SPDL: A Blockchain-Enabled Secure and Privacy-Preserving Decentralized Learning System

Minghui Xu, *Member, IEEE*, Zongrui Zou, Ye Cheng, Qin Hu, *Member, IEEE*, Dongxiao Yu, *Senior Member, IEEE*, and Xiuzhen Cheng, *Fellow, IEEE*

Abstract—Decentralized learning involves training machine learning models over remote mobile devices, edge servers, or cloud servers while keeping data localized. Even though many studies have shown the feasibility of preserving privacy, enhancing training performance or introducing Byzantine resilience, but none of them simultaneously considers all of them. Therefore we face the following problem: *how can we efficiently coordinate the decentralized learning process while simultaneously maintaining learning security and data privacy for the entire system?* To address this issue, in this paper we propose SPDL, a blockchain-secured and privacy-preserving decentralized learning system. SPDL integrates blockchain, Byzantine Fault-Tolerant (BFT) consensus, BFT Gradients Aggregation Rule (GAR), and differential privacy seamlessly into one system, ensuring efficient machine learning while maintaining data privacy, Byzantine fault tolerance, transparency, and traceability. To validate our approach, we provide rigorous analysis on convergence and regret in the presence of Byzantine nodes. We also build a SPDL prototype and conduct extensive experiments to demonstrate that SPDL is effective and efficient with strong security and privacy guarantees.

Index Terms—Blockchain, byzantine resilience, decentralized learning, privacy preservation

1 Introduction

With the increasing amount of data and growing complexity of machine learning models, there is a rigid demand for utilizing computational hardware and storage owned by various entities enrolled in distributed computing environment. State-of-the-art distributed machine learning systems adopt three major network topologies shown in Fig. 1. Federated learning [1] utilizes an efficient centralized network illustrated in Fig. 1a, where a parameter server aggregates gradients computed by distributed devices and updates the global model for them while preserving privacy since devices compute locally without communicating with each other. The fragility of the centralized network topology lies in that a centralized parameter server suffers from the single point of failure problem (a server might crash or be Byzantine). To solve this issue, El-Mhamdi *et al.* [2] proposed a Byzantine-resilient learning network shown in Fig. 1b, which substitutes the centralized server with a server group in which no more than $1/3$ servers can be Byzantine.

In this paper, we take a step further to break the barriers among the parameter servers and the computational devices, and allow all nodes to train models in a decentralized network, as shown in Fig. 1(3). Such a decentralized network is frequently adopted in edge computing [3], Internet-of-Things (IoT) [4], decentralized applications (Dapp), etc. It can greatly unleash the potential for building large-scale (even worldwide) machine learning models that can reasonably and fully maximize the utilization of computational resources [5]. Besides, devices such as mobile phones, IoT sensors and vehicles are generating a large amount of data nowadays. A decentralized network filled with real-time Big Data can lead to tremendous improvements in large-scale applications, e.g., illness detection, outbreak discovery, and disaster warning, which involve a large number of decentralized edge and cloud servers from different regions and/or countries. However, this poses a new critical challenge: *How can we efficiently coordinate the decentralized learning process while simultaneously maintaining learning security and data privacy for the entire system?*

Specifically, decentralized learning confronts with the following challenges. 1) Without a fully trusted centralized custodian, users have no incentive but are reluctant to participate in the learning process due to the lack of trust in a decentralized network. Therefore, it is highly possible that the volume of data might be insufficient to train a reliable model. 2) In decentralized learning, a Byzantine node who can behave arbitrarily (e.g., crash or launch attacks) might prevent the model from convergence or interrupt the training

- *Minghui Xu, Zongrui Zou, Ye Cheng, Dongxiao Yu, and Xiuzhen Cheng are with the School of Computer Science and Technology, Shandong University, Qingdao 266510, China. E-mail: {peter_xumh, chengye0311}@163.com, zou.zongrui@mail.sdu.edu.cn, {dxyu, xzcheng}@sdu.edu.cn.*
- *Qin Hu is with the Department of Computer and Information Science, Indiana University-Purdue University Indianapolis, Indianapolis, IN 46202 USA. E-mail: qinhu@iu.edu.*

Manuscript received 20 Jan. 2022; revised 2 Apr. 2022; accepted 10 Apr. 2022. Date of publication 21 Apr. 2022; date of current version 13 Jan. 2023.
This work was supported in part by the National Key R&D Program of China under Grant 2019YFB2102600, in part by the National Natural Science Foundation of China under Grants 61832012 and 62102232, and in part by the Blockchain Core Technology Strategic Research Program of Ministry of Education of China under Grant 2020KJ010301.
(Corresponding author: Minghui Xu.)
Recommended for acceptance by Q. Li.
Digital Object Identifier no. 10.1109/TC.2022.3169436

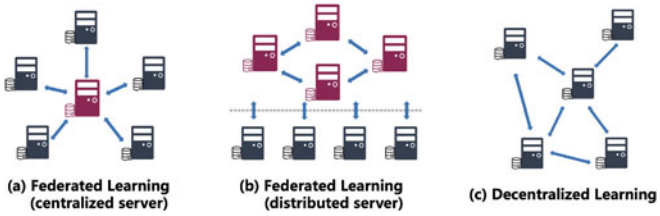


Fig. 1. Three major network topologies adopted in distributed learning.

process. 3) It is challenging to make the trade-off between privacy & security as well as efficiency, and to make data sharing frictionless with privacy and transparency guarantees.

To overcome the above challenges, we propose SPDL, a decentralized learning system that simultaneously ensures strong security using blockchain (as an immutable distributed ledger), BFT consensus, and BFT GAR, and preserves privacy utilizing local gradient computation and differential privacy (DP). Blockchain, as a key component, maintains a complete, immutable, traceable record of the machine learning process, covering user registration, gradients by rounds, and model parameters. With blockchain, a user can identify illegal and Byzantine peers, update its local model without concerns, and ultimately trust the SPDL. The BFT consensus algorithm and the BFT GAR are embedded into the blockchain. Concretely, the BFT consensus algorithm ensures consistency of model transition during multiple rounds while the BFT GAR offers an effective method of detecting and filtering Byzantine gradients at each round. Concerning privacy, we let nodes compute gradients with their local training data, and only share perturbed gradients with peers, which provides a strong privacy protection.

Our contributions are summarized as follows

1) To our best knowledge, this is the first secure and privacy-preserving machine learning system for decentralized networks in which learning processes are free from trusted parameter servers.
2) SPDL makes use of DP for data privacy protection and seamlessly embeds BFT consensus and BFT GAR into a blockchain system to benefit model training with Byzantine fault tolerance, transparency, and traceability while retaining high efficiency.
3) We conduct rigorous convergence and regret analyses on SPDL in the presence of Byzantine nodes, build a prototype, and carry out extensive experiments to demonstrate the feasibility and effectiveness of SPDL.

This paper is organized as follows. We summarize the most related work in Section 2. Section 3 outlines the necessary preliminary knowledge needed by the development of SPDL. Section 4 details the SPDL design. The analysis on convergence and regret are presented in Section 5. Evaluation results of SPDL are reported in Section 6. Finally, we conclude this paper in Section 7.

2 Related Work

2.1 Privacy and Byzantine Resilience in Distributed Learning

Private learning schemes include secure multiparty computation, encryption, homomorphic encryption, differential privacy, and aggregation models. For details, we recommend two comprehensive surveys [6], [7] to the interested readers. Su and Vaidya [8] introduced the distributed optimization problem in the presence of Byzantine failures. The problem was formulated as one in which each node has a local cost function, and aims to optimize the global cost function. The proposed method, namely the synchronous Byzantine gradient method (SBG), first trims the largest f gradients and the smallest f gradients, then computes the average of the minimum and the maximum of the remaining $N - 2f$ values. This approach sheds light on providing byzantine resilience for distributed learning. Following this idea, many byzantine-resilient aggregation rules were proposed. They all work towards a common goal – more precisely and efficiently trim the byzantine values. Blanchard *et al.* were the earliest to tackle the Byzantine resilience problem in distributed learning by the Krum algorithm which can guarantee convergence despite f Byzantine workers (in a server-worker architecture). Krum stimulates the arrivals of many BFT aggregation rules including Median [9], Bulyan [10], and MDA [2]. A recent work [11] demonstrates that these aggregation rules can function together with the DP technique under proper assumptions.

