Intelligent Systems*, 36(6):87–98.

[29] Churyumov, A. (2016). Byteball: A decentralized system for storage and transfer of value. *URL https://byteball. org/Byteball. pdf*, page 11.

[30] Coniks-Sys (2019). Coniks-sys/coniks-go: A coniks implementation in golang. https://github.com/coniks-sys/coniks-go. (Accessed: August 10, 2020).

[31] Costa, B., Bachiega Jr, J., de Carvalho, L. R., and Araujo, A. P. (2022). Orchestration in fog computing: A comprehensive survey. *ACM Computing Surveys (CSUR)*, 55(2):1–34.

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

[33] DataDog (2018). Datadog/go-python3: Go bindings to the cpython-3 api. https://github.com/DataDog/go-python3. (Accessed: January 11, 2020).

[34] Douceur, J. R. (2002). The sybil attack. In *International workshop on peer-to-peer systems*, pages 251–260. Springer.

[35] Dubey, P. (1982). The shapley value as aircraft landing fees–revisited. *Management Science*, 28(8):869–874.

[36] Eyal, I., Gencer, A. E., Sirer, E. G., and Van Renesse, R. (2016). {Bitcoin-NG}: A scalable blockchain protocol. In *13th USENIX symposium on networked systems design and implementation (NSDI 16)*, pages 45–59.

[37] Feng, L., Zhao, Y., Guo, S., Qiu, X., Li, W., and Yu, P. (2021a). Bafl: A blockchain-based asynchronous federated learning framework. *IEEE Transactions on Computers*, 71(5):1092–1103.

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

[39] Fraboni, Y., Vidal, R., and Lorenzi, M. (2021). Free-rider attacks on model aggregation in federated learning. In *International Conference on Artificial Intelligence and Statistics*, pages 1846–1854. PMLR.

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

[41] Fung, C., Yoon, C. J., and Beschastnikh, I. (2020). The limitations of federated learning in sybil settings. In *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)*, pages 301–316.

[42] Geng, J., Mou, Y., Li, F., Li, Q., Beyan, O., Decker, S., and Rong, C. (2021). Towards general deep leakage in federated learning. *arXiv preprint arXiv:2110.09074*.

[43] Ghorbani, A. and Zou, J. (2019). Data shapley: Equitable valuation of data for machine learning. In *International Conference on Machine Learning*, pages 2242–2251. PMLR.

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

[45] Goudarzi, M., Palaniswami, M., and Buyya, R. (2019). A fog-driven dynamic resource allocation technique in ultra dense femtocell networks. *Journal of Network and Computer Applications*, 145:102407.

[46] Guerraoui, R., Rouault, S., et al. (2018). The hidden vulnerability of distributed learning in byzantium. In *International Conference on Machine Learning*, pages 3521–3530. PMLR.

[47] Gul, F. (1989). Bargaining foundations of shapley value. *Econometrica: Journal of the Econometric Society*, pages 81–95.

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

[49] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

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

[51] Hu, Y. C., Patel, M., Sabella, D., Sprecher, N., and Young, V. (2015). Mobile edge computing—a key technology towards 5g. *ETSI white paper*, 11(11):1–16.

[52] Huang, H., Zhang, B., Sun, Y., Ma, C., and Qu, J. (2022). Delta-dagmm: a free rider attack detection model in horizontal federated learning. *Security and Communication Networks*, 2022.

[53] Ipfs (2017). Ipfs/go-ipfs-api: The go interface to ipfs's http api. https://github.com/ipfs/go-ipfs-api. (Accessed: March 2, 2020).

[54] Jia, R., Dao, D., Wang, B., Hubis, F. A., Hynes, N., Gürel, N. M., Li, B., Zhang, C., Song, D., and Spanos, C. J. (2019). Towards efficient data valuation based on the shapley value. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1167–1176. PMLR.

[55] Jin, X., Chen, P.-Y., Hsu, C.-Y., Yu, C.-M., and Chen, T. (2021). Cafe: Catastrophic data leakage in vertical federated learning. *Advances in Neural Information Processing Systems*, 34:994–1006.

[56] Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Nitin Bhagoji, A., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., et al. (2019). Advances and open problems in federated learning. *arXiv e-prints*, pages arXiv–1912.

[57] Kang, J., Xiong, Z., Niyato, D., Zou, Y., Zhang, Y., and Guizani, M. (2020). Reliable federated learning for mobile networks. *IEEE Wireless Communications*, 27(2):72–80.

[58] Kang, P., Yang, W., and Zheng, J. (2022). Blockchain private file storage-sharing method based on ipfs. *Sensors*, 22(14):5100.

[59] Karimireddy, S. P., He, L., and Jaggi, M. (2021). Learning from history for byzantine robust optimization. In *International Conference on Machine Learning*, pages 5311–5319. PMLR.

[60] Karimireddy, S. P., Kale, S., Mohri, M., Reddi, S., Stich, S., and Suresh, A. T. (2020). Scaffold: Stochastic controlled averaging for federated learning. In *International conference on machine learning*, pages 5132–5143. PMLR.

[61] Karimireddy, S. P., Rebjock, Q., Stich, S., and Jaggi, M. (2019). Error feedback fixes signsgd and other gradient compression schemes. In *International Conference on Machine Learning*, pages 3252–3261. PMLR.

[62] Khaled, A., Mishchenko, K., and Richtárik, P. (2019). First analysis of local gd on heterogeneous data. *arXiv preprint arXiv:1909.04715*.

[63] Khatal, S., Rane, J., Patel, D., Patel, P., and Busnel, Y. (2021). Fileshare: A blockchain and ipfs framework for secure file sharing and data provenance. In *Advances in Machine Learning and Computational Intelligence: Proceedings of ICMLCI 2019*, pages 825–833. Springer.

[64] Koloskova, A., Stich, S., and Jaggi, M. (2019). Decentralized stochastic optimization and gossip algorithms with compressed communication. In *International Conference on Machine Learning*, pages 3478–3487. PMLR.

