When Machine Learning Meets Blockchain: A Decentralized, Privacy-preserving and Secure Design

Xuhui Chen*, Jinlong Ji*, Changqing Luo†, Weixian Liao‡ and Pan Li*
*Department of Electrical Engineering and Computer Science, Case Western Reserve University, Cleveland, OH 44106
Email: {xxc296, jxj405, pxl288}@case.edu
†Department of Computer Science, Virginia Commonwealth University, Richmond, VA 23284
Email: cluo@vcu.edu
‡Department of Computer and Information Sciences, Towson University, Towson, MD 20252
Email: wliao@towson.edu

***Abstract*—With the onset of the big data era, designing efficient and effective machine learning algorithms to analyze large-scale data is in dire need. In practice, data is typically generated by multiple parties and stored in a geographically distributed manner, which spurs the study of distributed machine learning. Traditional master-worker type of distributed machine learning algorithms assumes a trusted central server and focuses on the privacy issue in linear learning models, while privacy in nonlinear learning models and security issues are not well studied. To address these issues, in this paper, we explore the blockchain technique to propose a decentralized privacy-preserving and secure machine learning system, called *LearningChain*, by considering a general (linear or nonlinear) learning model and without a trusted central server. Specifically, we design a decentralized Stochastic Gradient Descent (SGD) algorithm to learn a general predictive model over the blockchain. In decentralized SGD, we develop differential privacy based schemes to protect each party's data privacy, and propose an l-nearest aggregation algorithm to protect the system from potential *Byzantine attacks*. We also conduct theoretical analysis on the privacy and security of the proposed *LearningChain*. Finally, we implement *LearningChain* on Etheurum and demonstrate its efficiency and effectiveness through extensive experiments.**

***Index Terms*—Decentralized Machine Learning; Blockchain; Differential Privacy; Byzantine Attack**

I. Introduction

In the big data era, data has become a key resource of intelligence and brings new opportunities to the modern society. We for the first time have the opportunity to gain deeper insights into our problems and develop better solutions by leveraging massive available data, such as in agriculture [1], bioinformatics [2], [3], wireless communications [4], and finance [5]. Thus, designing efficient and effective machine learning algorithms to analyze enormous volumes of data is in dire need.

A straightforward approach to conducting machine learning is to first collect and store the data in one central server, and then process them altogether [6], [7]. However, in many applications, large-scale data is typically generated by multiple parties and stored in a geographically distributed manner. Sending such substantial amounts of data from multiple data holders to a central server would incur significant communication overhead and raise critical data privacy and security concerns [8], [9].

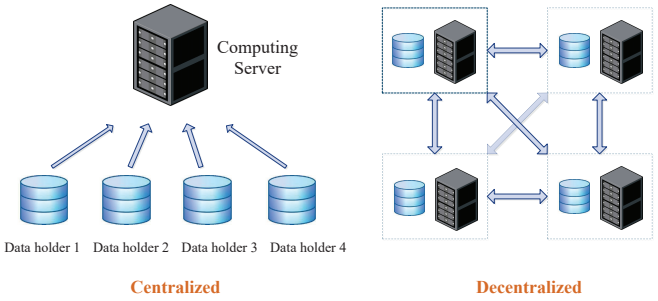


Fig. 1. Centralized Multi-party Learning (Master-Worker Pattern) vs. Decentralized Multi-party Learning

Towards that end, many researchers have designed distributed machine learning algorithms. A classic distributed machine learning paradigm is the master-worker pattern as shown in Fig. 1. Generally, each *data holder* (worker) computes certain abstract summary information (e.g., gradients) locally and transmits it to a central computing server. Then, the *central server* (master) aggregates these gradients for model updating. However, although the data holders only expose their abstract summary information to others, adversaries can still extract part of their original data information. For instance, Fredrikson et al. demonstrate a model-inversion attack that recovers images from the shared information in a facial recognition system [10].

To avoid privacy leakage, researchers have proposed several schemes to protect data privacy under the master-worker distributed machine learning architecture. A typical approach is to utilize secure multi-party computation (SMC) [11], where each individual party uses a set of cryptographic techniques and the oblivious transfer scheme to jointly compute a function using their private data [11], [12]. Such systems can guarantee that none of the parties can learn anything about any

978-1-5386-5035-6/18/$31.00 ©2018 IEEE

other individual's data. Nonetheless, cryptographic methods are computationally expensive for data holders because of the repeated encryption and decryption processes. An alternate approach to learning a model privately from multiple data sources is based on differential privacy. Differential privacy based distributed machine learning has been well studied, including private local classifier aggregation [13], and private exchange of gradient information to minimize empirical risks [14]. Note that these algorithms focus on linear learning models only and leave nonlinear learning models unexplored.

Another concern about the master-worker type of distributed learning systems is security threat. Particularly, the success of such learning systems heavily depends on the integrity of the central server, which, however, is vulnerable to attackers and can even be malicious itself. For instance, the central server may be "lazy" in the computing process for financial incentives and return imprecise results. What is worse, the server may intentionally return erroneous results to mislead the workers. Besides, some workers can be malicious as well who attempt to disrupt the learning process [15]. Recently, with the rapid development of cryptocurrency, some researchers propose distributed learning systems based on blockchain [16]–[18], the underlying technology of modern digital currency that provides a peer-to-peer network enabling sensitive participants to communicate and collaborate in a secure way without a fully-trusted third party or an authorized central node. For example, Kuo et al. [16] propose a blockchain based decentralized learning system for medical data and utilize blockchain as an information exchange platform. Note that these decentralized learning frameworks do not protect data privacy and are subject to malicious workers as well. Particularly, if an adversary, i.e., a *Byzantine attacker* [19], [20], pretends to be a data holder, it can easily break down the learning system by broadcasting misinformation over the network. Although there are a few Byzantine attack tolerant machine learning algorithms [19], [21]–[23], most of them study the general Byzantine attack in machine learning problems, and their computational complexity is usually very high.

To address the aforementioned privacy and security concerns, we explore blockchain to design a privacy-preserving and secure decentralized machine learning system, called *LearningChain*, by considering a general (linear or nonlinear) learning model and without a trusted central server. There are two main challenges for such a decentralized distributed machine learning: one is how to protect data holders' privacy, and the other is how to guarantee the resilience of the system to malicious users' attacks, i.e., Byzantine attacks. Specifically, we design a privacy-preserving and secure decentralized Stochastic Gradient Descent (SGD) algorithm over blockchain, which consists of blockchain initialization, local gradient computation, and global gradient aggregation. After network initialization, data holders compute their local gradients according to the common loss function and the current predictive model. Then they broadcast the local gradients protected by a differential privacy scheme designed for a general learning model to all the computing nodes in the network. Once the computing nodes reach a consensus at whom has the authority to create the next block in the chain, the authority holder will aggregate the local gradients using a proposed efficient Byzantine Attack tolerant aggregation scheme. Repetitively, the participants in the network can collaboratively train a predictive model without revealing their own data.