2.2 Blockchain-Enhanced Distributed Learning

The BinDaaS [12] system provides a blockchain-based deep learning service, ensuring data privacy and confidentiality in sharing Electronic Health Records (EHRs). With BinDaaS, a patient can mine a block filled with a private health record, which can be accessed by legitimate doctors. BinDaaS trains each model based on a patient's private EHRs independent of others' data. Hu *et al.* [13] utilized a blockchain and a game-theoretic approach to protect the user privacy for federated learning in mobile crowdsensing. The collective extortion (CE) strategy was proposed in [14] as an incentive mechanism that can regulate workers' behavior. These two game-based approaches cannot strictly guarantee the safety of model training against byzantine nodes. Lu *et al.* [15] developed a learning scheme that protects privacy by differential privacy, and proposed the Proof of Training Quality (PoQ) consensus algorithm for model convergence. This scheme does not consider byzantine users who might disturb a training process by proposing erroneous model parameters. FL-Block [16] allows end devices to train a global model secured by a PoW-based blockchain. LearningChain [17] is a differential privacy based scheme to protect each party's data privacy, also resting on a PoW-based blockchain. BAFL [4] is a asynchronous federated learning system designed for IoT scenarios. Warnat-Herresthal *et al.* [18] proposed the concept of swarm learning which is analogous to decentralized learning, in which each node can join the learning process managed by Ethereum (again, PoW-based) and smart contracts. Another noteworthy system is Biscotti [19], which utilizes a commitment scheme to protect data privacy and adopts a novel Proof-of-Federation based blockchain for security guarantee in decentralized learning. Using a commitment scheme, the aggregation of parameters can be verifiable and tamper-proof.

Compared to a pure adoption of centralized federated learning schemes, PoW-based blockchains can help avoid the single point of failures and mitigate poisoning attacks

caused by a central parameter server. With PoW, a miner can propose a block containing model parameters used for a global model update. However, this method cannot rigorously prevent malicious miners from harming the training processes by proposing wrong updates. The PoW consensus itself is not sufficient to judge whether given parameters are byzantine or not. Therefore, PoW-based blockchains are vulnerable to poisoning attacks launched by byzantine nodes. Besides, PoW-based blockchains are hard to scale since PoW can incur much overhead and result in heavy consumption of computational resources.

In this paper, we leverage the DP technique for privacy protection since adding noises (only needs an addition operation using pre-calculated noise) is more efficient than employing complicated cryptographic tools. In addition, using DP and a BFT consensus, we can ensure the byzantine fault tolerance for the whole training process, which cannot be realized by existing works. Our unique contributions lie in two aspects: 1) providing security and privacy guarantees for the complete training process by seamlessly integrating DP, BFT aggregation, and blockchain in one system, considering byzantine nodes; and 2) offering rigorous analysis on the Byzantine fault-tolerance, convergence and regret for decentralized learning.

3 Preliminaries

3.1 Decentralized Learning

With the proliferation of machine learning tasks that need to process massive data, distributed learning was proposed to coordinate a large number of devices to complete a training process so as to achieve a rapid convergence. In recent years, the research on distributed learning was mainly carried out along two directions: centralized learning and decentralized learning, whose schematic diagrams are shown in Fig. 1. A centralized topology uses a parameter server (PS) to coordinate all workers by gathering gradients, performing local updates, and broadcasting new parameters; while in a decentralized topology, all nodes are considered as equal and exchanges information without the intervention of a PS.

Decentralized learning has many advantages over centralized learning. In [5], Lian *et al.* rigorously proved that a decentralized algorithm has lower communication complexity and possesses the same convergence rate as those under a centralized parameter-server model. Furthermore, a centralized topology might not hold in a decentralized network where no one can be trusted enough to act as a parameter server. Therefore, We consider the following decentralized optimization during a learning process:

$$\min_{x\in\mathbb{R}^N} f(x) = \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}_{\xi\sim D_i} F(x;\xi),$$

where N is the network size, D_i is the local data distribution for node i, $F(x;\xi)$ denotes the loss function given model parameter x and data sample ξ. Let $f_i(x) = \mathbb{E}_{\xi\sim D_i} F(x;\xi)$. In a fully-connected graph, during any synchronous round, each node performs a deterministic aggregation function (*e.g.* average function) $\mathcal{K}$ on perturbed gradients received from all other peers to update its local parameter, i.e.,

$$x_i^{(t+1)} = x_i^{(t)} - \gamma \cdot \mathcal{K}(g_1^{(t)}, g_2^{(t)}, \ldots, g_n^{(t)}),$$

where γ is the learning rate. Notice that since $\mathcal{K}$ is deterministic, all nodes should have exactly the same parameter and should initialize it with the same value. In this case, we denote by $\mathcal{A}$ an arbitrary synchronous distributed learning algorithm that updates the mutual parameter x, and use $\mathcal{A}(D_1, D_2, \ldots, D_n; t)$ to denote the output parameter x at round t. In the rest of this paper we omit the subscript if all peers have the same local parameter.

3.2 Blockchain Basics

Blockchain, as a distributed ledger technology, refers to a chain of blocks linked by hashes and spread over all the nodes in a peer-to-peer network, namely blockchain network. A full node stores a complete blockchain in its local database. A blockchain starts from a genesis block, and each block except for the genesis block is chained to a previous block by referencing its hash. Typically, there exist two categories of blockchain systems based on scale and openness: permissioned and permissionless. In this paper, we adopt a permissioned blockchain since it can provide faster speed and more restricted registration control than permissionless ones.

Blockchain consists of three major components: blockchain network, distributed ledger, and consensus algorithm. A blockchain system organizes registered nodes into a P2P network, formulating a complete graph. A distributed ledger is immutable and can be organized as a chain, a Direct Acyclic Graph (DAG), or a mesh. In this paper, we use a chain as the data structure of our ledger. As the core of a blockchain system, the consensus process determines how to append a new block to the chain. Two types of consensus algorithms are commonly adopted: proof-of-resources and message passing. Proof-of-resources means that nodes compete for proposing blocks by demonstrating their utilization of resources, e.g., computational resources, stake, storage, memory and specific trust hardware. On the other hand, message passing based consensus has been widely researched in the area of distributed computing. Such algorithms always provide clear assumptions on nodes' faulty behaviors such as fail-stop and Byzantine attacks. In this paper, we leverage Byzantine fault tolerance (BFT) consensus algorithm, which can address Byzantine nodes who can launch arbitrary attacks.

3.3 Gradient Aggregation Rule (GAR)

A Gradient Aggregation Rule (GAR) is used to aggregate gradients received from peers during each round. A traditional GAR averages gradients to eliminate errors. Concerning gradients generated by Byzantine nodes, a GAR can be more elaborately designed to inject robustness. For example, Krum and Multi-Krum are the pioneering GARs that satisfy Byzantine resilience [20]. The essence behind these two GARs are to choose the gradient with the closest $(N-f)$ (N denotes the network size and f denotes the number of Byzantine nodes) neighbors based on the assumption that the honest majority should have similar gradients. Median [9] and MDA [2] are another two GARs that adopt analogous ideas to ensure the BFT gradient aggregation. In this paper, SPDL leverages Krum as the BFT GAR but is not limited to it.