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

[66] Kumar, R., Tripathi, R., Marchang, N., Srivastava, G., Gadekallu, T. R., and Xiong, N. N. (2021). A secured distributed detection system based on ipfs and blockchain for industrial image and video data security. *Journal of Parallel and Distributed Computing*, 152:128–143.

[67] Kwon, Y. and Zou, J. (2022). Beta shapley: a unified and noise-reduced data valuation framework for machine learning. In *International Conference on Artificial Intelligence and Statistics*, pages 8780–8802. PMLR.

[68] LAMPORT, L., SHOSTAK, R., and PEASE, M. (1982). The byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401.

[69] LeMahieu, C. (2018). Nano: A feeless distributed cryptocurrency network. *Nano [Online resource]. URL: https://nano. org/en/whitepaper*, 16:17.

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

[71] Li, C., Li, P., Zhou, D., Xu, W., Long, F., and Yao, A. (2018a). Scaling nakamoto consensus to thousands of transactions per second. *arXiv preprint arXiv:1805.03870*.

[72] Li, H., Ota, K., and Dong, M. (2018b). Learning iot in edge: Deep learning for the internet of things with edge computing. *IEEE network*, 32(1):96–101.

[73] Li, L., Xu, W., Chen, T., Giannakis, G. B., and Ling, Q. (2019a). Rsa: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1544–1551.

[74] Li, S., Cheng, Y., Wang, W., Liu, Y., and Chen, T. (2020b). Learning to detect malicious clients for robust federated learning. *arXiv preprint arXiv:2002.00211*.

[75] Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., and Smith, V. (2020c). Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2:429–450.

[76] Li, X., Huang, K., Yang, W., Wang, S., and Zhang, Z. (2020d). On the convergence of fedavg on non-iid data. In *International Conference on Learning Representations*.

[77] Li, X., Yang, W., Wang, S., and Zhang, Z. (2019b). Communication-efficient local decentralized sgd methods. *arXiv preprint arXiv:1910.09126*.

[78] Li, Y., Chen, C., Liu, N., Huang, H., Zheng, Z., and Yan, Q. (2020e). A blockchain-based decentralized federated learning framework with committee consensus. *IEEE Network*, 35(1):234–241.

[79] Lian, Z. and Su, C. (2022). Decentralized federated learning for internet of things anomaly detection. In *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, pages 1249–1251.

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

[81] Lin, J., Du, M., and Liu, J. (2019). Free-riders in federated learning: Attacks and defenses. *arXiv preprint arXiv:1911.12560*.

[82] Liu, F., Tang, G., Li, Y., Cai, Z., Zhang, X., and Zhou, T. (2019). A survey on edge computing systems and tools. *Proceedings of the IEEE*, 107(8):1537–1562.

[83] Lu, Y., Huang, X., Dai, Y., Maharjan, S., and Zhang, Y. (2019). Blockchain and federated learning for privacy-preserved data sharing in industrial iot. *IEEE Transactions on Industrial Informatics*, 16(6):4177–4186.

[84] Luan, H. and Tsai, C.-C. (2021). A review of using machine learning approaches for precision education. *Educational Technology & Society*, 24(1):250–266.

[85] Ma, R. T., Chiu, D. M., Lui, J. C., Misra, V., and Rubenstein, D. (2007). Internet economics: The use of shapley value for isp settlement. In *Proceedings of the 2007 ACM CoNEXT conference*, pages 1–12.

[86] Maleki, S., Tran-Thanh, L., Hines, G., Rahwan, T., and Rogers, A. (2013). Bounding the estimation error of sampling-based shapley value approximation. *arXiv preprint arXiv:1306.4265*.

[87] Mao, Y., You, C., Zhang, J., Huang, K., and Letaief, K. B. (2017). A survey on mobile edge computing: The communication perspective. *IEEE communications surveys & tutorials*, 19(4):2322–2358.

[88] McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. (2017). Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR.

[89] MELLODDY (2019). Machine learning ledger orchestration for drug discovery (melloddy). https://www.melloddy.eu/. (Accessed: June 23, 2023).

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

[91] Mills, J., Hu, J., and Min, G. (2020). Communication-efficient federated learning for wireless edge intelligence in iot. *IEEE Internet of Things Journal*, 7(7):5986–5994.

[92] Mills, J., Hu, J., and Min, G. (2023). Faster federated learning with decaying number of local sgd steps. *IEEE Transactions on Parallel and Distributed Systems*, 34(7):2198–2207.

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

[94] Nagalapatti, L. and Narayanam, R. (2021). Game of gradients: Mitigating irrelevant clients in federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 9046–9054.

[95] Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. *Decentralized business review*.

[96] Park, J., Samarakoon, S., Elgabli, A., Kim, J., Bennis, M., Kim, S.-L., and Debbah, M. (2021). Communication-efficient and distributed learning over wireless networks: Principles and applications. *Proceedings of the IEEE*, 109(5):796–819.

[97] Paul, G., Hutchison, F., and Irvine, J. (2014). Security of the maidsafe vault network. In *Wireless World Research Forum Meeting 32 (WWRF32)*.

[98] Pichler, G., Romanelli, M., Vega, L. R., and Piantanida, P. (2023). Perfectly accurate membership inference by a dishonest central server in federated learning. *IEEE Transactions on Dependable and Secure Computing*.

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

[100] Qu, Y., Gao, L., Luan, T. H., Xiang, Y., Yu, S., Li, B., and Zheng, G. (2020a). Decentralized privacy using blockchain-enabled federated learning in fog computing. *IEEE Internet of Things Journal*, 7(6):5171–5183.

[101] Qu, Y., Pokhrel, S. R., Garg, S., Gao, L., and Xiang, Y. (2020b). A blockchained federated learning framework for cognitive computing in industry 4.0 networks. *IEEE Transactions on Industrial Informatics*, 17(4):2964–2973.

[102] Reddi, S., Charles, Z., Zaheer, M., Garrett, Z., Rush, K., Konečnỳ, J., Kumar, S., and McMahan, H. B. (2020). Adaptive federated optimization. *arXiv preprint arXiv:2003.00295*.