Our main contributions in this paper are summarized as follows:

- We propose a decentralized privacy-preserving and secure learning framework, called *LearningChain* (and its extension called *LearningChainEx*), which can be applied to any existing gradient descent based machine learning systems and any types of blockchain.
- We develop an efficient Byzantine attack tolerant aggregation algorithm, called "l-nearest aggregation", for decentralized learning systems.
- We design differential privacy mechanisms that can protect data holders' privacy in a general linear or nonlinear learning system.
- We provide theoretical analysis on the privacy and security of *LearningChain*.
- We implement *LearningChain* on Ethereum, and experiment results demonstrate its efficiency and efficacy.

The rest of this paper is organized as follows. In Section II, we introduce the related background techniques, i.e., blockchain and differential privacy. In Section III, we formulate the problem of decentralized machine learning and its privacy and security threats. After that, we describe the details of our proposed *LearningChain* (and *LearningChainEx*) in Section IV. Then we theoretically and empirically analyze *LearningChain* in Section V and Section VI, respectively. We finally conclude the paper in Section VII.

II. Background

In this section, we briefly introduce two important technologies, i.e., blockchain and differential privacy, for our proposed privacy-preserving and secure decentralized distributed machine learning framework.

A. Blockchain

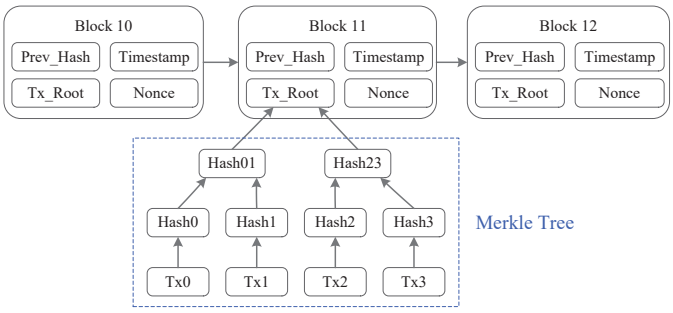


Fig. 2. The Structure of Bitcoin Blockchain Network

Blockchain, proposed by Nakamoto in 2008 for Bitcoin [24], is the cornerstone of the modern digital cryptocurrency

system. In the last few years, academia and industry have conducted extensive research on it and found that this advanced technology can be applied to applications in various fields (e.g., finance, healthcare, asset registration) [25]. Particularly, a blockchain is a distributed system which records all state update information, e.g., transactions in a cryptocurrency system, that has ever occurred in the system [26]. The state update information is replicated and shared among all the participants. The main feature of blockchain is that it allows untrusted participants to communicate and send the state update information to each other in a secure way without a fully-trusted third party or an authorized central party.

As shown in Fig. 2, taking Bitcoin network as an example, blockchain is a chronologically ordered list of blocks, where each block, identified by its unique cryptographic hash, refers to the block came before it, resulting in a chain of blocks. The miners in the system compete to win the authority to append a new block to the chain by proof-of-work (PoW) [27]. If a miner wins this game, it constructs a Merkle Tree [24] with the transactions in its memory pool, and adds these transactions into the new block. Once a block is created and appended to the blockchain, the transaction information in that block cannot be changed or reverted, which ensures the integrity of the system.

According to its participant permission, a blockchain system can be categorized as either a public chain or a private chain. In the former, anyone can join and leave the network at any time. In the latter, the blockchain enforces strict membership[1]. More specifically, there is an access control mechanism to determine who can join the system.

B. Differential Privacy

In this paper, we consider the differential privacy model introduced by Dwork [29]. Given any two databases D and D' differing by one element, which are referred to as *adjacent databases*, a randomized query function f is said to be differentially private if the probability that f produces a response S on D is close to the probability that f produces the same response S on D'. As the query output is almost the same in the presence or absence of an individual entry with high probability, nothing can be learned about any individual entry from the output.

Definition: ϵ-differential privacy: A randomized function f with a well-defined probability density P satisfies ϵ-differential privacy only if, for all adjacent databases D and D' and for any $S \in rang(M)$,

$$\frac{P(f(D) = S)}{P(f(D') = S)} \leq e^{\epsilon}.$$

[1]Some people categorize blockchain into three types: public chain, private chain, and consortium chain. Specifically, a consortium blockchain only allows a group of pre-selected ones to participate, while for a private blockchain, all the nodes are from one specific organization [28]. In this paper, we define any blockchain with an access control of participants as private blockchain.

When a function f outputs a real-value vector $g \in \mathbb{R}^d$, its L_2 sensitivity can be defined as

$$S(f) = \max_{D \sim D'} \|f(D) - f(D')\|_2, \tag{1}$$

where $\|\cdot\|_2$ is the L_2 norm.

Given such function f that has L_2 sensitivity $S(f) \leq C$, one approach to achieve ϵ-differential privacy is to perturb the output of f by $f(D) + \boldsymbol{\alpha}$, where $\boldsymbol{\alpha} \in \mathbb{R}^d$ is a noise vector generated randomly according to the probability density function:

$$\mu(\boldsymbol{\alpha}) \propto exp(-\beta \|\boldsymbol{\alpha}\|_2), \tag{2}$$

where $\beta = \frac{\epsilon}{C}$.

III. Problem Formulation

In this section, we mathematically formulate the problem of decentralized machine learning, and then introduce the threat model considered in this paper.

A. Decentralized Machine Learning

We consider that the training dataset $D = \{(\mathbf{x}_1, \mathbf{y}_1), ..., (\mathbf{x}_N, \mathbf{y}_N)\}$ is distributed among K parties $P_1, P_2, ..., P_K$. The k^{th} party P_k only processes a subset of the data $D_k \subseteq D$ with N_k samples, where $k \in [1, K]$. We assume that the data points in D assigned to any two distinct parties $P_k, P_l (k \neq l)$ are disjoint (i.e., $\{D_1, ..., D_k\}$ forms a *partition* of D). The goal is to learn a global general classifier $h(\mathbf{w}; \mathbf{x})$, which could be linear or nonlinear, from the complete dataset D.

One popular method of training a predictive model is the regularized risk minimization algorithm, which minimizes the objective function $J(\mathbf{w})$ formed as:

$$J(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^{N} \phi(h(\mathbf{w}; \mathbf{x}_i), \mathbf{y}_i) + \frac{\lambda}{2} \|\mathbf{w}\|_2^2, \tag{3}$$

where $\|\mathbf{w}\|_2^2$ is the regularizing term, $\lambda > 0$ is a regularization constant, and ϕ is the loss function (such as the logistic loss when using logistic regression, or the hinge loss when using support vector machine), which is assumed to be continuously differentiable.

Stochastic Gradient Descent (SGD), as one of the most popular optimization algorithms, is usually utilized to solve this minimization problem. We assume that the loss function ϕ is convex and differentiable, which ensures that gradients exist and that the minimizer of the objective is unique.