3.4 Differential Privacy

The privacy guarantee is of vital importance if the nodes carrying out decentralized learning do not admit their local training data to be shared. Although each node communicates with its neighbors by transmitting parameters instead of sending raw data, the risk of leaking information still exists [21]. Differential privacy is an effective method to avoid leaking any information of a single individual by adjusting the feedback of the query operations, no matter what auxiliary information the adversary node might have. In decentralized learning, the process of exchanging parameters involves a sequence of queries. The differential privacy in our setup can be formally defined as follows:

Definition 1. *((ϵ, δ)-differential-privacy) Denote by $\mathcal{A}$ an arbitrary synchronous distributed learning algorithm that updates model parameter x. For any node i in a decentralized system and any two possible local data-sets D_i and D_i' with D_i differing from D_i' by at most one record, if for any round t and any $S \subseteq Range(\mathcal{A})$, it holds that*

$$Pr(\mathcal{A}(\mathcal{D}, D_i; t) \in S) \leq e^{\epsilon} Pr(\mathcal{A}(\mathcal{D}, D_i'; t) \in S) + \delta,$$

we then claim that $\mathcal{A}$ preserves (ϵ, δ)-differential-privacy, and pack all $D_j (j \neq i)$ into $\mathcal{D}$.

Definition 2. *For any function $f : \mathcal{D} \rightarrow R^N$, the L_2-sensitivity of f is defined as*

$$\Delta_2 f = \max_{d_1, d_2} ||f(d_1) - f(d_2)||,$$

for all d_1, d_2 differing in at most one element.

Differential privacy can be realized by adding Gaussian noises to the query results [22]. The following lemma demonstrates how to properly choose a Gaussian noise in SPDL. The detailed proof of Lemma 1 can be found in [23].

Lemma 1. *For each node transmitting gradients perturbed by Gaussian noise with distribution $\mathcal{N}(0, \sigma^2)$, the gradient exchanges within T successive rounds preserve (ϵ, δ)-differential-privacy as long as $\sigma \geq CT\gamma\sqrt{2\ln(1.25/\delta)}/\epsilon$, where $C = \Delta_2 g^{(t)}$ and γ is the learning rate.*

In brief, the differential private scheme we employed in this paper is sketched as follows. For any node i, we use random Gaussian noise to perturb its gradients before transmitting to other nodes. When nodes obtain the result of the aggregation function $\mathcal{K}$ whose inputs are their perturbed gradients, the differential privacy for node i is guaranteed.

4 System Design

4.1 System Model

We focus on scenarios where nodes are able to communicate with each other in a decentralized network. Specifically, we formalize the decentralized communication topology as a directed and fully connected graph $G = (V, E)$, where $V(|V| = N)$ denotes the set of all peers and for any $i, j \in V$,we have $(i, j) \in E$. Time is divided into epochs (denoted by k), with each consisting of synchronous rounds (denoted by t), and a model can be trained within each epoch. We denote frequently-used notations of transaction, block, blockchain, chain of block headers, by tx, $B_k^{(t)}$, $BC_k^{(t)}$, and $BH_k^{(t)}$, respectively.

TABLE 1
Summary of Notations

Symbol	Description
$\tilde{g}_i^{(t)}$	the gradient computed by i in round t
$\mathcal{G}_i^{(t)}$	the random noise added to i's gradient in round t
$g_i^{(t)}$	the perturbed gradient to be transmitted in round t
η	the learning rate
$x^{(t)}$	model parameters at the end of round t
$\xi_i^{(t)}$	datasets randomly sampled from local datasets D_i
n, f	the number of nodes and Byzantine nodes
σ^2	the variance of Gaussian noise
σ_f^2	the upper bound of $\mathbb{E}\|\mathbb{E}(g_i^{(t)}) - g_i^{(t)}\|^2$
$\Delta^{(t)}$	the output of BFT GARs
$B_k^{(t)}, BC_k^{(t)}$	block and blockchain in t-th round of k-th epoch
ϵ, δ	budgets of differential privacy
d	the dimension of gradients
$N(i)$	the neighbors of node i

We assume the network is unreliable with at most f possible Byzantine nodes, and $N = 3f + 1$. A Byzantine node i can behave arbitrarily. For example, it may refuse to compute gradient, transmit arbitrary but unlawful gradients to mislead correct nodes, and make incorrect votes during the consensus process. When i chooses not to send any data in a synchronous round t, it's neighbor acts like receiving $g_i^{(t)} = 0$. A summary of all important notations is provided in Table 1.

4.2 Design Objectives

In this subsection, we briefly summarize our design goals.

1) Decentralization: SPDL should work in a decentralized network setting without the intervention of any centralized party such as a parameter server.
2) Differential Privacy: SPDL should guarantee (ϵ, δ)-DP by adding random Gaussian noises to perturb gradients. Meanwhile we aim to reach a balance between privacy leakage and convergence rate.
3) Byzantine Fault-Tolerance: SPDL can ensure convergence against at most $f(N = 3f + 1)$ Byzantine nodes who can behave arbitrarily.
4) Immutability, Transparency, and Traceability: the full record of the machine learning process should be immutable and transparent, and provide traceability enabling Byzantine node detection.

4.3 SPDL Workflow

4.3.1 Initialization

Initially, each node creates a pair of private key sk and public key pk, and generates its unique 256-bit identity id based on pk. Public keys and identities are broadcast through the network to be publicly known by all nodes. Then a genesis block B_0 is created, which records the information of the nodes who initially participate in the blockchain network. A new coming node should have a related tx being added to the blockchain before joining the permissioned network,

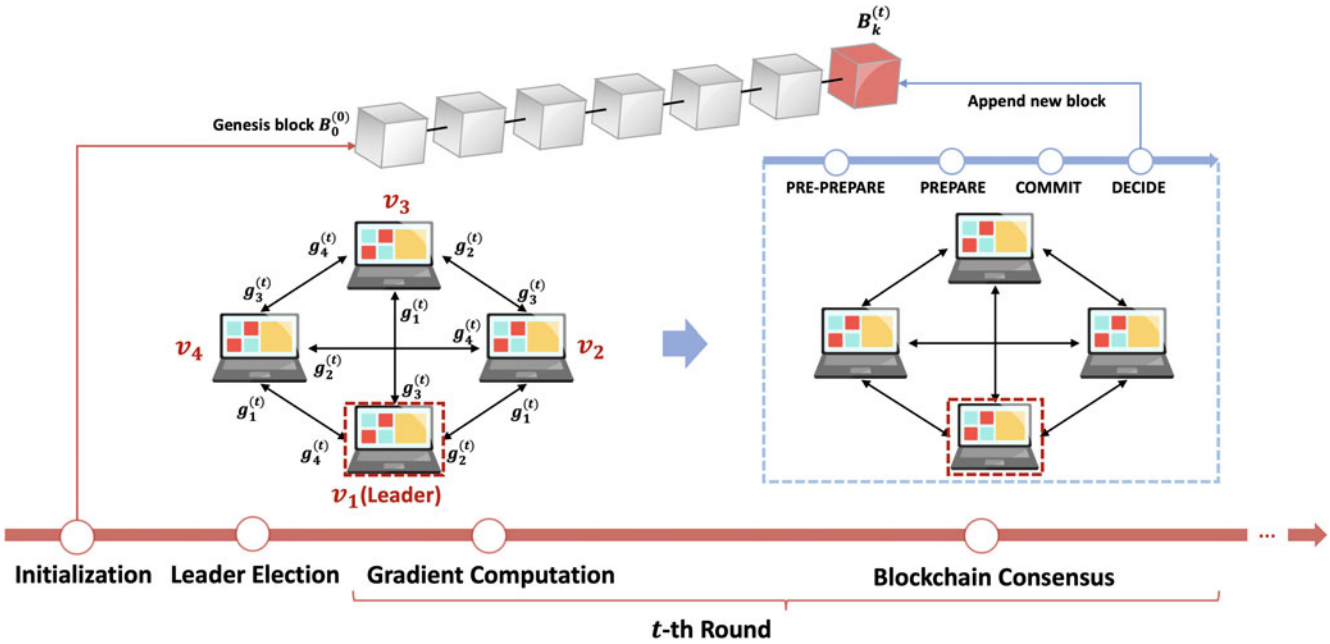


Fig. 2. SPDL workflow (t-th round).

where tx contains necessary information including its pk, id, IP address, etc. Initially, all nodes have an identical reputation value, that is $w_i = \hat{w}$. Each node initializes itself with a learning rate γ, the number of total rounds T, and the variance of Gaussian noise for perturbing the gradients. For simplicity and consistency of the iterating process, we assume all nodes start the learning procedure with the same initial parameter value $x_i^{(0)}$. After initialization, nodes undergo a leader election process to determine who is in charge of the training process.