[103] Regulation, P. (2016). Regulation (eu) 2016/679 of the european parliament and of the council. *Regulation (eu)*, 679:2016.

[104] Reisizadeh, A., Mokhtari, A., Hassani, H., and Pedarsani, R. (2019). An exact quantized decentralized gradient descent algorithm. *IEEE Transactions on Signal Processing*, 67(19):4934–4947.

[105] Ren, J., Zhang, D., He, S., Zhang, Y., and Li, T. (2019). A survey on end-edge-cloud orchestrated network computing paradigms: Transparent computing, mobile edge computing, fog computing, and cloudlet. *ACM Computing Surveys (CSUR)*, 52(6):1–36.

[106] Salkuti, S. R. (2020). A survey of big data and machine learning. *International Journal of Electrical & Computer Engineering (2088-8708)*, 10(1).

[107] Sattler, F., Wiedemann, S., Müller, K.-R., and Samek, W. (2019). Robust and communication-efficient federated learning from non-iid data. *IEEE transactions on neural networks and learning systems*, 31(9):3400–3413.

[108] Shapley, L. S. et al. (1953). A value for n-person games.

[109] Shayan, M., Fung, C., Yoon, C. J., and Beschastnikh, I. (2020). Biscotti: A blockchain system for private and secure federated learning. *IEEE Transactions on Parallel and Distributed Systems*, 32(7):1513–1525.

[110] Shi, W., Cao, J., Zhang, Q., Li, Y., and Xu, L. (2016). Edge computing: Vision and challenges. *IEEE internet of things journal*, 3(5):637–646.

[111] Shi, W. and Dustdar, S. (2016). The promise of edge computing. *Computer*, 49(5):78–81.

[112] Sompolinsky, Y., Lewenberg, Y., and Zohar, A. (2016). Spectre: A fast and scalable cryptocurrency protocol. *Cryptology ePrint Archive*.

[113] Sompolinsky, Y. and Zohar, A. (2015). Secure high-rate transaction processing in bitcoin. In *Financial Cryptography and Data Security: 19th International Conference, FC 2015, San Juan, Puerto Rico, January 26-30, 2015, Revised Selected Papers 19*, pages 507–527. Springer.

[114] Sompolinsky, Y. and Zohar, A. (2018). Phantom. *IACR Cryptology ePrint Archive, Report 2018/104*.

[115] Taleb, T., Samdanis, K., Mada, B., Flinck, H., Dutta, S., and Sabella, D. (2017). On multi-access edge computing: A survey of the emerging 5g network edge cloud architecture and orchestration. *IEEE Communications Surveys & Tutorials*, 19(3):1657–1681.

[116] Uddin, M. P., Xiang, Y., Lu, X., Yearwood, J., and Gao, L. (2020). Mutual information driven federated learning. *IEEE Transactions on Parallel and Distributed Systems*, 32(7):1526–1538.

[117] Vershynin, R. (2010). Introduction to the non-asymptotic analysis of random matrices. *arXiv preprint arXiv:1011.3027*.

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

[119] Wadu, M. M., Samarakoon, S., and Bennis, M. (2020). Federated learning under channel uncertainty: Joint client scheduling and resource allocation. In *2020 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6.

[120] Wang, G., Dang, C. X., and Zhou, Z. (2019a). Measure contribution of participants in federated learning. In *2019 IEEE international conference on big data (Big Data)*, pages 2597–2604. IEEE.

[121] Wang, J., Chang, X., Mišić, J., Mišić, V. B., and Wang, Y. (2023). Pass: A parameter audit-based secure and fair federated learning scheme against free-rider attack. *IEEE Internet of Things Journal*.

[122] Wang, J., Zhang, L., Li, A., You, X., and Cheng, H. (2022). Efficient participant contribution evaluation for horizontal and vertical federated learning. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pages 911–923. IEEE.

[123] Wang, S., Tuor, T., Salonidis, T., Leung, K. K., Makaya, C., He, T., and Chan, K. (2019b). Adaptive federated learning in resource constrained edge computing systems. *IEEE Journal on Selected Areas in Communications*, 37(6):1205–1221.

[124] Wang, T., Rausch, J., Zhang, C., Jia, R., and Song, D. (2020). A principled approach to data valuation for federated learning. *Federated Learning: Privacy and Incentive*, pages 153–167.

[125] Wang, Z. and Hu, Q. (2021). Blockchain-based federated learning: A comprehensive survey. *arXiv preprint arXiv:2110.02182*.

[126] Wei, K., Li, J., Ding, M., Ma, C., Yang, H. H., Farokhi, F., Jin, S., Quek, T. Q., and Poor, H. V. (2020). Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security*, 15:3454–3469.

[127] Wei, W., Liu, L., Wut, Y., Su, G., and Iyengar, A. (2021). Gradient-leakage resilient federated learning. In *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*, pages 797–807. IEEE.

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

[129] Wu, H. and Wang, P. (2022). Node selection toward faster convergence for federated learning on non-iid data. *IEEE Transactions on Network Science and Engineering*, 9(5):3099–3111.

[130] Xiao, P., Cheng, S., Stankovic, V., and Vukobratovic, D. (2020a). Averaging is probably not the optimum way of aggregating parameters in federated learning. *Entropy*, 22(3):314.

[131] Xiao, Y., Zhang, N., Lou, W., and Hou, Y. T. (2020b). A survey of distributed consensus protocols for blockchain networks. *IEEE Communications Surveys & Tutorials*, 22(2):1432–1465.

[132] Xie, C., Chen, M., Chen, P.-Y., and Li, B. (2021). Crfl: Certifiably robust federated learning against backdoor attacks. In Meila, M. and Zhang, T., editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 11372–11382. PMLR.

[133] Xie, C., Koyejo, O., and Gupta, I. (2020). Fall of empires: Breaking byzantine-tolerant sgd by inner product manipulation. In *Uncertainty in Artificial Intelligence*, pages 261–270. PMLR.