To run such an SGD algorithm, the computing nodes in a decentralized system iteratively update the global gradient for the predictive model $\mathbf{w}$. In the tth iteration, the gradient of $J(\cdot)$ with respect to the model $\mathbf{w}_t$ is given by

$$\nabla J(\mathbf{w}_t) = \frac{1}{N} \sum_{i=1}^{N} \frac{d}{d\mathbf{w}_t} \phi(h(\mathbf{w}_t; \mathbf{x}_i), \mathbf{y}_i) + \lambda \mathbf{w}_t, \tag{4}$$

which is equivalent to

$$\nabla J(\mathbf{w}_t) = \frac{1}{N} \sum_{k=1}^{K} \sum_{i=1}^{N_k} \frac{d}{d\mathbf{w}_t} \phi(h(\mathbf{w}_t; \mathbf{x}_i^k), \mathbf{y}_i^k) + \lambda \mathbf{w}_t. \tag{5}$$

It indicates that the gradient of $J(\mathbf{w}_t)$ can be computed via aggregating the local gradients from K different parties. In particular, each party P_k provides the gradient

$$\nabla g_k(\mathbf{w}_t) = \sum_{i=1}^{N_k} \frac{d}{d\mathbf{w}_t} \phi(h(\mathbf{w}_t; \mathbf{x}_i^k), \mathbf{y}_i^k) \tag{6}$$

of its local cumulative loss $g_k(\mathbf{w}_t) = \sum_{i=1}^{N_k} \phi(h(\mathbf{w}_t; \mathbf{x}_i^k), \mathbf{y}_i^k)$, while a computing node can calculate $\nabla J(\mathbf{w}_t)$ as

$$\nabla J(\mathbf{w}_t) = \frac{1}{N} \sum_{k=1}^{K} \nabla g_k(\mathbf{w}_t) + \lambda \mathbf{w}_t. \tag{7}$$

Subsequently, the model $\mathbf{w}_{t+1}$ is updated by:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \eta \nabla J(\mathbf{w}_t) \tag{8}$$

Repetitively, the data holders can collaboratively train a global predictive model.

B. Threat Model

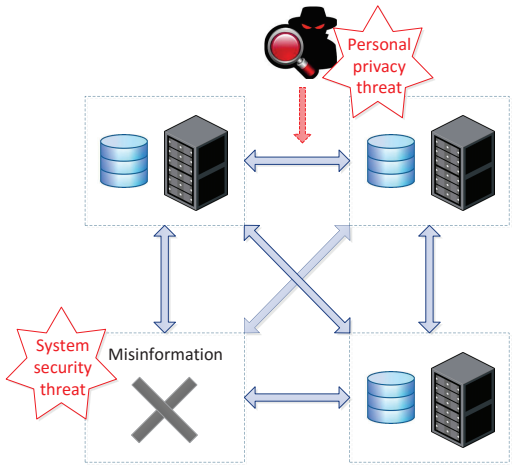


Fig. 3. Threat Model of Distributed Machine Learning

In this work, our goal is to develop a privacy-preserving and secure decentralized machine learning system. As shown in Fig. 3, there are mainly two kinds of threats: *personal privacy threat* and *system security threat*.

Privacy: From the data holders' perspective, their privacy include identity privacy and data privacy. Attackers, including outside attackers and malicious participants, attempt to steal data holders' private information by intercepting or eavesdropping the broadcasted messages.

Security: From the system's perspective, we consider any threat that may make the decentralized network fail to effectively learn an accurate predictive model as a system security attack. In our work, we mainly focus on Byzantine attacks in the system. The Byzantine attack in decentralized machine learning refers to the case that the system participants behave arbitrarily and do not follow the designed protocols. For example, if a data holder P_k sends a random vector instead of its real local gradient, we regard it as a Byzantine data holder. If a computing node deliberately miscomputes the sum of the local gradients from all the data holders, it is a Byzantine computing node.

Byzantine data holders: The Byzantine data holders in the system send forged information to other honest participants so as to jeopardize the learning performance. Since most of the systems cannot work when there are more malicious participants than honest participants, we assume that the majority of the data holders are honest in our system.

Byzantine computing nodes: The Byzantine computing nodes who append new blocks onto blockchain can generate incorrect model update information so as to trick data holders to calculate local gradients in a wrong way. The same as in the Bitcoin system, if the malicious computing nodes control more than 51% computing power in the network, which makes it more likely for them to obtain the authority to append new blocks, the learning model cannot converge. Therefore, we assume that the malicious computing nodes in our system control less than 51% of the total computing power.

IV. LearningChain

In this section, we design *LearningChain*, a decentralized, privacy-preserving and secure machine learning framework, which is able to protect the system from potential threats discussed in Section III-B.

A. System Overview

We outline the architecture of *LearningChain* in Fig. 4. The proposed *LearningChain* is a general decentralized learning framework and does not have any requirement on the type of blockchain. We consider that a number of data holders have their own private data, but the amount is limited. Thus, their common objective is to collaborate with each other to obtain a global predictive model. The network also contains a number of computing nodes that assist the data holders in learning the model while earning a financial profit. Once the learning process is done, the data holders share the cost, and the computing nodes get paid based on their contributions. Thus, we categorize the participants in the system into two roles: data holders, like coin owners in a cryptocurrency system, and computing nodes, like miners in a cryptocurrency system. Note that there is no rigid boundary between the two roles. Any computing node who owns data available for decentralized learning is also a data holder. If a data holder has extra computation resources, it is also a computing node. Nonetheless, the ones without any learning task related data can only work as computing nodes.

LearningChain consists of the following three processes:

- **Blockchain Initialization**: The learning system participants, both data holders and computing nodes, establish connections, set up a peer-to-peer network, and reach a consensus on the initial learning model settings.
- **Local Gradient Computation**: The data holders first create pseudo-identities and calculate the local gradients based on the current global model. Then, they employ

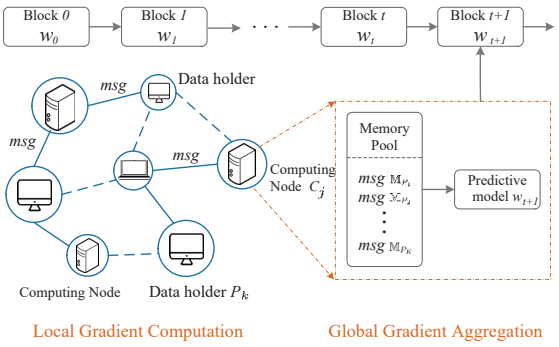


Fig. 4. The Training Process of LearningChain

a differential privacy scheme to perturb their local gradients. Encapsulated with other related information, the messages are broadcasted to all the computing nodes.

- **Global Gradient Aggregation**: The computing nodes compete to obtain the authority of appending a new block to the chain by solving a mathematical puzzle. Once one computing node wins the game, it applies a Byzantine attack tolerant aggregation scheme to the received local gradients in its memory pool and calculates the global gradient to update the model parameters $\mathbf{w}$. At last, it appends a newly created block with related information to the chain.