4.3.2 Leader Election

Each node executes the leader election algorithm shown in Algorithm 1. The algorithm is based on the verifiable random function (VRF), which takes as inputs a private key sk and a random seed, and outputs a hash string h as well as the corresponding proof π. Each contender broadcasts (h_i, π_i) to the network and receives (h, π) from its peers. Note that each node is assigned with a reputation variable $r \in [0, 1]$. A node i with $r_i = 0$ is prohibited from being a leader. Then the node with the largest h and $r > 0$ is recognized as the leader who is responsible for the blockchain consensus. The case when more than one leaders are selected is extremely small since h has a large space of 2^{256} if we adopt the commonly used SHA-256; but if this extreme case happens, all nodes relaunch the leader election process to ensure that only one leader is finally selected. We also set a timeout for the leader election process as $\texttt{Time}() > start + \delta_1$, where $\texttt{Time}()$ extracts the current UNIX time. To summarize, our leader election algorithm achieves the following three basic functionalities:

- A node with a zero reputation value has no right of being a leader.
- The leader election process possesses full randomness and unpredictability properties.
- A Byzantine node cannot disguise itself as a leader since VRF ensures that the proof h is unforgeable.

After leader election, nodes start the round-based training process, with each round consisting of the gradient computation and blockchain consensus processes blockchain consensus processes as shown in Fig. 2.

Algorithm 1. Leader Election

1: (h_i, π_i) = VRF(sk$_i$, $seed$)
2: $L_i = (h_i, \pi_i, id_i)$
3: Broadcast (h_i, π_i) to the network and receive (h, π) form peers
4: **while** *TRUE* **do**
5: **if** receive (h_j, π_j) from id_j && VerifyVRF($pk, h_j, \pi_j, seed$)=1 **then**
6: add (h_j, π_j, id_j) to L_i
7: **if** $\texttt{Time}() > start + \delta_1$ **then**
8: select the largest h_{max} from L_i and obtain id_{max}
Output: id_{max}

4.3.3 Gradient Computation

At each round, node i exchanges its perturbed gradients with all other nodes in the blockchain network. Specifically, each node preserves a true stochastic gradient $\tilde{g}_i^{(t)}$ and a perturbed one $g_i^{(t)}$ to be shared. The whole exchange process can be summarized into the following steps:

- *Local Gradient computation:* compute local stochastic gradient $\tilde{g}_i^{(t)} = \nabla F_i(x^{(t)}, \xi_i^{(t)})$, where $\xi_i^{(t)}$ is randomly sampled from local dataset D_i.
- *Adding noise:* add random Gaussian noise to the local gradient to be shared. The variance of the noise is denoted by input variable σ.
- *Broadcast gradients:* send the perturbed local gradients to all other nodes, and receive gradients from others at the same time.

4.3.4 Blockchain Consensus

The blockchain consensus process deeply integrates a blockchain, a BFT consensus protocol (e.g., PBFT, Tendermint),

and a BFT aggregation function (e.g., Krum, Median). In this paper, we adopt the Practical Byzantine Fault Tolerant (PBFT) protocol as our consensus backbone due to its effectiveness validated by the Hyperledger Sawtooth. The aggregation rule used in blockchain consensus is Krum. Concretely, the blockchain consensus consists of four phases: PRE-PREPARE, PREPARE, COMMIT, and DECIDE.

Algorithm 2. Gradient Computation

Initialize: $x_i^{(0)}$, learning rate γ, number of total rounds T, and variance of noise σ
for $t = 0$ *to* $T-1$ **do**
  ▷ Local Computation
  Randomly sample $\xi_i^{(t)}$ and compute local stochastic gradient $\tilde{g}_i^{(t)} = \nabla F_i(x^{(t)}, \xi_i^{(t)})$
  ▷ Adding Noise
  Randomly generate Gaussian noise $\mathcal{G}_i^{(t)} \sim \mathcal{N}(0, \sigma^2)$ and add noise to the variable $g_i^{(t)} = \tilde{g}_i^{(t)} + \mathcal{G}_i^{(t)}$
  ▷ Broadcast Gradients
  Broadcast $g_i^{(t)}$ to the network and receive $g_j^{(t)}$ from each peer j

Algorithm 3. Blockchain Consensus

▷ To prevent deadlock, each node starts a view change if $\texttt{Time}() > start + \delta_2$
▷ PRE-PREPARE
if *role is leader* **then**
  $\Delta^{(t)} = \mathcal{K}(g_1^{(t)}, g_2^{(t)}, \ldots, g_n^{(t)})$
  $B_k^{(t)} \leftarrow \texttt{MSGB}(\Delta^{(t)})$
  broadcast $\langle \texttt{PREOmm-PREPARE}, id, B_k^{(t)}, h \rangle_{\hat{\sigma}}$
▷ PREPARE
if *role is follower* **then**
  compute $\tilde{\Delta}^{(t)} = \mathcal{K}(g_1^{(t)}, g_2^{(t)}, \ldots, g_n^{(t)})$
  while *receive* $\langle \texttt{PREOmm-PREPARE}, id, B_k^{(t)}, h \rangle_{\hat{\sigma}}$ **do**
    if σ *and* $B_k^{(t)}$ *are valid and* $\tilde{\Delta}^{(t)} \approx B_k^{(t)}.\Delta^{(t)}$ **then**
      broadcast $\langle \texttt{PREPARE}, id, h, vote \rangle_{\hat{\sigma}}$
▷ COMMIT
while *receive* $2f+1$ $\langle \texttt{PREPARE}, id, h, vote \rangle_{\hat{\sigma}}$ **do**
  broadcast $\langle \texttt{COMMIT}, id, vote \rangle_{\hat{\sigma}}$
▷ DECIDE
while *receive* $2f+1$ $\langle \texttt{COMMIT}, id, h, vote \rangle_{\hat{\sigma}}$ **do**
  $\texttt{Append}(BC_k^{(t)}, B_k^{(t)})$
  $x_i^{(t+1)} = x_i^{(t)} - \gamma \Delta^{(t)}$
  Update reputation

In the PRE-PREPARE phase, the leader computes an aggregated gradient using an aggregation function $\Delta^{(t)} = \mathcal{K}(g_1^{(t)}, g_2^{(t)}, \ldots, g_n^{(t)})$, where $\mathcal{K}$ is a (b, α)-Byzantine resilient Krum function. The core idea of Krum is to eliminate the gradients that are too far away from others. We use Euclid distance $\|g_i^{(t)} - g_j^{(t)}\|^2$ to measure how far two gradients are separated. Then we define $near(i)$ to be the set of $n - f - 2$ closest gradients to $g_i^{(t)}$. We expect that the Krum function chooses one gradient $g_i^{(t)}$ that is the "closest" to its surrounding gradients. More precisely, the output of Krum is one of its input gradients, and the index of this gradient is:

$$arg\min_i \sum_{j \in near(i)} \|g_i^{(t)} - g_j^{(t)}\|.$$

MSGB takes as input $\Delta^{(t)}$ and forms a new block $B_k^{(t)}$ which records $\Delta^{(t)}$. Then the leader broadcasts a signed pre-prepare message as $\langle \texttt{PRE} - \texttt{PREPARE}, id, B_k^{(t)}, h \rangle_{\hat{\sigma}}$.

Inthe PREPARE phase, each follower computes $\tilde{\Delta}^{(t)} = \mathcal{K}(g_1^{(t)}, g_2^{(t)}, \ldots, g_n^{(t)})$ based on its local perturbed gradients, then waits for pre-prepare messages. If a pre-prepare message is received, the follower first verifies the digital signature σ and the block (height, block hash, etc.). Then it compares $\tilde{\Delta}^{(t)}$ with $B_k^{(t)}.\Delta^{(t)}$. The requirement of $\tilde{\Delta}^{(t)} \approx B_k^{(t)}.\Delta^{(t)}$ means $\tilde{\Delta}^{(t)} \in (B_k^{(t)}.\Delta^{(t)} - \delta, B_k^{(t)}.\Delta^{(t)} + \delta)$, where δ is a small variation. This condition indicates that each follower should have a similar view on non-Byzantine gradients as the leader. If verification is passed, the follower broadcasts a prepare message $\langle \texttt{PREPARE}, id, h, vote \rangle_{\hat{\sigma}}$.