[134] Xie, C., Koyejo, O., Gupta, I., and Lin, H. (2019). Local adaalter: Communication-efficient stochastic gradient descent with adaptive learning rates. *arXiv preprint arXiv:1911.09030*.

[135] Xu, C., Qu, Y., Luan, T. H., Eklund, P. W., Xiang, Y., and Gao, L. (2022a). An efficient and reliable asynchronous federated learning scheme for smart public transportation. *IEEE Transactions on Vehicular Technology*.

[136] Xu, J., Huang, S.-L., Song, L., and Lan, T. (2022b). Byzantine-robust federated learning through collaborative malicious gradient filtering. In *2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS)*, pages 1223–1235.

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

[138] Xu, X., Liu, Q., Luo, Y., Peng, K., Zhang, X., Meng, S., and Qi, L. (2019). A computation offloading method over big data for iot-enabled cloud-edge computing. *Future Generation Computer Systems*, 95:522–533.

[139] Xu, X. and Lyu, L. (2021). A reputation mechanism is all you need: Collaborative fairness and adversarial robustness in federated learning.

[140] Yang, H., Ge, M., Xue, D., Xiang, K., Li, H., and Lu, R. (2023). Gradient leakage attacks in federated learning: Research frontiers, taxonomy and future directions. *IEEE Network*.

[141] Yang, H. H., Liu, Z., Quek, T. Q., and Poor, H. V. (2019). Scheduling policies for federated learning in wireless networks. *IEEE transactions on communications*, 68(1):317–333.

[142] Yin, D., Chen, Y., Kannan, R., and Bartlett, P. (2018). Byzantine-robust distributed learning: Towards optimal statistical rates. In *International Conference on Machine Learning*, pages 5650–5659. PMLR.

[143] Yu, H., Jin, R., and Yang, S. (2019a). On the linear speedup analysis of communication efficient momentum SGD for distributed non-convex optimization. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 7184–7193. PMLR.

[144] Yu, H., Yang, S., and Zhu, S. (2019b). Parallel restarted sgd with faster convergence and less communication: Demystifying why model averaging works for deep learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5693–5700.

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

[146] Zhang, Z., Cao, X., Jia, J., and Gong, N. Z. (2022). Fldetector: Defending federated learning against model poisoning attacks via detecting malicious clients. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, page 2545–2555, New York, NY, USA. Association for Computing Machinery.

[147] Zhao, Y., Zhao, J., Jiang, L., Tan, R., Niyato, D., Li, Z., Lyu, L., and Liu, Y. (2020). Privacy-preserving blockchain-based federated learning for iot devices. *IEEE Internet of Things Journal*, 8(3):1817–1829.

[148] Zheng, Q., Li, Y., Chen, P., and Dong, X. (2018). An innovative ipfs-based storage model for blockchain. In *2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, pages 704–708.

Appendix A

Chapter 5 Supplementary Material

A.1 Proof of Theorem 1

Before proving Theorem 1, we first restate our SVRFL algorithm and prove some lemmas. Recall that the global model is updated with equation (5.7), and the aggregated global gradient is $\hat{g}^t = \frac{1}{\sum_{i\in I_t}^{|I_t|} s(u_i)} \sum_{i\in I_t}^{|I_t|} s(u_i) g_i^t$, where I_t ($|I_t| = m$) is the set of selected clients in round t, and $g_i = \nabla F(\xi_i, w_g^{t-1})$ is the local model gradient from client i. Then the global model update could be rewritten as $w_g^t = w_g^{t-1} - \eta \hat{g}^t$. When βn adversarial clients exist in FL system, we denote by G ($\mathbb{E}|G| = (1-\beta)m$) the set of benign clients in a training round. The averaged benign gradients can be expressed by $\bar{g}^t = \frac{1}{|G|} \sum_{i\in G} g_i^t$.

Lemma 1. *Suppose Assumption 1 holds, we have the following in any training round $t \geq 1$:*

$$\mathbb{E}[\|w^{t-1} - w^* - \eta \nabla F(w^{t-1})\|^2] \leq \quad (1 + \eta^2 L^2 - \eta\mu)\mathbb{E}[\|w^{t-1} - w^*\|^2]. \tag{A.1}$$

$$\|w^{t-1} - w^* - \eta \nabla F(w^{t-1})\| \leq \quad \sqrt{1 - \frac{\mu^2}{4L^2}} \|w^{t-1} - w^*\|^2. \tag{A.2}$$

Proof. Since $\nabla F(w^*) = 0$, we have

$$\begin{aligned} &\|w^{t-1} - w^* - \eta \nabla F(w^{t-1})\|^2 \\ &= \|w^{t-1} - w^* - \eta(\nabla F(w^{t-1}) - \nabla F(w^*))\|^2 \\ &= \|w^{t-1} - w^*\|^2 + \eta \|\nabla F(w^{t-1}) - \nabla F(w^*)\|^2 - \\ &2\eta \langle w^{t-1} - w^*, \nabla F(w^{t-1}) - \nabla F(w^*) \rangle \end{aligned}. \tag{A.3}$$

$$\begin{aligned}&\mathbb{E}[\|w^{t-1}-w^*-\eta\nabla F(w^{t-1})\|^2]\\&=\mathbb{E}[\|w^{t-1}-w^*\|^2+\eta\|\nabla F(w^{t-1})-\nabla F(w^*)\|^2-.\\&2\eta\langle w^{t-1}-w^*,\nabla F(w^{t-1})-\nabla F(w^*)\rangle]\end{aligned} \tag{A.4}$$

According to Assumption 1, we have

$$\|\nabla F(w)-\nabla F(w^*)\|\leq L\|w-w^*\|, \tag{A.5}$$

$$F(w)\geq F(w^*)+\langle\nabla F(w^*),w-w^*\rangle+\frac{\mu}{2}\|w-w^*\|^2, \tag{A.6}$$

$$F(w^*)\geq F(w)+\langle\nabla F(w),w^*-w\rangle. \tag{A.7}$$

Summing up inequalities (A.6) and (A.7), we could have

$$\langle\nabla F(w)-\nabla F(w^*),w^*-w\rangle\leq-\frac{\mu}{2}\|w-w^*\|^2. \tag{A.8}$$