Repetitively executing local gradient computation and global gradient aggregation, *LearningChain* can train a global predictive model. We summarize *LearningChain* in Algorithm 1, and illustrate the details of each process in the following.

B. Blockchain Initialization

The primary objective of this phase is to construct a fully decentralized peer-to-peer (P2P) system and prepare the network for the following iterative learning process. The initialization process consists of two parts:

Network Construction: At the beginning of initialization, system participants, including data holders and computing nodes, get connected and form a peer-to-peer network.

Parameter Initialization: Before the learning iterations begin, all nodes on the blockchain reach a consensus on the hyper-parameters of the learning model, like the learning rate η, the learning batch size m, and the initial model $\mathbf{w_0}$. To initialize the system efficiently, a randomly chosen node can decide the initial settings of the system and broadcast them to all the other nodes.

After the initialization process, the *LearningChain* starts the following iterative learning process.

C. Local Gradient Computation

In this step, data holders compute and share their local gradients while attempt to limit the privacy leakage. Particularly, the data holders first need to protect the privacy of their identities. Then, they calculate their local gradients and perturb them with a differential privacy mechanism. Finally,

Algorithm 1 ***LearningChain***

Blockchain Initialization:
(a) Network Construction: Computing nodes $C_1, ..., C_M$, and data holders $P_1, ..., P_K$ form a peer-to-peer network
(b) Parameter Initialization: computing nodes $C_1, ..., C_M$ and data holders $P_1, ..., P_K$ initialize hyper-parameters: learning rate η, batch size m, and initial predictive model $\mathbf{w_0}$
for $t \in [1, T]$ and performance of the learned model does not meet requirement **do**
Local Gradient Computation:
(a) Data holder P_k, $\forall k \in [1, K]$, generates pseudo-identity P_k^*
(b) P_k calculates its local gradient:
- P_k retrieves current $\mathbf{w}_t$ from Block B_t
- P_k calculates local gradient using Eq. (6)
- P_k applies a differential privacy scheme to the local gradient using Eq. (10)
- P_k normalizes the gradient using Eq. (11), Eq. (12)

(c) P_k broadcasts the message msg_k:

$$\mathbb{M}_{P_k}^t = (P_k^*, \nabla \hat{g_k}(\mathbf{w}_t)^*, s_k^t, t)$$

Global Gradients Aggregation:
(a) Computing node C_j competes to solve PoW problem
(b) If it wins, C_j updates the predictive model:
- C_j retrieves local gradients from its memory pool
- C_j finds the sum gradient descent direction calculated by Eq. (14)
- C_j selects the l-nearest local gradients based on their cosine distances calculated by Eq. (15)
- C_j calculates the global gradient using Eq. (16)
- C_j updates the model using Eq. (8)

(c) C_j creates a new block B_{t+1} containing the following data:

$$B_{t+1} = (\mathbf{w}_{t+1}, [\mathbb{M}_{P_1}, \mathbb{M}_{P_2}, ..., \mathbb{M}_{P_K}])$$

a block contains hash, nounce, etc. What we show here is only the data load.
end for

the perturbed gradients are broadcasted to all the computing nodes. We present the detail as follows.

(a) Pseudo-identity Creation: To protect customers' privacy in a digital cryptocurrency system, the Bitcoin utilizes a pseudo-identity mechanism to anonymize customers in its blockchain network. In a decentralized learning system, the data holders have strong motivations to protect their identities as well. Therefore, we use the same method as that in the Bitcoin system to generate pseudo-identities for data holders [30]. Moreover, the data holders can generate as many pseudo-identities as they desire to avoid potential threats. We denote the pseudo-indentity of data holder P_k by P_k^*.

(b) Local Gradients Calculation: Each data holder P_k retrieves the current model $\mathbf{w}_t$ from the latest block B_t on

the chain. Then P_k calculates its local gradient based on $\mathbf{w}_t$ using its own training data by following Eq. (6), and then caps it by G, i.e.,

$$\nabla g_k(\mathbf{w}_t) := \nabla g_k(w_t)/max(1, \frac{||\nabla g_k(w_t)||_2}{G}). \quad (9)$$

After that, P_k perturbs its local gradient as follows:

$$\nabla g_k(\mathbf{w}_t)^* = \nabla g_k(\mathbf{w}_t) + \boldsymbol{\rho}_t^k, \quad (10)$$

where $\boldsymbol{\rho}$ is a noise vector following the distributed below:

$$\mu(\boldsymbol{\rho}^k) \propto exp(-\beta \left\|\boldsymbol{\rho}^k\right\|_2),$$

where $\beta = \epsilon/C$. Since $||\Delta g_k(w_t)||_2 \leq G$, it can be proved that the L_2 sensitivity of this algorithm computing $\Delta g_k(\mathbf{w}_t)$ from D_k is $2G$ (as shown in Appendix A). Thus, by letting $C = 2G$, the proposed perturbation scheme satisfies ϵ-differential privacy.

To reduce the workload of the computing node who aggregates local gradients from all the data holders, each data holder computes its normalized perturbed local gradient vector $\nabla \hat{g_k}(\mathbf{w}_t)^*$ i.e.,

$$s_k^t = \left\|\nabla g_k(\mathbf{w}_t)^*\right\|_2 \quad (11)$$

$$\nabla \hat{g_k}(\mathbf{w}_t)^* = \frac{\nabla g_k(\mathbf{w}_t)^*}{s_k^t}, \quad (12)$$

where $\|\cdot\|_2$ is the L_2 norm.

(c) Message Encapsulation and Sharing: Data holder P_k encapsulates its pseudo-identity, its normalized local gradient with magnitude, and the last block timestamp t into a message $\mathbb{M}_{P_k}$, then broadcasts it to computing users.

$$\mathbb{M}_{P_k} = (P_k^*, \nabla \hat{g_k}(\mathbf{w}_t)^*, s_k^t, t) \quad (13)$$

D. Global Gradients Aggregation

Computing nodes collect the local gradients broadcasted by data holders and store them in their memory pool. The winner node of PoW updates the model by aggregating the local gradients using a proposed Byzantine attack tolerant gradient aggregation algorithm, and then appends a new block to the chain. This process includes three steps: PoW competition, gradients aggregation, and new block creation.

(a) PoW Competition: In the Bitcoin system, the blockchain is replicated and held by all the computing nodes in the network. However, only one computing node has the authority to create a new block and append it to the main chain. Meanwhile, this node wins the reward for its contribution to the chain. Several protocols can reach a consensus in blockchain-based applications, such as Proof-of-Elapsed-Time (PoET), Proof-of-Authority (PoA). Proof-of-Work (PoW) [27], as one of the most popular consensus protocols, has proved its efficiency and correctness.

In *LearningChain*, we can take advantage of any consensus protocols. Taking PoW as an example, computing node C_j increments a nonce in the block until such a value is found that the block's hash meets a requirement. Then C_j has the right to create a new block on the chain and update the global model. C_j will earn the reward for its computation contribution to the system.