In the COMMIT phase, if a node receives $2f+1$ valid commit messages, it can broadcast a decision $\langle \texttt{COMMIT}, id, vote \rangle_{\hat{\sigma}}$ and enter into the following DECIDE phase.

In the DECIDE phase, upon receiving $2f+1$ valid commit messages, a node can append a new block $B_k^{(t)}$ to its local blockchain BC_k, update its local gradient as $x_i^{(t+1)} = x_i^{(t)} - \gamma \Delta^{(t)}$, and finally update its reputation. The reputation of a certain node i can be reduced if its gradient deviates from the aggregated gradient by more than $\pi/2$. Even though we do not explicitly introduce the view change, we do have such process to address the case when a leader is a Byzantine node. To avoid the occurrence of a deadlock, we set a timeout in the blockchain consensus process. If $\texttt{Time}() > start + \delta_2$, each node broadcasts a view change message $\langle \texttt{VIEW} - \texttt{CHANGE}, id, h \rangle_{\sigma}$ and waits for other peers' responses. Upon receiving $2f+1$ view change messages, a node can abandon the current round.

5 Theoretical Analysis

In this section, we provide both convergence analysis and regret analysis on SPDL in the presence of Byzantine nodes.

Without loss of generality, let $g_1^{(t)}, g_2^{(t)}, \ldots, g_{n-f}^{(t)}$ be the perturbed gradients sent out by honest nodes in round t, and $g_{n-f+1}^{(t)}, g_{n-f+2}^{(t)}, \ldots, g_n^{(t)}$ be the gradients sent out by possible Byzantine nodes in round t. We assume that the gradients derived by correct nodes are independently sampled from the random viable $\tilde{G}$ and that $\mathbb{E}(\tilde{G}) = g$. By adding Gaussian noise, a perturbed gradient then can be considered as an instance sampled from random variable $G = \tilde{G} + \mathcal{G}$, where $\mathcal{G}$ follows the Gaussian distribution of mean 0 and variance σ^2. Therefore we also have $\mathbb{E}(G) = g$. In this section, if we only concentrate on a certain round t, we omit the superscript on variables when ambiguity can be avoided from context.

Definition 3. *((k, f)-Byzantine Resilience) Let $0 < k \leq 1$ and f be the number of Byzantine nodes in a distributed system, then our anti-Byzantine mechanism $\mathcal{K}$ is said to be (k, f)-Byzantine Resilient if*

$$h^{(t)} = \mathcal{K}(g_1^{(t)}, g_2^{(t)}, \ldots, g_n^{(t)})$$

satisfies that $\langle \mathbb{E}h^{(t)}, g^{(t)} \rangle \geq k\|g^{(t)}\|^2$.

Theorem 1. *In round t, If $\|g\|^2 f^{-\frac{3}{4}} > 18 d \sigma_f^2$ and*

$$\epsilon > \frac{\sqrt{2C\ln(1.25/\delta)}}{C_2},$$

where

$$C_2 = \left(\frac{\|g\|^2}{18\ d} f^{-\frac{3}{4}} - \sigma_f^2\right)^{\frac{1}{2}}, \tag{1}$$

our anti-Byzantine mechanism $\mathcal{K}$ achieves (k, f)-Byzantine Resilience with

$$k = 1 - \frac{3\sqrt{2} f^{\frac{3}{2}} \sqrt{d}}{\|g\|}\left(\sigma_f^2 + \frac{2\ C^2 \ln(1.25/\delta)}{\epsilon^2}\right).$$

Proof. Denote by $N(i)$ the set of $n-f-2$ closest gradients of the i-th node, $N_c(i)$ the collection of the perturbed gradients in $N(i)$ sent by correct nodes, while $N_f(i)$ the gradients in $N(i)$ sent by the Byzantine nodes. Let i^* be the index of the gradient chosen by $\mathcal{K}$, we then have

$$\begin{aligned}
\|\mathbb{E}h^{(t)} - g\|^2 &= \left\|\mathbb{E}\left(h^{(t)} - \frac{1}{|N_c(i^*)|}\sum_{j\in N_c(i^*)}(g_i + r_i)\right)\right\|^2 \\
&\le \mathbb{E}\left\|\left(h^{(t)} - \frac{1}{|N_c(i^*)|}\sum_{j\in N_c(i^*)}(g_i + r_i)\right)\right\|^2 \\
&\text{(the above inequality holds since } \|\cdot\|\text{is convex.)} \\
&\le \sum_{i\notin\mathcal{B}}\mathbb{E}\left\|\left(g_i + r_i - \frac{1}{|N_c(i)|}\sum_{j\in N_c(i)}(g_j + r_j)\right)\right\|^2 \\
&\quad + \sum_{i\in\mathcal{B}}\mathbb{E}\left\|\left(\mathcal{B}_i - \frac{1}{|N_c(i)|}\sum_{j\in N_c(i)}(g_j + r_j)\right)\right\|^2
\end{aligned}$$

If $i^* = i$ is one of the correct nodes, i.e., $i^* \notin \mathcal{B}$,

$$\begin{aligned}
&\mathbb{E}\left\|\left(g_i + r_i - \frac{1}{|N_c(i)|}\sum_{j\in N_c(i)}(g_j + r_j)\right)\right\|^2 \\
&= \mathbb{E}\left\|\left(\frac{1}{|N_c(i)|}\sum_{j\in N_c(i)}(g_i + r_i - g_j - r_j)\right)\right\|^2 \\
&= \frac{1}{|N_c(i)|^2}\mathbb{E}\left\|\left(\sum_{j\in N_c(i)}(g_i - g_j) + (r_i - r_j)\right)\right\|^2 \\
&= \frac{1}{|N_c(i)|^2}\mathbb{E}\left\|\left(\sum_{j\in N_c(i)}(g_i - g_j) + (r_i - r_j)\right)\right\|^2 \\
&\le \frac{1}{|N_c(i)|}\sum_{j\in N_c(i)}\mathbb{E}\|g_i - g_j\|^2 + \mathbb{E}\|r_i - r_j\|^2 \\
&\le 2d(\sigma^2 + \sigma_f^2).
\end{aligned}$$

The above inequality holds since $\mathbb{E}(r_i - r_j)$ and $\mathbb{E}(g_i - g_j)$ are both 0. Because there are exactly $n-f$ correct nodes, we have

$$\begin{aligned}
&\mathbb{E}\left\|\left(g_i + r_i - \frac{1}{|N_c(i)|}\sum_{j\in N_c(i)}(g_j + r_j)\right)\right\|^2 \\
&\le 2d(n-f)(\sigma^2 + \sigma_f^2).
\end{aligned}$$