Substituting inequalities (A.5) and (A.8) into equation (A.4), according to linearity and monotonicity of expectation,

$$\begin{aligned}&\mathbb{E}[\|w^{t-1}-w^*-\eta\nabla F(w^{t-1})\|^2]\leq\\&(1+\eta^2L^2-\eta\mu)\mathbb{E}[\|w^{t-1}-w^*\|^2]\end{aligned}. \tag{A.9}$$

Substituting inequalities (A.5) and (A.8) into equation (A.3), we have:

$$\|w^{t-1}-w^*-\eta\nabla F(w^{t-1})\|^2\leq(1+\eta^2L^2-\eta\mu)\|w^{t-1}-w^*\| \tag{A.10}$$

By choosing $\eta=\frac{\mu}{2L^2}$, we conclude the proof.

Suppose Assumptions 2-4 hold, we have the following lemma for the aggregated gradient using SVRFL.

Lemma 2. *The deviation between the aggregated gradient $\hat{g}^t$ and the true global gradient $\nabla F(w^t)$ could be characterized as follows:*

$$\begin{aligned}\mathbb{E}[\|\hat{g}^t-\nabla F(w^t)\|^2]\leq 4c\delta(\sigma^2+\tau^2)+2b^2+\frac{\beta^2\tau^2}{(1-\beta)^2}+\\\frac{\sigma^2}{(1-\beta)m}\end{aligned} \tag{A.11}$$

Proof. Given a set of selected clients in a training round, let $A=\sum_{i\in G}(g_i-\nabla F(w))$ and $B=\sum_{j\notin G}(g_i-\nabla F(w))$, then A and B are independent, thus $\mathbb{E}[A+B]=0$. According to

Assumption 2 and 3, by applying Jensen's inequality, we can have

$$\begin{aligned} \|\mathbb{E}[A]\|^2 &= \left\| \mathbb{E}\left[\sum_{i \in G} (g_i - \nabla F(w)) \right] \right\|^2 \\ &\leq \mathbb{E}\left[\left\| \sum_{i \in G} (g_i - \nabla F(w)) \right\|^2 \right]. \end{aligned} \tag{A.12}$$

$$\leq (1-\beta) m \sum_{i \in G} \|\nabla F_i(w) - \nabla F(w)\|^2$$

$$\leq (1-\beta)^2 m^2 \tau^2$$

$$\|\mathbb{E}[B]\|^2 \leq \beta m \sum_{i \notin G} \|\nabla F_i(w) - \nabla F(w)\|^2 \leq \beta^2 m^2 \tau^2. \tag{A.13}$$

As $\mathbb{E}[A] = -\mathbb{E}[A]$, we could obtain

$$\|\mathbb{E}[A]\|^2 = \|\mathbb{E}[B]\|^2 = \min\{(1-\beta)^2 m^2 \tau^2, \beta^2 m^2 \tau^2\}. \tag{A.14}$$

Since $\beta < 0.5$, $\|\mathbb{E}[A]\|^2 = \|\mathbb{E}[B]\|^2 = \beta^2 m^2 \tau^2$. According to the basic relation between expectation and variance,

$$\mathbb{E}[\|A\|^2] = \|\mathbb{E}[A]\|^2 + \mathrm{var}[A] \leq \beta^2 m^2 \tau^2 + (1-\beta) m \sigma^2. \tag{A.15}$$

Then, we could have

$$\begin{aligned} \mathbb{E}[\|\bar{g} - \nabla F(w)\|^2] &= \mathbb{E}\left[\left\| \frac{1}{|G|} \sum_{i \in G} g_i - \nabla F(w) \right\|^2 \right] \\ &= \frac{1}{(1-\beta)^2 m^2} \mathbb{E}[\|A\|^2] \\ &\leq \frac{\beta^2 \tau^2}{(1-\beta)^2} + \frac{\sigma^2}{(1-\beta) m} \end{aligned} \tag{A.16}$$

Given the above inequality, we have

$$\begin{aligned}
&\mathbb{E}[\|\hat{g}^t - \nabla F(w^t)\|^2] \\
&= \mathbb{E}[\|\hat{g}^t - \bar{g}^t + \bar{g}^t - \nabla F(w^t)\|^2] \\
&\leq \mathbb{E}[2\|\hat{g}^t - \bar{g}^t\|^2 + 2\|\bar{g}^t - \nabla F(w^t)\|^2] \\
&= 2\mathbb{E}[\|\hat{g}^t - \bar{g}^t\|^2] + 2\mathbb{E}[\|\bar{g}^t - \nabla F(w^t)\|^2] \\
&= 2[\mathbb{E}\|\hat{g}^t - \bar{g}^t\|]^2 + 2\text{var}(\hat{g}^t) + 2\mathbb{E}[\|\bar{g}^t - \nabla F(w^t)\|^2] \\
&\leq 2c\delta \sup_{i,j\in G} \mathbb{E}[\|g_i^t - g_j^t\|^2] + 2\text{var}(\hat{g}^t) + 2\mathbb{E}[\|\bar{g}^t - \nabla F(w^t)\|^2] \\
&\leq 4c\delta(\sigma^2 + \tau^2) + 2b^2 + \frac{\beta^2\tau^2}{(1-\beta)^2} + \frac{\sigma^2}{(1-\beta)m}
\end{aligned} \quad , \tag{A.17}$$

Thus we complete the proof.