(b) Gradient Aggregation: As discussed in the threat model, Byzantine data holders can broadcast forged information to the computing nodes so as to prevent them from calculating the correct global gradient. Since the Byzantine attack tolerant machine learning algorithms in the literature [19], [21]–[23] usually bear high computational complexity, we design an efficient l-nearest aggregation scheme that is resilient to Byzantine attacks.

Specifically, assume that a computing node C_j has the block appending authority in the tth iteration, and hence has the responsibility to aggregate the local gradients in its memory pool correctly. First, C_j decapsulates the received K messages $\{\mathbb{M}_{P_1}, \mathbb{M}_{P_2}, \cdots, \mathbb{M}_{P_K}\}$, and adds the normalized gradient together. As mentioned in Section III-B, we assume that the majority of data holders are honest. Therefore, the sum vector $\mathbf{v}_s^t$ will guide the global decent to an appropriate direction:

$$\mathbf{v}_s^t = \sum_{k=1}^{K} \nabla \hat{g_k}(\mathbf{w}_t)^*. \quad (14)$$

Then the computing node C_j calculates the cosine distance between each data holders normalized local gradient and the sum vector:

$$dist_k^t = \frac{\mathbf{v}_s^t \cdot \nabla \hat{g_k}(\mathbf{w}_{t+1})^*}{\parallel \mathbf{v}_s^t \parallel \cdot \parallel \hat{g_k}(\mathbf{w}_{t+1})^* \parallel} \quad (15)$$

for $1 \leq k \leq K$. Based on $dist_k^t$, C_j picks the top l-nearest data holders' gradients to update the global gradient, i.e.,

$$\nabla J(\mathbf{w}_t) = \frac{1}{l} \sum_{i=1}^{l} s_i^t \times \nabla \hat{g}_i(\mathbf{w}_{t+1})^* \quad (16)$$

and subsequently update the predictive model:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \eta \nabla J(\mathbf{w}_t). \quad (17)$$

(c) New Block Creation: At the end, C_j writes the updated global model and the received local gradients into a new block and appends it to the main chain. The data contained in the new block is

$$B_{t+1} = (\mathbf{w}_{t+1}, [\mathbb{M}_{P_1}, \mathbb{M}_{P_2}, \cdots, \mathbb{M}_{P_K}]). \quad (18)$$

The iterations continue until the learned model meets the required performance or the maximum number of iteration T is reached.

V. Privacy and Security Analysis

We further analyze the privacy and security of *LearningChain* in this section.

A. Privacy Analysis

In *LearningChain*, we focus on data holders' identity privacy and sensitive data privacy. Recall that each data holder P_k only transmits $\mathbb{M}_{P_k}$ to others, inside which there are the perturbed local gradients ∇g_k^* and the pseudo-identity P_k^*.

Identity Privacy In *LearningChain*, every data holder utilizes a pseudo-identity to protect the real identity. Without auxiliary information, the real identity cannot be inferred from the pseudo-identity like in the Bitcoin system [30]. Besides, data holders can change their pseudo-identities in different learning processes, and even in different iterations for the same learning process to further protect their identities. The computing nodes can use the same protocols to protect their identities as well.

Data Privacy As described above, by letting each data holder set $C = 2G$ and $\beta = \epsilon/C$, the proposed algorithm computing $\Delta g_k(\mathbf{w}_t)$ from D_k satisfies ϵ-differential privacy. In other words, disclosing $\Delta g_k(\mathbf{w}_t)$ does not reveal the privacy of any individual data point in D_k. However, while each time the proposed algorithm is ϵ-differentially private, after J iterations, if an attacker collects all the J perturbed local gradients for a particular data holder, it can be shown that the data holder can only get $J\epsilon$-differential privacy.

There are two ways to address this issue. First, we let each data holder use a different pseudo-identity in each iteration. In so doing, the attacker would not be able to recognize the local gradients for any particular data holder. Thus, the local gradient computing algorithm remains ϵ-differentially private. Second, we propose the following extension for the previous differential privacy scheme. In the tth iteration, each data holder sets $\beta_t = \epsilon^t/C$, where $C = 2G$ and $\epsilon^t = \epsilon/T$ or $\epsilon^t = \epsilon \cdot \frac{1}{t(t+1)}$. Then, we can show that the local gradient computing algorithm is also ϵ-differentially private. We call the system "*LearningChainEx*" in this case.

B. Security Analysis

The objective of the learning system is to learn a predictive model from the distributed data holders effectively and successfully. In this study, we focus on Byzantine attacks in which attackers behave arbitrarily and do not follow the designed algorithms.

Byzantine Computing Nodes When a computing node solves the PoW problem and wins the authority to append a new block, it will check the previous block based on two rules. First, it checks the authenticity of all the data holders' messages in this block by comparing them with the history messages stored in the computing node's memory pool. Second, it checks the correctness of the global model update by repeating the predictive model update process using the data holders' local gradients. If the previous block does not satisfy any of these rules, the computing node will check the precedent block of this one. It will not stop checking until it finds a correct block.

Since we assume that the honest computing nodes are in control of the majority computing power ($\geq 51\%$) in the network, the honest ones will always obtain the opportunities to eliminate the impact brought by the Byzantine computing nodes. On the contrary, if the Byzantine computing nodes have more than 51% computation power in the network, they can always append erroneous information to the main chain, thus affecting the system. In any case, after iterations are done, each data holder can conduct a security check to quickly verify the correctness of the learned model. Assume the honest computing nodes are in control of at least p ($0 \leq p \leq 1$) of the total computing power in the network. The probability that there is at least one honest computing node involved in the last s blocks is $1-(1-p)^s$. Thus, by repeating the model update process in the last s blocks, each data holder can finally obtain the true model with a probability of $1-(1-p)^s$.

Byzantine Data Holders Byzantine data holders can attack the system by sending forged local gradients. Let ∇g_b denote the sum of all local gradients from Byzantine data holders and ∇g_h the sum of all local gradients from honest data holders. The optimal attack strategy for Byzantine data holders is to send their local gradients which satisfy

$$\|\nabla g_b\|_2 \geq \|\nabla g_h\|_2$$

$$dist = \frac{\nabla g_b \cdot \nabla g_h}{\| \nabla g_b \|_2 \cdot \| \nabla g_h \|_2} = -1,$$

i.e., such that Δg_b is in the opposite direction of Δg_h and with a larger magnitude.

To address this problem, in *LearningChain*, computing nodes only need to aggregate l nearest data holders' local gradients and do not need to wait for all local gradients to come. Thus, if Byzantine data holders follow the optimal attack strategy and wait to send their local gradients after receiving all the honest data holders', their local gradients may be disregarded in the global gradients aggregation process.