If $i^* = k$ is one of the Byzantine nodes, i.e., $i^* \in \mathcal{B}$,

$$\begin{aligned}
&\mathbb{E}\left\|\left(\mathcal{B}_k - \frac{1}{|N_c(k)|}\sum_{j\in N_c(k)}(g_j + r_j)\right)\right\|^2 \\
&\le \frac{1}{|N_c(k)|}\sum_{j\in N_c(k)}\mathbb{E}\|\mathcal{B}_k - g_j - r_j\|^2 \\
&\le \frac{1}{|N_c(k)|}\sum_{j\in N_c(i)}\mathbb{E}\|g_i + r_i - g_j - r_j\|^2 \\
&\quad + \frac{1}{|N_c(k)|}\sum_{j\in N_f(i)}\mathbb{E}\|g_i + r_i - g_j - r_j\|^2,
\end{aligned}$$

where $i \notin \mathcal{B}$ is any correct node. By the definition of $N(i)$, a node labeled $\zeta(i)$ is correct, but is farther away from any node in $N(i)$. Therefore we have:

$$\begin{aligned}
&\mathbb{E}\left\|\left(\mathcal{B}_k - \frac{1}{|N_c(k)|}\sum_{j\in N_c(k)}(g_j + r_j)\right)\right\|^2 \\
&\le \frac{1}{|N_c(k)|}\sum_{j\in N_c(i)}\mathbb{E}\|g_i + r_i - g_j - r_j\|^2 \\
&+ \frac{N_f(i)}{N_c(k)}\mathbb{E}\|g_i + r_i - g_{\zeta(i)} - r_{\zeta(i)}\|^2 \\
&\le \frac{N_c(i)}{N_c(k)}2d(\sigma^2 + \sigma_f^2) + \frac{N_f(i)}{N_c(k)}\sum_{j\notin\mathcal{B}\wedge i\neq i}\mathbb{E}\|g_i - g_j + r_i - r_j\|^2 \\
&\le 2d(\sigma^2 + \sigma_f^2)\left(\frac{N_c(i)}{N_c(k)} + \frac{N_f(i)}{N_c(k)}(n-f-1)\right) \\
&\le 2d(\sigma^2 + \sigma_f^2)\left(\frac{n-f-2}{n-2f-2} + \frac{b}{n-2f-2}(n-f-1)\right).
\end{aligned}$$

Combining the two results where i^* is a correct node or a Byzantine node, we have:

$$\begin{aligned}
\|\mathbb{E}h^{(t)} - g\|^2 &\le 2d(\sigma^2 + \sigma_f^2)\left(n - f + \frac{n-f-2}{n-2f-2}\right) \\
&\quad + 2d(\sigma^2 + \sigma_f^2)\left(\frac{b}{n-2f-2}(n-f-1)\right) \\
&\le 2d(\sigma^2 + \sigma_f^2)(f + 3 + f(f-1) + f^2(f+2)) \\
&\le 36df^3(\sigma^2 + \sigma_f^2).
\end{aligned}$$

Since

$$\epsilon > \frac{\sqrt{2C\ln(1.25/\delta)}}{C_2},$$

then

$$\epsilon^2\left(\frac{\|g\|^2}{18\ d}b^{-\frac{4}{3}} - \sigma_f^2\right) > 2c\ln(1.25/\delta).$$

Therefore

$$\|g\| > 3\sqrt{2}b^{\frac{3}{2}}\sqrt{d}(\sigma_f^2+\sigma^2)^{\frac{1}{2}}.$$

Finally, we have

$$\langle \mathbb{E}h^{(t)}, g\rangle \geq \left(\|g\| - 3\sqrt{2}b^{\frac{3}{2}}\sqrt{d}(\sigma_f^2+\sigma^2)^{\frac{1}{2}}\right)\|g\| = k\|g\|^2.$$

□

Regret analysis is commonly used in online learning to investigate the loss difference caused by two learning methods. Therefore we present a regret analysis on SPDL to show the incurred loss difference with and without Byzantine nodes.

In a decentralized system with Byzantine nodes, let $A_t \in \mathbb{R}^{d\times 1}$ be the parameter model of a node in round t, and $X_t \in \mathbb{R}^{d\times n}$ be the raw data randomly sampled by nodes in round t. The loss function in round t can be written as $F(A_t, X_t)$. Consider the destructive effect on the training process caused by Byzantine nodes, we denote by $\tilde{A}_t$ the parameter learned by the system per round in the absence of Byzantine nodes. It's meaningful to compare the gap of loss function computed by two different parameters A_t and $\tilde{A}_t$. If there is no Byzantine node, we assume $\tilde{A}_t$ is updated by any correct gradient $g_{\xi(t)}^{(t)}$ given by a random node $\xi(t) \in \{1, 2, \ldots, n\}$.

Definition 4. *(Regret)*

$$\mathcal{R}(A, \tilde{A}) = \sum_{i=0}^{T-1} F(\mathbb{E}A_t, X_t) - F(\mathbb{E}\tilde{A}_t, X_t).$$

Theorem 2. *If the loss function satisfies L_1-Lipschitz continuity, and $L_1 < 1$, then $\mathcal{R}(A, \tilde{A}) \leq \rho\sqrt{T} + o(\sqrt{T})$, where*

$$\rho = \frac{6L_1 f^{\frac{3}{2}}\sqrt{d(\sigma^2+\sigma_f^2)}}{1-L_1}.$$

Proof. By the property of L_1-Lipschitz continuity, we have

$$\begin{aligned}
&\sum_{i=0}^{T-1} F(\mathbb{E}A_t, X_t) - F(\mathbb{E}\tilde{A}_t, X_t)\\
&\leq \sum_{i=0}^{T-1} L_1\|\mathbb{E}A_t - \mathbb{E}\tilde{A}_t\|\\
&\leq \sum_{i=0}^{T-1} L_1\|\mathbb{E}A_{t-1} - \mathbb{E}\tilde{A}_{t-1} + \eta\mathbb{E}\left(h^{(t-1)} - g_{\xi(t)}^{(t-1)}\right)\|\\
&\leq \sum_{i=1}^{T-1} L_1\mathbb{E}\|A_{t-1} - \tilde{A}_{t-1}\| + \eta L_1\|\mathbb{E}h^{(t-1)} - \mathbb{E}g_{\xi(t-1)}^{(t-1)}\|.
\end{aligned}$$

Since

$$\mathbb{E}\|\mathbb{E}h^{(t)} - g^{(t-1)}\| \leq 6f^{\frac{3}{2}}\sqrt{d(\sigma^2+\sigma_f^2)},$$

we have

$$\eta\|\mathbb{E}h^{(t-1)} - \mathbb{E}g_{\xi(t-1)}^{(t-1)}\| \leq 6\eta f^{\frac{3}{2}}\sqrt{d(\sigma^2+\sigma_f^2)}.$$

Then we obtain

$$\begin{aligned}
&\sum_{i=0}^{T-1} F(\mathbb{E}A_t, X_t) - F(\mathbb{E}\tilde{A}_t, X_t)\\
&\leq \sum_{i=1}^{T-1} L_1\mathbb{E}\|A_{t-1} - \tilde{A}_{t-1}\| + 6L_1\eta f^{\frac{3}{2}}\sqrt{d(\sigma^2+\sigma_f^2)}\\
&\leq 6L_1\eta f^{\frac{3}{2}}\sqrt{d(\sigma^2+\sigma_f^2)} + \frac{6TL_1\eta f^{\frac{3}{2}}\sqrt{d(\sigma^2+\sigma_f^2)}}{1-L_1}.
\end{aligned}$$

□

Thus the theorem can be immediately proved by setting $\eta = \frac{1}{\sqrt{T}}$.

6 Evaluation

6.1 Configuration

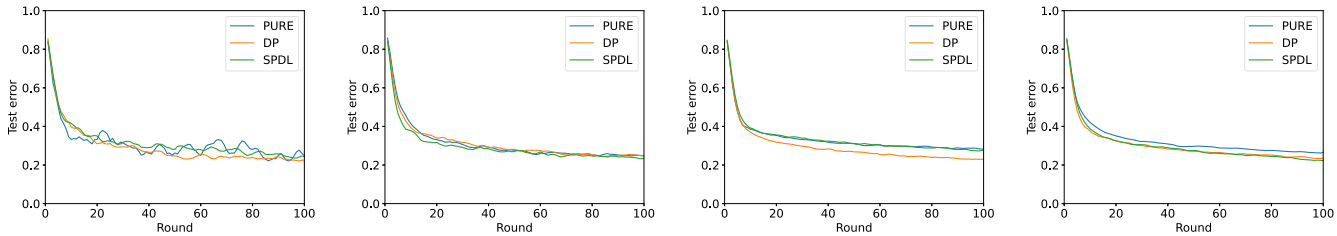
We implement SPDL with 3500 lines of Python code and conduct the experiments on a DELL PowerEdge R740 server which has 2 CPUs (Intel Xeon 4214R) with 24 cores (2.40 GHz) and 128 GB of RAM. SPDL adopts the gRPC framework, a P2P network for underlying communications, Pytorch for machine learning libraries, and a blockchain system with the PBFT consensus algorithm. We make SPDL open-sourced at Github.[1] Note that SPDL is not tightly bound up with a specific machine learning or deep learning method. In practice, we can adopt appropriate methods according to application requirements without extra modifications to the framework.