Proof of Theorem 1. With the lemmas above, we prove Theorem 1 next. We have the following for the t^{th} training round:

$$\begin{aligned}
&\mathbb{E}[\|w^t - w^*\|^2] \\
&= \mathbb{E}[\|w^{t-1} - \eta\hat{g}^{t-1} - w^*\|^2] \\
&= \mathbb{E}[\|w^{t-1} - \eta\nabla F(w^{t-1}) - w^* + \eta\nabla F(w^{t-1}) - \eta\hat{g}^{t-1}\|^2] \\
&\leq \mathbb{E}[2\|w^{t-1} - \eta\nabla F(w^{t-1}) - w^*\|^2 + \\
&2\eta^2\|\nabla F(w^{t-1}) - \hat{g}^{t-1}\|^2] \\
&= 2\underbrace{\mathbb{E}[\|w^{t-1} - \eta\nabla F(w^{t-1}) - w^*\|^2]}_{C_1} + \\
&2\eta^2\underbrace{\mathbb{E}[\|\hat{g}^{t-1} - \nabla F(w^{t-1})\|^2]}_{C_2} \\
&\overset{(a)}{\leq} 2(1 + \eta^2L^2 - \eta\mu)\mathbb{E}[\|w^{t-1} - w^*\|^2] + \\
&\underbrace{\eta^2[8c\delta(\sigma^2 + \tau^2) + 4b^2 + \frac{2\beta^2\tau^2}{(1-\beta)^2} + \frac{2\sigma^2}{(1-\beta)m}]}_{\Delta_1} \\
&\leq 2(1 + \eta^2L^2 - \eta\mu)\mathbb{E}[\|w^{t-1} - w^*\|^2] + \eta^2\Delta_1
\end{aligned} \quad . \tag{A.18}$$

where (a) is obtained by plugging Lemma 1 and Lemma 2 into $\mathbf{C_1}$ and $\mathbf{C_2}$ respectively. By recursively applying the above inequality in each global iteration, we have:

$$\mathbb{E}[\|w^t - w^*\|^2] \leq (1-q)^t\mathbb{E}[\|w^0 - w^*\|^2] + \frac{\eta^2\Delta_1}{q} \tag{A.19}$$

where $q = 2\eta\mu - 2\eta^2L^2 - 1$. If $0 < 1-q < 1$, then $0 < 2 - 2\eta\mu + 2\eta^2L^2 < 1$, we could obtain

$$\sqrt{\frac{1}{2L^2} + \frac{\mu^2}{4L^4}} + \frac{\mu}{2L^2} < \eta < \sqrt{\frac{1}{L^2} + \frac{\mu^2}{4L^4}} + \frac{\mu}{2L^2}. \tag{A.20}$$

Thus, if the server-side learning rate η satisfies lies in $(\sqrt{\frac{1}{2L^2} + \frac{\mu^2}{4L^4}} + \frac{\mu}{2L^2}, \sqrt{\frac{1}{L^2} + \frac{\mu^2}{4L^4}} + \frac{\mu}{2L^2})$, we could have

$$\lim_{t\to\infty} \mathbb{E}[\|w^t - w^*\|^2] \leq \frac{\eta^2\Delta_1}{q}, \tag{A.21}$$

which finishes the proof.

A.2 Proof of Theorem 2

Recall that the global model is updated with equation (5.7), let $\mathscr{G}(w_g) = \frac{1}{\sum_{i\in S}^{|S|} s(u_i)} \sum_{i\in S}^{|S|} s(u_i) g_i$ and $q = \sum_{i\in S}^{|S|} s(u_i)$, where $S \triangleq \{I_t | \exists i \in I_t, u_i > 0\}$ and $g_i = \nabla f(D_i, w_g)$. Then q is the positive integer that $1 \leq q \leq m \leq n$, and the global model update could be rewrited as $w_g^t = w_g^{t-1} - \eta\mathscr{G}(w_g^{t-1})$.

Lemma 2. *Suppose Assumption 6 and 8 holds, for any $\delta \in (0,1)$, let $K = \sqrt{2(d\log 6 + \log\frac{3}{\delta})}$,*

$$\Delta_1 = \begin{cases} \sqrt{2}\sigma_1\sqrt{d\log 6 + \log\frac{3}{\delta}}, & \text{if } K \leq \frac{\sigma_1}{\alpha_1} \\ 2\alpha_1(d\log 6 + \log\frac{3}{\delta}), & \text{otherwise} \end{cases}, \tag{A.22}$$

where d is the dimension of w, then,

$$\mathbb{P}\{\|\mathscr{G}(w^*) - \nabla F(w^*)\| \geq 2\Delta_1\} \leq \frac{\delta}{3} \tag{A.23}$$

Proof Sketch. Let $V = \{v_1, \cdots, v_{N_{\frac{1}{2}}}\}$ denote an $\frac{1}{2}$-cover of unit sphere B. As shown in [27] that $\log N_{\frac{1}{2}} \leq d\log 6$, and

$$\|\mathscr{G}(w^*) - \nabla F(w^*)\| \leq 2\sup_{v\in V}\{\langle \mathscr{G}(w^*) - \nabla F(w^*), v\rangle\} \tag{A.24}$$

As $\nabla F(w^*) = 0$, by Assumption 6 and the concentration inequalities for sub-exponential random variables, for $v \in V$,

$$\begin{aligned}\mathbb{P}\{\langle \mathscr{G}(w^*) - \nabla F(w^*), v\rangle \geq \Delta_1\} &\leq \begin{cases} e^{-\frac{q\Delta_1^2}{2\sigma_1^2}}, & \text{if } K \leq \frac{\sigma_1}{\alpha_1} \\ e^{-\frac{q\Delta_1}{2\alpha_1}}, & \text{otherwise} \end{cases} \\ &\leq \begin{cases} e^{-\frac{\Delta_1^2}{2\sigma_1^2}}, & \text{if } K \leq \frac{\sigma_1}{\alpha_1} \\ e^{-\frac{\Delta_1}{2\alpha_1}}, & \text{otherwise} \end{cases}. \end{aligned} \tag{A.25}$$