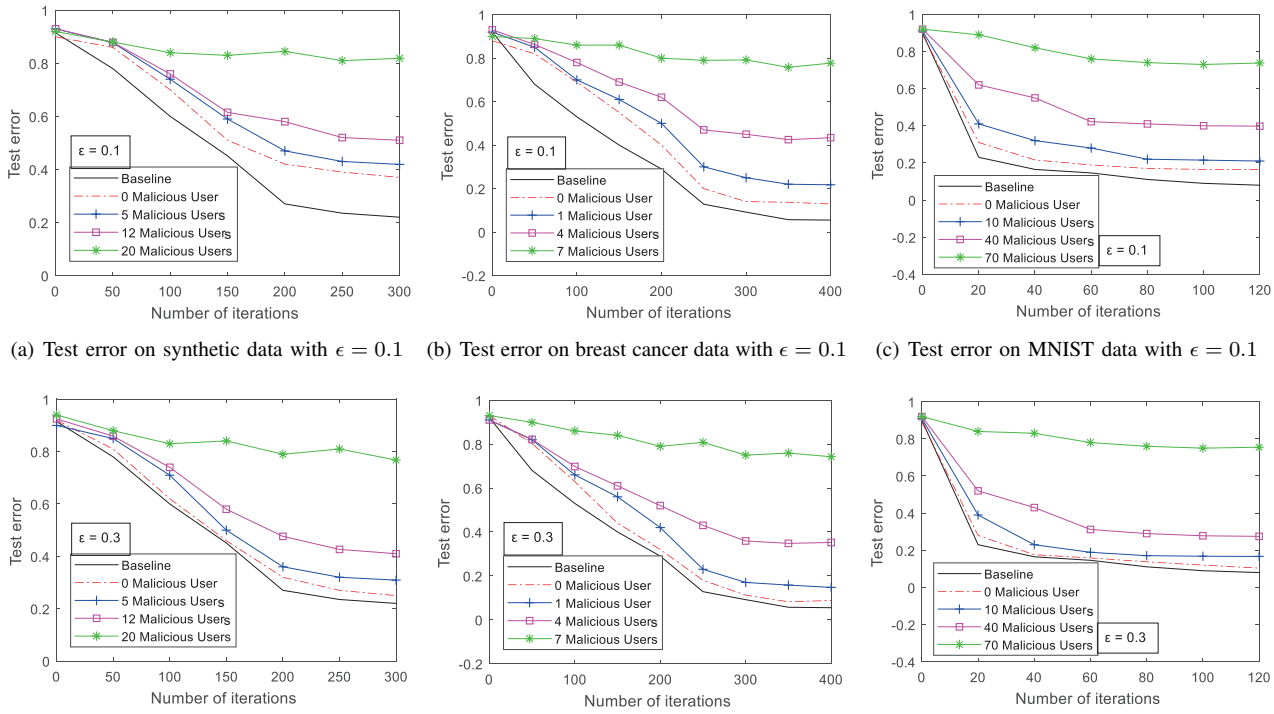
On the other hand, Byzantine data holders can behave arbitrarily, for example, to send random local gradients. In this scenario, since finally only the l nearest local gradients are selected, the attackers' impact is limited, particularly under the assumption that the majority of the data holders are honest.

VI. Experiment Results

In this section, we present the details of the experiments conducted to investigate the efficiency and efficacy of the proposed *LearningChain* and *LearningChainEx*. Particularly, we evaluate the performance of *LearningChain* with different privacy budgets and study the system's resilience to Byzantine attacks at different levels. Then, we compare with performance of *LearningChain* with that of *LearningChainEx*. Finally, we compare our l-nearest gradient aggregation scheme with the state-of-the-art Byzantine tolerant algorithms. Experiment results show that the proposed systems perform well on a variety of learning tasks.

A. Experiment Setup

Datasets: We select three different datasets to conduct decentralized learning tasks, which are synthetic dataset, Wisconsin breast cancer dataset [31], and MNIST dataset [32].



(a) Test error on synthetic data with $\epsilon = 0.1$ (b) Test error on breast cancer data with $\epsilon = 0.1$ (c) Test error on MNIST data with $\epsilon = 0.1$

(d) Test error on synthetic data with $\epsilon = 0.3$ (e) Test error on breast cancer data with $\epsilon = 0.3$ (f) Test error on MNIST data with $\epsilon = 0.3$

Fig. 5. Test Error on Synthetic Dataset, Breast Cancer Dataset and MNIST Dataset

- **Synthetic dataset**: We generate 30000 training and 30000 testing instances in $R^{10} \times \{\pm 1\}$ as follows: each instance $\mathbf{x_i}$ is drawn uniformly and randomly from a 10-dimensional unit hypersphere; a weight vector $\mathbf{w}$ is then chosen at random as the true classification vector, and instances are labeled according to the function sign($\mathbf{w} \cdot \mathbf{x}$). We equally divide the training dataset into 30 subsets and assign each subset to an individual agent who plays both roles of computing node and data holder.
- **Wisconsin breast cancer dataset**: The dataset contains 569 instances with 30 dimensions. Each instance is described by 10 features with a binary class label. We randomly choose 420 instances as training data and the rest as testing data. The training dataset is randomly divided into 10 parties, and each party in the experiment has roles of both computing node and data holder.
- **MNIST dataset**: The dataset has 60,000 training samples and 10,000 test samples. Each instance is a black and white (bilevel) image with a size of 28×28 pixels. We equally distribute the training dataset to 100 agents, who are computing nodes and data holders at the same time.

Implementation Settings: We implement *LearningChain* and *LearningChainEx* with Python and use Ethereum as the backbone blockchain. Ethereum is a new blockchain-based decentralized computing platform [33], [34], which is equipped with an attractive technique, Smart Contracts [35], [36]. They are applications with a state stored in the blockchain, which can facilitate, verify and enforce the process of a contract. Once they are created and deployed to Ethereum, the content of the contracts can never be modified. In the experiment, we use Smart Contracts to convey the secure global gradient aggregation algorithm.

B. Results Analysis

As shown in Fig. 5, we evaluate our proposed *LearningChain* with different differential privacy budgets and different numbers of Byzantine attackers. In our experiments, the baseline is the model trained in a centralized manner, which collects all the data on one computing server and trains the learning model without any privacy or security protection scheme (its performance is evaluated when there are no attackers in the system). We compare the performance of *LearningChain* with that of the baseline on the three testing datasets, respectively. We use the test error as the metric, which is defined as the ratio of the number of wrongly labeled samples to the total number of samples in the testing dataset.

Impact of Privacy Budget For synthetic dataset, to evaluate the impact of different privacy budgets, we can compare Fig. 5(a) and Fig. 5(d), where $\epsilon = 0.1$ and $\epsilon = 0.3$, respectively, on the synthetic dataset. According to the definition of differential privacy, a lower ϵ provides stronger privacy protection. We can observe that when ϵ is smaller, the test errors become relatively higher, which indicates a trade-off between differential privacy level and prediction accuracy. The same conclusion can be made on experiments on both the breast cancer dataset (Fig. 5(b) 5(e)) and the MNIST dataset (Fig. 5(c) 5(f)).