In SPDL, nodes bootstrap by generating key pairs using ECDSA, initializing the genesis block, establishing the gRPC connection, exchanging node list, and joining the P2P network. We evaluate SPDL over the image classification of MNIST dataset, which consists of handwritten digits of 70,000 28×28 images in 10 classes. The dataset is equally divided into N groups, with each assigned to one node. Each node can add Gaussian noise to its local gradients with the setting of $\epsilon = 0.02$ (if not stated otherwise) and $\delta = 10^{-6}$. We evaluate the performance of SPDL using the following standard metrics. 1) Test error: the fraction of wrong predictions among all predictions, using the test dataset. We measure the test error with respect to rounds, network size, batch size, privacy budget, and Byzantine ratio. 2) Latency: the latency of each round.

6.2 Evaluation Results

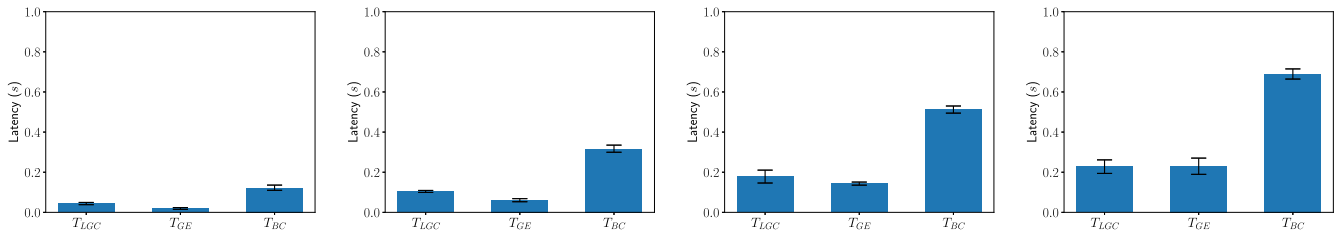
Convergence With Network Size. For simplicity in our context, we denote "PURE" as the decentralized learning scheme without leveraging any DP technique, BFT GARs, and blockchain system, and use "DP" to represent a decentralized learning scheme using the DP technique only based on "PURE". We first compare our SPDL with PURE and DP schemes in a non-Byzantine environment. As shown in Fig. 3, the test error nearly converges after 20 rounds, but fluctuates a lot when N is as small as four. When $N = 30$, all schemes almost achieve the same convergence. The SPDL and DP schemes sometimes (e.g., $N = 20$ or $N = 30$ in our

1. https://github.com/isSPDL/SPDL



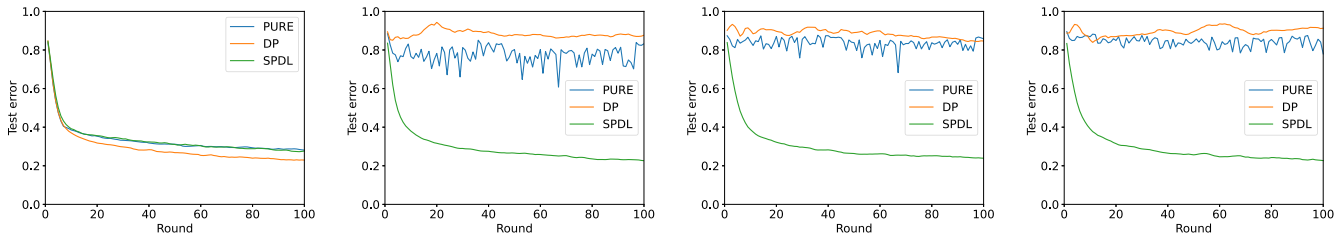
(a) $N = 4$ (b) $N = 10$ (c) $N = 20$ (d) $N = 30$

Fig. 3. Test error evolution with various network size $N = 4, 10, 20, 30$.



(a) $N = 4$ (b) $N = 10$ (c) $N = 20$ (d) $N = 30$

Fig. 4. Latency of different stages (T_{LGC}, T_{GE} and T_{BC}) concerning $N = 4, 10, 20, 30$.



(a) $BR = 0\%$ (b) $BR = 10\%$ (c) $BR = 20\%$ (d) $BR = 30\%$

Fig. 5. Test error evolution with various Byzantine ratio $BR \in 0\%, 10\%, 20\%, 30\%$.

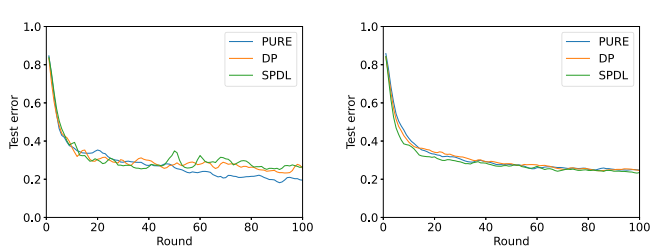
experiments) have lower test error than PURE because adding noises could prevent the training process from overfitting. Besides, the network size does not impact the convergence rate and a large network size contributes to stable convergence.

Latency. To better illustrate the latency of each round, we divide a round into three stages: local gradient computation plus adding noise whose overall time overhead is denoted by T_{LGC}, gradient exchange (T_{GE}), and blockchain consensus (T_{BC}). As Fig. 4 shows, T_{LGC}, T_{GE} and T_{BC} are in the same order of magnitude. T_{GE} grows with N simply because more nodes contend for computational resources. The MNIST classification task can be finished quickly (< 0.1 s/round), so T_{LGC} is lower than T_{BC} in our experiments. When the machine learning task becomes more difficult (e.g., 10 min/round), $T_{BC} \approx 1\ s$ is an acceptable overhead and could be even ignored.

Convergence in the Presence of Byzantine Nodes. We then make a comparison of three different schemes PURE, DP, SPDL with respect to Byzantine Ratio (BR), which is the number of existing Byzantine nodes over N. We set $N = 20$ and $BR \in 0\%, 10\%, 20\%, 30\%$ considering $f = 33\% \times N$, where $BR = 0$ represents the non-Byzantine case. As shown in Fig. 5, the results of the non–Byzantine experiments indicate that the three schemes can achieve similar convergence. However, DP and PURE schemes have high test error when $BR > 0\%$, and fail to ensure model convergence even in the presence of $10\%N$ Byzantine nodes. It is clearly shown that SPDL can still grantee the same convergence with respect to different levels of Byzantine attacks.

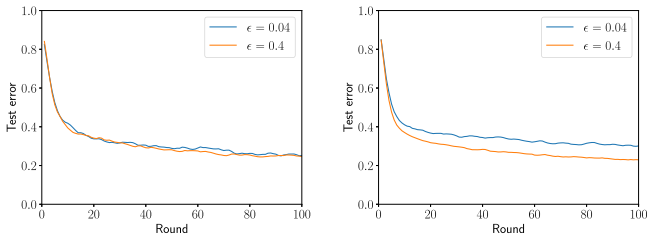
Batch Size. We then present the performance of the three schemes PURE, DP, SPDL with two different batch sizes (abbreviated as "BS") in Fig. 6. When $BS = 10$, the test error in all deployments fluctuate a lot, with the PURE scheme outperforming others because adding noises can perturb the model convergence. However, Fig. 6b indicates that we can increase the batch size to ensure a stable convergence and make our SPDL perform well as a PURE scheme.

Privacy Budget. We finally test our SPDL by setting $\delta = 10^{-6}$ with two varying $\epsilon \in 0.4, 0.04$. A smaller ϵ represents stronger privacy guarantee. The results presented in Fig. 7 demonstrate that when $N = 10$, adding noise with $\epsilon = 0.4$ or $\epsilon = 0.04$ have similar convergence. However, when $N = 20$,



(a) $BS = 10$ (b) $BS = 100$

Fig. 6. Test error evolution with batch sizes $BS = 10, 100$.