Recall that V contains at most 6^d vectors. In view of the union bound, it further yields that

$$\begin{aligned}&\mathbb{P}\{2\sup_{v\in V}\{\langle \mathscr{G}(w^*) - \nabla F(w^*), v\rangle\} \geq 2\Delta_1\} \\ &\leq \begin{cases} e^{-\frac{\Delta_1^2}{2\sigma_1^2} + d\log 6}, & \text{if } K \leq \frac{\sigma_1}{\alpha_1} \\ e^{-\frac{\Delta_1}{2\alpha_1} + d\log 6}, & \text{otherwise} \end{cases}. \end{aligned} \tag{A.26}$$

Combining it with inequality (A.24), we have

$$\begin{aligned}&\mathbb{P}\{\|\mathscr{G}(w^*) - \nabla F(w^*)\| \geq 2\Delta_1\} \\ &\leq \begin{cases} e^{-\frac{\Delta_1^2}{2\sigma_1^2} + d\log 6}, & \text{if } K \leq \frac{\sigma_1}{\alpha_1} \\ e^{-\frac{\Delta_1}{2\alpha_1} + d\log 6}, & \text{otherwise} \end{cases}. \end{aligned} \tag{A.27}$$

By choosing Δ_1 as stated in equation (A.22), we conclude the proof.

Lemma 3. *Suppose Assumption 7 holds, for any $w \in \Theta$ and any $\delta \in (0,1)$, let $K' = \sqrt{2(d\log 6 + \log\frac{3}{\delta})}$,*

$$\Delta_2 = \begin{cases} \sqrt{2}\sigma_2\sqrt{d\log 6 + \log\frac{3}{\delta}}, & \text{if } K' \leq \frac{\sigma_2}{\alpha_2} \\ 2\alpha_2(d\log 6 + \log\frac{3}{\delta}), & \text{otherwise} \end{cases}, \tag{A.28}$$

then

$$\mathbb{P}\left\{ \frac{\left\| \frac{1}{\sum_{i\in S}^{|S|} s(u_i)} \sum_{i\in S}^{|S|} s(u_i)h(D_i, w) - \mathbb{E}[h(D,w)] \right\|}{\|w - w^*\|} \geq 2\Delta_2 \right\} \leq \frac{\delta}{3} \tag{A.29}$$

The proof of Lemma 3 is similar to Lemma 2, we omit it for brevity.

Proposition 1. *Suppose all assumptions hold, and* $\Theta \subset \{w : \|w - w^*\| \leq r\sqrt{d}\}$ *for some positive parameter r. For any* $\delta \in (0,1)$ *and any integer n, we have*

$$\begin{aligned} &\mathbb{P}\{\forall w \in \Theta : \|\mathscr{G}(w) - \nabla F(w)\| \leq 8\Delta_3\|w - w^*\| + 2\Delta_1 + 4\Delta_3\tau\} \\ &\geq 1 - \delta \end{aligned}, \tag{A.30}$$

where $\tau = \frac{\alpha_2\sigma_1}{2\sigma_2^2}\sqrt{\frac{d}{n}}$,

$$\Delta_3 = \begin{cases} \sqrt{2}\sigma_2\sqrt{d\log\frac{18K_1}{K_2} + \frac{1}{2}d\log\frac{n}{d} + \log\frac{6\sigma_2^2 r\sqrt{n}}{\alpha_2\sigma_1\delta}}, & \text{if } K_3 \leq K_4 \\ 2\alpha_2(d\log\frac{18K_1}{K_2} + \frac{1}{2}d\log\frac{n}{d} + \log\frac{6\sigma_2^2 r\sqrt{n}}{\alpha_2\sigma_1\delta}), & \text{otherwise} \end{cases}, \tag{A.31}$$

$K_1 = \max(L, L_1)$, $K_2 = \min(\alpha_2, \sigma_2)$, $K_3 = d\log(18K_1) + \frac{1}{2}d\log\frac{n}{d} + \log\frac{6r\sqrt{n}}{\sigma_1,\delta}$, *and* $K_4 = d\log K_2 - \log\frac{\delta_2^2}{\alpha_2} + \frac{\sigma_2^2}{2\alpha_2^2}$.

Proof Sketch. The proof is based on the classic ε-net argument. Let $\ell^* = \lceil\frac{r\sqrt{d}}{\tau}\rceil$. We assume ℓ^* is an integer. For integers $1 \leq \ell \leq \ell^*$, define $\Theta_\ell \triangleq \{w : \|w - w^*\| \leq \tau\ell\}$. For a given ℓ, let $w_1, \cdots, w_{N_{\varepsilon_\ell}}$ be an ε_ℓ-cover of Θ_ℓ, where $\varepsilon_\ell = \frac{K_2\tau\ell}{K_1}\sqrt{\frac{d}{n}}$. According to [117], $\log N_{\varepsilon_\ell} \leq d\log(\frac{3\tau\ell}{\varepsilon_\ell})$. Fix any $w \in \Theta_\ell$. there exists a j_ℓ ($1 \leq j_\ell \leq N_{\varepsilon_\ell}$), such that

$$\|w - w_{j_\ell}\|_2 \leq \varepsilon_\ell. \tag{A.32}$$

By triangle inequality, we have:

$$\begin{aligned} \|\mathscr{G}(w) - \nabla F(w)\| \leq \|\nabla F(w) - \nabla F(w_{j_\ell})\| + \|\mathscr{G}(w) - \mathscr{G}(w_{j_\ell})\| \\ + \|\mathscr{G}(w_{j_\ell}) - \nabla F(w_{j_\ell})\| \end{aligned}. \tag{A.33}$$

According to Assumption 1 and inequality (A.32),

$$\|\nabla F(w) - \nabla F(w_{j_\ell})\| \leq L\|w - w_{j_\ell}\| \leq L\varepsilon_\ell. \tag{A.34}$$

Define event

$$\varepsilon_1 = \left\{ \sup_{w,w' \in \Theta : w \neq w'} \frac{\|\mathscr{G}(w) - \mathscr{G}(w')\|}{\|w - w'\|} \leq L_1 \right\}. \tag{A.35}$$