Impact of Byzantine Attacks Since Byzantine attacks launched by computing nodes can be effectively addressed as

mentioned in Section V, we focus on the impact of Byzantine data holders here. Specifically, we assume that Byzantine data holders send local gradients drawn from a Gaussian distribution with zero mean and an isotropic covariance matrix with the standard deviation of 200.

Taking synthetic data as an example, as shown in Fig. 5(a) and Fig. 5(d), the baseline system converges as the number of iterations reaches 200. The proposed *LearningChain* converges less quickly since it only selects several local gradients to aggregate. When there are several Byzantine attackers (5, 12), the model can still converge but with higher test errors. Once the number of Byzantine attackers exceeds the 51% threshold (20 malicious nodes out of 30 nodes), the system can hardly converge. Similar conclusions can be made on the breast cancer dataset (Fig. 5(b) and Fig. 5(e)) and MNIST data (Fig. 5(c) and Fig. 5(f)).

Comparison Between *LearningChain* and *LearningChainEx* Recall that *LearningChainEx* has a stronger differential privacy scheme. We can observe in Fig. 6 that the performance of *LearningChainEx* is comparable to that of *LearningChain*, and there is not much performance loss. It means that the differential privacy scheme in *LearningChainEx* is very efficient and effective.

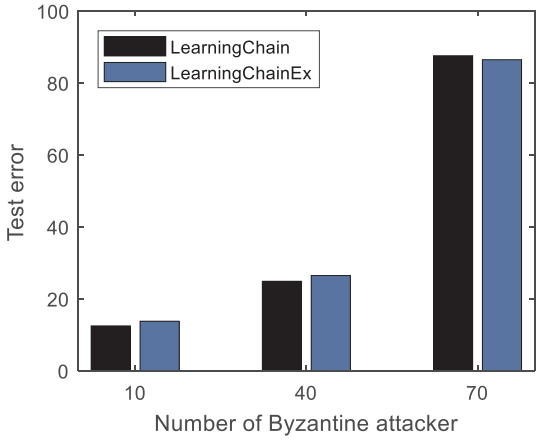


Fig. 6. Learning performances of *LearningChain* and *LearningChainEx*

Performance of l-nearest Gradients Aggregation We further explore the performance of the proposed l-nearest gradients aggregation scheme under Byzantine attack. Specifically, we compare it with one of the state-of-the-art algorithms, multi-Krum [20], on the MNIST dataset. In this experiment, we focus on the performance of Byzantine attack resilience and do not apply any differential privacy protection scheme.

The evaluation results are shown in Fig. 7. When only 10% (10 out of 100) of the data holders are Byzantine attackers, our proposed scheme and multi-Krum have similar performance. When the percentage of Byzantine data holders increases to 40% (40 out of 100), the l-nearest aggregation achieves a much lower test error than multi-Krum. If this percentage keeps growing to 70% (70 out of 100), both multi-Krum and the l-nearest aggregation are unable to remain resilient to Byzantine

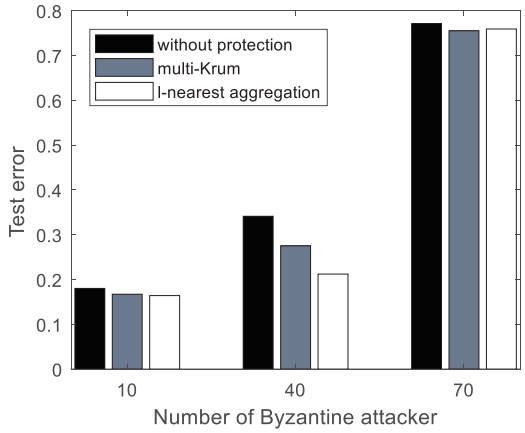


Fig. 7. Test Error for Byzantine Attack Tolerant Schemes

attacks. Note that multi-Krum is much more computationally expensive than l-nearest aggregation. Therefore, our proposed l-nearest aggregation is more efficient and effective when the majority of data holders are honest.

VII. Conclusions

In this paper, we have proposed *LearningChain*, which is general privacy-preserving and secure decentralized learning framework. Specifically, in *LearningChain*, we develop a differential privacy mechanism to protect data holders' data privacy in the local gradient computing process, and design the l-nearest aggregation scheme to defend against Byzantine attacks in the global gradients aggregation process. We also develop an extended differential privacy scheme that provides stronger privacy protection, resulting in *LearningChainEx*. Both theoretical analysis and experiment results demonstrate that our systems can efficiently and effectively learn models.

Acknowledgment

This work was partially supported by the US National Science Foundation under grants CNS-1602172 and CNS-1566479.

References

[1] R. J. McQueen, S. R. Garner, C. G. Nevill-Manning, and I. H. Witten, "Applying machine learning to agricultural data," *Computers and electronics in agriculture*, vol. 12, no. 4, pp. 275–293, 1995.

[2] X. Chen, J. Ji, T. Ji, and P. Li, "Cost-sensitive deep active learning for epileptic seizure detection," in *Proceedings of the 2018 ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*. ACM, 2018, pp. 226–235.

[3] X. Chen, J. Ji, K. Loparo, and P. Li, "Real-time personalized cardiac arrhythmia detection and diagnosis: A cloud computing architecture," in *Biomedical & Health Informatics (BHI), 2017 IEEE EMBS International Conference on*. IEEE, 2017, pp. 201–204.

[4] C. Luo, J. Ji, Q. Wang, X. Chen, and P. Li, "Channel state information prediction for 5g wireless communications: A deep learning approach," *IEEE Transactions on Network Science and Engineering*, 2018.

[5] M. L. de Prado, *Advances in financial machine learning*. John Wiley & Sons, 2018.

[6] V. S. Verykios, E. Bertino, I. N. Fovino, L. P. Provenza, Y. Saygin, and Y. Theodoridis, "State-of-the-art in privacy preserving data mining," *ACM Sigmod Record*, vol. 33, no. 1, pp. 50–57, 2004.