(a) $N = 10$ (b) $N = 20$

Fig. 7. Test error evolution with $\epsilon = 0.4, 0.04$ and $N = 10, 20$.

smaller ϵ can cause larger test error. This implies that the tradeoff between accuracy and privacy preservation should be carefully adjusted according to specific demands on privacy protection and model accuracy.

7 Conclusion and Future Works

SPDL is a novel decentralized machine learning system that can ensure efficiency while achieving strong security and privacy guarantee. In particular, SPDL utilizes BFT consensus and BFT GAR to protect model updates from harsh Byzantine behaviors, leverages blockchain to take advantage of the benefits of transparency and traceability, and adopts the DP technique for privacy protection. We provide rigorous theoretical analysis on the effectiveness of our approach and conduct extensive studies on the performance of SPDL with variations of network size, batch size, privacy budget, and Byzantine ratio.

In our future research, we intend to investigate the open problem of realizing heterogeneous decentralized learning in the presence of Byzantine nodes. Our basic idea is to generalize SPDL making it flexibly adapt to different types of heterogeneous environments. For example, we would consider training with heterogeneous data and training different personalized local models. We also intend to explore the feasibility of developing blockchain consensus algorithms that can deeply integrate with the process of model training in decentralized learning. Additionally, we are interested in implementing a SPDL system considering complicated non-convex optimization. A straightforward idea is to apply convex relaxation transforming a non-convex problem to an approximate convex one. Nevertheless, such a relaxation might introduce errors, which will be carefully investigated.

References

[1] K. Bonawitz *et al.*, "Practical secure aggregation for privacy-preserving machine learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 1175–1191.
[2] E.-M. El-Mhamdi, R. Guerraoui, A. Guirguis, L. N. Hoang, and S. Rouault, "Genuinely distributed Byzantine machine learning," in *Proc. 39th Symp. Princ. Distrib. Comput.*, 2020, pp. 355–364.
[3] Y. Zhan, S. Guo, P. Li, and J. Zhang, "A deep reinforcement learning based offloading game in edge computing," *IEEE Trans. Comput.*, vol. 69, no. 6, pp. 883–893, Jun. 2020.
[4] L. Feng, Y. Zhao, S. Guo, X. Qiu, W. Li, and P. Yu, "BAFL: A Blockchain-based asynchronous federated learning framework," *IEEE Trans. Comput.*, vol. 71, no. 5, pp. 1092–1103, May 2022, doi: 10.1109/TC.2021.3072033.
[5] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu, "Can decentralized algorithms outperform centralized algorithms? A case study for decentralized parallel stochastic gradient descent," 2017, *arXiv:1705.09056*.
[6] B. Liu, M. Ding, S. Shaham, W. Rahayu, F. Farokhi, and Z. Lin, "When machine learning meets privacy: A survey and outlook," *ACM Comput. Surv.*, vol. 54, no. 2, pp. 1–36, 2021.
[7] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, pp. 1–19, 2019.
[8] L. Su and N. H. Vaidya, "Fault-tolerant multi-agent optimization: Optimal iterative distributed algorithms," in *Proc. ACM Symp. Princ. Distrib. Comput.*, 2016, pp. 425–434.
[9] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 5650–5659.
[10] R. Guerraoui *et al.*, "The hidden vulnerability of distributed learning in Byzantium," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 3521–3530.
[11] R. Guerraoui, N. Gupta, R. Pinot, S. Rouault, and J. Stephan, "Differential privacy and Byzantine resilience in SGD: Do they add up?," 2021, *arXiv:2102.08166*.
[12] P. Bhattacharya, S. Tanwar, U. Bodke, S. Tyagi, and N. Kumar, "BinDaaS: Blockchain-based deep-learning as-a-service in healthcare 4.0 applications," *IEEE Trans. Netw. Sci. Eng.*, vol. 8, no. 2, pp. 1242–1255, Jun. 2021.
[13] Q. Hu, Z. Wang, M. Xu, and X. Cheng, "Blockchain and federated edge learning for privacy-preserving mobile crowdsensing," *IEEE Internet Things J.*, to be published, doi: 10.1109/JIOT.2021.3128155.
[14] Q. Hu, S. Wang, Z. Xiong, and X. Cheng, "Nothing wasted: Full contribution enforcement in federated edge learning," *IEEE Trans. Mobile Comput.*, to be published, doi: 10.1109/TMC.2021.3123195.
[15] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Blockchain and federated learning for privacy-preserved data sharing in industrial IoT," *IEEE Trans. Ind. Informat.*, vol. 16, no. 6, pp. 4177–4186, Jun. 2020.
[16] Y. Qu *et al.*, "Decentralized privacy using blockchain-enabled federated learning in fog computing," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 5171–5183, Jun. 2020.
[17] X. Chen, J. Ji, C. Luo, W. Liao, and P. Li, "When machine learning meets blockchain: A decentralized, privacy-preserving and secure design," in *Proc. IEEE Int. Conf. Big Data*, 2018, pp. 1178–1187.
[18] S. Warnat-Herresthal *et al.*, "Swarm learning for decentralized and confidential clinical machine learning," *Nature*, vol. 594, no. 7862, pp. 265–270, 2021.
[19] M. Shayan, C. Fung, C. J. M. Yoon, and I. Beschastnikh, "Biscotti: A blockchain system for private and secure federated learning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 7, pp. 1513–1525, Jul. 2021.
[20] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 118–128.
[21] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, and H. Qi, "Beyond inferring class representatives: User-level privacy leakage from federated learning," in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 2512–2520.
[22] H. Jiang, J. Pei, D. Yu, J. Yu, B. Gong, and X. Cheng, "Applications of differential privacy in social network analysis: A survey," *IEEE Trans. Knowl. Data Eng.*, to be published, doi: 10.1109/TKDE.2021.3073062.
[23] D. Yu *et al.*, "Decentralized parallel SGD with privacy preservation in vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 70, no. 6, pp. 5211–5220, Jun. 2021.

Minghui Xu (Member, IEEE) received the BS degree in physics from Beijing Normal University, Beijing, China, in 2018, and the PhD degree in computer science from George Washington University, Washington DC, USA, in 2021. He is currently an assistant professor with the School of Computer Science and Technology, Shandong University, China. His research focuses on blockchain, distributed computing, and applied cryptography.

Zongrui Zou is currently working toward the undergraduate degree in the School of Computer Science and Technology, Shandong University, Qingdao, China. His research interests mainly include theoretical aspects of private data analysis and machine learning.

Ye Cheng received the bachelor's degree in mechanical engineering from the Wuhan University of Technology, Wuhan, China, in 2018. He is currently working toward the master's degree in computer science and technology with Shandong University, Jinan, China. His current research direction is blockchain and privacy protection.

Qin Hu (Member, IEEE) received the PhD degree in computer science from George Washington University, Washington, DC, in 2019. She is currently an assistant professor with the Department of Computer and Information Science, Indiana University-Purdue University Indianapolis (IUPUI). Her research interests include wireless and mobile security, edge computing, blockchain, and crowdsourcing/crowdsensing.

Dongxiao Yu (Senior Member, IEEE) received the BS degree in mathematics from Shandong University, Jinan, China, in 2006, and the PhD degree in computer science from the University of Hong Kong, Hong Kong, in 2014. He became an associate professor with the School of Computer Science and Technology, Huazhong University of Science and Technology, in 2016. Currently, he is a professor with the School of Computer Science and Technology, Shandong University. His research interests include wireless networking, distributed computing, and graph algorithms.

Xiuzhen Cheng (Fellow, IEEE) received the MS and PhD degrees in computer science from the University of Minnesota, Twin Cities, in 2000 and 2002, respectively. She was a faculty member with the Department of Computer Science, George Washington University, from 2002–2020. Currently, she is a professor of computer science with Shandong University, Qingdao, China. Her research focuses on blockchain computing, security and privacy, and Internet of Things. She is a fellow of the CSEE.