By Assumption 2, we have $\mathbb{P}\{\varepsilon_1\} \geq 1 - \frac{\delta}{3}$. On event ε_1, we have

$$\sup_{w \in \Theta} \|\mathscr{G}(w) - \mathscr{G}(w_{j_\ell})\| \leq L_1 \varepsilon_\ell. \tag{A.36}$$

According to triangle inequality,

$$\begin{aligned} \|\mathscr{G}(w_{j_\ell}) - \nabla F(w_{j_\ell})\| \leq & \|\mathscr{G})w^*) - \nabla F(w^*)\| + \\ & \|\mathscr{G}(w_{j_\ell}) - \mathscr{G}(w^*) - (\nabla F(w_{j_\ell}) - \nabla F(w^*)\| \\ & \overset{(a)}{\leq} \|\mathscr{G})w^*) - \nabla F(w^*)\| + \\ & \left\| \frac{1}{q} \sum_{i \in S}^{|S|} s(u_i) h(D_i, w_{j_\ell}) - \mathbb{E}[h(D, w_{j_\ell})] \right\| \end{aligned}, \tag{A.37}$$

where (a) is according to $\mathbb{E}[h(D, w_{j_\ell})] = \nabla F(w) - \nabla F(w^*)$. Define event ε_2 and $\mathscr{F}_\ell$ as,

$$\varepsilon_2 = \{\|\mathscr{G}(w^*) - \nabla F(w^*)\| \leq 2\Delta_1\}, \tag{A.38}$$

$$\mathscr{F}_\ell = \left\{ \sup_{1 \leq j \leq N_\varepsilon} \left\| \frac{1}{q} \sum_{i \in S}^{|S|} s(u_i) h(D_i, w_{j_\ell}) - \mathbb{E}[h(D, w_{j_\ell})] \right\| \leq 2\tau\ell\Delta_3 \right\}. \tag{A.39}$$

According to Lemma 3 and [117], $\mathbb{P}\{\varepsilon_2\} \geq 1 - \frac{\delta}{3}$ and $\mathbb{P}\{\mathscr{F}_\ell\} \geq 1 - \frac{\delta}{3\ell^*}$. Therefore, on event $\varepsilon_1 \cap \varepsilon_2 \cap \mathscr{F}_\ell$, we have

$$\begin{aligned} \sup_{w \in \Theta_\ell} \|\mathscr{G}(w) - \nabla F(w)\| &\leq (L + L_1)\varepsilon_\ell + 2\Delta_1 + 2\Delta_3\tau\ell \\ &\overset{(a)}{\leq} 4\Delta_3\tau\ell + 2\Delta_1 \end{aligned} \tag{A.40}$$

where (a) is according to $(L + L_1)\varepsilon_\ell \leq 2K_1\varepsilon_\ell \leq \Delta_3\tau\ell$. Let event $\varepsilon = \varepsilon_1 \cap \varepsilon_2 \cap (\cap_{\ell=1}^{\ell^*} \mathscr{F}_\ell)$. Follows the union bound, we have $\mathbb{P}\{\varepsilon\} = 1 - \delta$. Suppose event ε holds, then for all $w \in \Theta_{\ell^*}$, there exists an ℓ ($1 \leq \ell \leq \ell^*$) such that $(\ell - 1)\tau \leq \|w - w^*\| \leq \ell\tau$. If $\ell \geq 2$, then $\ell \leq 2(\ell - 1)$, thus

$$\|\mathscr{G}(w) - \nabla F(w)\| \leq 4\Delta_3\tau\ell + 2\Delta_1 \leq 8\Delta_3\|w - w^*\| + 2\Delta_1. \tag{A.41}$$

If $\ell = 1$, then

$$\|\mathscr{G}(w) - \nabla F(w)\| \leq 4\Delta_3\tau + 2\Delta_1. \tag{A.42}$$

Combining inequality (A.41) and (A.42), we have:

$$\sup_{w \in \Theta_{\ell^*}} \|\mathscr{G}(w) - \nabla F(w)\| \leq 8\Delta_3\|w - w^*\| + 2\Delta_1 + 4\Delta_3\tau. \tag{A.43}$$

We conclude the proof since $\Theta \subset \Theta_{\ell^*}$.

Proof of Theorem 2. With the proposition and lemmas above, we prove Theorem 2 next. We have the following for the t^{th} training round:

$$\begin{aligned}
&\|w^t - w^*\| \\
&= \|w^{t-1} - \eta \mathscr{G}(w^{t-1}) - w^*\| \\
&= \|w^{t-1} - \eta \nabla F(w^{t-1}) - w^* + \eta \nabla F(w^{t-1}) - \eta \mathscr{G}(w^{t-1})\| \\
&\leq \underbrace{\|w^{t-1} - \eta \nabla F(w^{t-1}) - w^*\|}_{C_1} + \underbrace{\eta \|\mathscr{G}(w^{t-1}) - \nabla F(w^{t-1}\|}_{C_2} \qquad . \qquad \text{(A.44)} \\
&\overset{(a)}{\leq} \sqrt{1 - \frac{\mu^2}{4L^2}} \|w^{t-1} - w^*\| + \eta(8\Delta_3 \|w^{t-1} - w^*\| + 2\Delta_1 + 4\Delta_3 \tau) \\
&= (\sqrt{1 - \frac{\mu^2}{4L^2}} + 8\Delta_3 \eta) \|w^{t-1} - w^*\| + 2\Delta_1 \eta + 4\Delta_3 \tau \eta
\end{aligned}$$

where (a) is obtained by plugging Lemma 1 and Proposition 1 into $\mathbf{C_1}$ and $\mathbf{C_2}$ respectively. By recursively applying the above inequality in each global iteration, we have:

$$\|w^t - w^*\| \leq (1-p)^t \|w^0 - w^*\|^2 + \frac{2\Delta_1 \eta + 4\Delta_3 \tau \eta}{p} \qquad \text{(A.45)}$$

where $p = 1 - (\sqrt{1 - \frac{\mu^2}{4L^2}} + 8\Delta_3 \eta)$. Thus, we conclude the proof.