[7] X. Wu, X. Zhu, G.-Q. Wu, and W. Ding, "Data mining with big data," *IEEE transactions on knowledge and data engineering*, vol. 26, no. 1, pp. 97–107, 2014.
[8] W. Liao, C. Luo, S. Salinas, and P. Li, "Efficient secure outsourcing of large-scale convex separable programming for big data," *IEEE Transactions on Big Data*, 2017.
[9] S. Salinas, C. Luo, X. Chen, W. Liao, and P. Li, "Efficient secure outsourcing of large-scale sparse linear systems of equations," *IEEE Transactions on Big Data*, vol. 4, no. 1, pp. 26–39, 2018.
[10] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2015, pp. 1322–1333.
[11] A. C. Yao, "Protocols for secure computations," in *Foundations of Computer Science, 1982. SFCS'08. 23rd Annual Symposium on*. IEEE, 1982, pp. 160–164.
[12] J. Vaidya, C. Clifton, M. Kantarcioglu, and A. S. Patterson, "Privacy-preserving decision trees over vertically partitioned data," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 2, no. 3, p. 14, 2008a.
[13] J. Hamm, Y. Cao, and M. Belkin, "Learning privately from multiparty data," in *International Conference on Machine Learning*, 2016, pp. 555–563.
[14] A. Rajkumar and S. Agarwal, "A differentially private stochastic gradient descent algorithm for multiparty classification," in *Artificial Intelligence and Statistics*, 2012, pp. 933–941.
[15] Z. Ghodsi, T. Gu, and S. Garg, "Safetynets: Verifiable execution of deep neural networks on an untrusted cloud," in *Advances in Neural Information Processing Systems*, 2017, pp. 4675–4684.
[16] T.-T. Kuo and L. Ohno-Machado, "Modelchain: Decentralized privacy-preserving healthcare predictive modeling framework on private blockchain networks," *arXiv preprint arXiv:1802.01746*, 2018.
[17] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, pp. 1–32, 2014.
[18] K. Fanning and D. P. Centers, "Blockchain and its coming impact on financial services," *Journal of Corporate Accounting & Finance*, vol. 27, no. 5, pp. 53–57, 2016.
[19] P. Blanchard, R. Guerraoui, J. Stainer *et al.*, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Advances in Neural Information Processing Systems*, 2017, pp. 119–129.
[20] P. Blanchard, E. M. E. Mhamdi, R. Guerraoui, and J. Stainer, "Byzantine-tolerant machine learning," *arXiv preprint arXiv:1703.02757*, 2017.
[21] C. Xie, O. Koyejo, and I. Gupta, "Zeno: Byzantine-suspicious stochastic gradient descent," *arXiv preprint arXiv:1805.10032*, 2018.
[22] L. Chen, H. Wang, and D. Papailiopoulos, "Draco: Robust distributed training against adversaries," 2018.
[23] C. Xie, O. Koyejo, and I. Gupta, "Generalized byzantine-tolerant sgd," *arXiv preprint arXiv:1802.10116*, 2018.
[24] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.
[25] M. Crosby, P. Pattanayak, S. Verma, and V. Kalyanaraman, "Blockchain technology: Beyond bitcoin," *Applied Innovation*, vol. 2, pp. 6–10, 2016.
[26] T. T. A. Dinh, R. Liu, M. Zhang, G. Chen, B. C. Ooi, and J. Wang, "Untangling blockchain: A data processing view of blockchain systems," *arXiv preprint arXiv:1708.05665*, 2017.
[27] M. Jakobsson and A. Juels, "Proofs of work and bread pudding protocols," in *Secure Information Networks*. Springer, 1999, pp. 258–272.
[28] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An overview of blockchain technology: Architecture, consensus, and future trends," in *Big Data (BigData Congress), 2017 IEEE International Congress on*. IEEE, 2017, pp. 557–564.
[29] C. Dwork, "Differential privacy: A survey of results," in *International Conference on Theory and Applications of Models of Computation*. Springer, 2008, pp. 1–19.
[30] A. Lei, H. Cruickshank, Y. Cao, P. Asuquo, C. P. A. Ogah, and Z. Sun, "Blockchain-based dynamic key management for heterogeneous intelligent transportation systems," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 1832–1843, 2017.
[31] D. Dheeru and E. Karra Taniskidou, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml
[32] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist/
[33] G. Wood, "Ethereum: a secure decentralized transaction ledger (2014)," 2017.
[34] Z. Li, J. Kang, R. Yu, D. Ye, Q. Deng, and Y. Zhang, "Consortium blockchain for secure energy trading in industrial internet of things," *IEEE Transactions on Industrial Informatics*, 2017.
[35] S. Hu, C. Cai, Q. Wang, C. Wang, X. Luo, and K. Ren, "Searching an encrypted cloud meets blockchain: A decentralized, reliable and fair realization," 2018.
[36] S. Wang, Y. Zhang, and Y. Zhang, "A blockchain-based framework for data sharing with fine-grained access control in decentralized storage systems," *IEEE Access*, vol. 6, pp. 38 437–38 450, 2018.

APPENDIX A

Theorem The L_2 sensitivity of the algorithm that computes $\nabla g_k(\mathbf{w}_t)$ from D_k is at most $2G$.

Proof. Let D_k, D'_k be any two data sets differing in a single element. Without loss of generality, we assume that D_k and D'_k differ only in their last element, with $D_k = \{(\mathbf{x}_i^k, \mathbf{y}_i^k)\}_{i=1}^{N_k}$ and $D_k = \{(\mathbf{x}_1^k, \mathbf{y}_1^k), \cdots, (\mathbf{x}_{N_k-1}^k, \mathbf{y}_{N_k-1}^k), (\mathbf{x'}_{N_k}^k, \mathbf{y'}_{N_k}^k)\}$. Let $\nabla g'_k(\mathbf{w}_t)$ denote the gradient at $\mathbf{w}_t$ of the cumulative loss on D'_k.

By definition, for datasets D_k and D'_k we have

$$\nabla g_k(\mathbf{w}_t) = \sum_{i=1}^{N_k} \frac{d}{d\mathbf{w}_t}\phi(h(\mathbf{w}_t; \mathbf{x}_i^k), \mathbf{y}_i^k)$$

and

$$\nabla g'_k(\mathbf{w}_t) = \sum_{i=1}^{N_k-1} \frac{d}{d\mathbf{w}_t}\phi(h(\mathbf{w}_t; \mathbf{x}_i^k), \mathbf{y}_i^k) + \frac{d}{d\mathbf{w}_t}\phi(h(\mathbf{w}_t; \mathbf{x'}_{N_k}^k), \mathbf{y'}_{N_k}^k)$$

Then we have

$$\begin{aligned}
&\|\nabla g_k(\mathbf{w}_t) - \nabla g'_k(\mathbf{w}_t)\|_2 \\
&= \|\frac{d}{d\mathbf{w}_t}\phi(h(\mathbf{w}_t; \mathbf{x}_{N_k}^k), \mathbf{y}_{N_k}^k) - \frac{d}{d\mathbf{w}_t}\phi(h(\mathbf{w}_t; \mathbf{x'}_{N_k}^k), \mathbf{y'}_{N_k}^k)\|_2 \\
&\leq \|\frac{d}{d\mathbf{w}_t}\phi(h(\mathbf{w}_t; \mathbf{x}_{N_k}^k), \mathbf{y}_{N_k}^k)\|_2 + \|\frac{d}{d\mathbf{w}_t}\phi(h(\mathbf{w}_t; \mathbf{x'}_{N_k}^k), \mathbf{y'}_{N_k}^k)\|_2 \\
&\leq 2G
\end{aligned}$$

where the first inequality directly follows triangle inequality, and the second inequality is because we impose a cap function to ensure that each gradient $\frac{d}{d\mathbf{w}_t}\phi(h(\mathbf{w}_t; \mathbf{x}_i^k), \mathbf{y}_i^k)$ is upper bounded by G.