This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2020.2987843, IEEE Internet of Things Journal

AI at the Edge: Blockchain-Empowered Secure Multiparty Learning with Heterogeneous Models

Qianlong Wang, *Student Member, IEEE,* Yifan Guo, *Student Member, IEEE,* Xufei Wang, *Student Member, IEEE,* Tianxi Ji, *Student Member, IEEE,* Lixing Yu, *Student Member, IEEE,* and Pan Li, *Member, IEEE*

***Abstract*—Edge computing, an emerging computing paradigm pushing data computing and storing to network edges, enables many applications that require high computing complexity, scalability, and security. In the big data era, one of the most critical applications is multiparty learning or federated learning, which allows different parties to collaborate with each other to obtain better learning models without sharing their own data. However, there are several main concerns about the current multiparty learning systems. First, most existing systems are distributed and need a central server to coordinate the learning process. However, such a central server can easily become a single point of failure and may not be trustworthy. Second, although quite a few schemes have been proposed to study Byzantine attacks, a very common and challenging kind of attack in distributed systems, they generally consider the scenario of learning a global model. However, in fact, all parties in multiparty learning usually have their own local models. The learning methods and security issues in this case are not fully explored. In this paper, we propose a novel blockchain-empowered decentralized secure multiparty learning system with heterogeneous local models called BEMA. Particularly, we consider two types of Byzantine attacks, and carefully design "off-chain sample mining" and "on-chain mining" schemes to protect the security of the proposed system. We theoretically prove the system performance bound and resilience under Byzantine attacks. Simulation results show that the proposed system obtains comparable performance with that of conventional distributed systems, and bounded performance in the case of Byzantine attacks.**

***Index Terms*—Multiparty learning, security, blockchain, decentralized network, heterogeneous models.**

I. Introduction

Edge computing is an emerging computing paradigm and has attracted intensive attention in recent years. It enables data storing, computing, and analytics to be deployed at the network edge, thus facilitating various intelligent applications for the Internet of Things, smart city, etc. In the big data era, one of the most critical intelligent applications is multiparty learning or federated learning [1]. In particular, as institutions, companies, and smart devices are collecting huge amounts of data every day, it is inefficient and insecure to collect and learn on all the data together at a single location. In contrast, in multiparty learning, the learning is conducted in a distributed fashion, where each party can keep their data local.

Most existing multiparty learning systems only focus on training a global model for all the parties, ignoring the fact that a local model may have already been trained at each party based on its own dataset. Effectively utilizing these local models can substantially improve the training efficiency of multiparty learning. For example, McMahan et al. [2] and Li et al. [3] use model averaging in multiparty learning, and further propose Byzantine attack resilient model averaging methods. However, these works assume that the local models at each party are homogeneous, which may not be practical. A few works try to address this issue. For instance, Wu et al. [4] propose a multiparty multiclass margin to calibrate heterogeneous local models in the system. Note that all these works rely on a trusted central server to coordinate the distributed learning process, which obviously becomes a single point of failure and can be subject to attacks. Only a couple of works like [5] attempt to design decentralized multiparty learning systems. Note that only linear models are considered in [5]. To the best of our knowledge, secure decentralized multiparty learning with heterogeneous models remains an open and challenging problem.

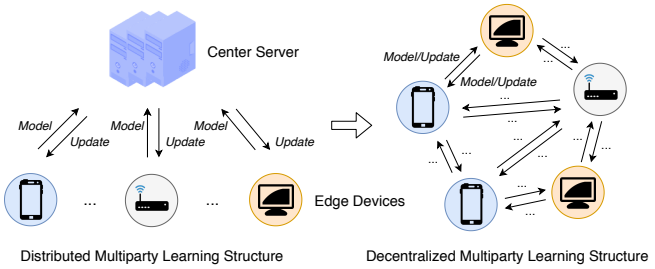


Figure 1: Different structures for distributed and decentralized multiparty learning.

In this paper, we propose a novel secure decentralized multiparty learning system by taking advantage of the blockchain technology, called BEMA. In particular, each party in a decentralized system broadcasts its local model, and meanwhile, processes the received (heard) models from other parties over his local dataset, and identifies the models that need to be calibrated. Following our designed protocol, the party sends the calibration message to the corresponding parties. In so doing, the parties in the system do not need to share their whole dataset with other parties. In this system, we consider two types of Byzantine attacks in the system, which can occur in model broadcasting and model calibration processes. To protect system security, we carefully design "off-chain sample mining" and "on-chain mining" schemes. Theoretical analysis is performed to show that the proposed system has bounded performance under Byzantine attacks. The main contributions

Q. Wang, Y. Guo, X. Wang, T. Ji, L. Yu and P. Li are with the Department of Electrical, Computer and Systems Engineering, Case Western Reserve University, Cleveland, OH, 44106, email: {qxw204, yxg383, xxw512, txj116, lxy257, pxl288}@case.edu.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2020.2987843, IEEE Internet of Things Journal

of this paper are summarized as follows:

- We propose a novel blockchain-empowered decentralized secure multiparty learning system called BEMA, where learning parties hold heterogeneous local models.
- We formulate two types of Byzantine attacks in the system, and devise secure "off-chain sample mining" and "on-chain mining" schemes to defend against the attacks.
- We theoretically analyze the performance of the proposed system and prove that it is bounded under Byzantine attacks.
- We evaluate the system with the real-world dataset. We show that the proposed system has a comparable performance compared with traditional multiparty learning with a central server, and is resilient to Byzantine attacks.

The rest of the paper is organized as follows. We introduce problem formulation and detail our proposed blockchain-empowered secure multiparty learning system in Section II and Section III, respectively. We conduct simulations to evaluate our system's efficiency and resilience to Byzantine attacks in Section IV. Related work is summarized in Section V. Finally, we conclude the paper in Section VI.

II. Problem Formulation

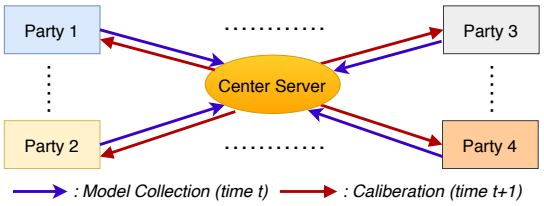
A. System Overview

We consider that there are N parties in a decentralized learning system. Each party $i \in [1, N]$ has its own local dataset $\mathcal{D}_i = \{\mathcal{X}_i, \mathcal{Y}_i\} = \{(x_i^1, y_i^1), \ldots, (x_i^{n_i}, y_i^{n_i})\}$ containing n_i data samples and their labels. Besides, for any party $i \in [1, N]$, we have $D_i \subset D$, where $\mathcal{D} = \{\mathcal{X}, \mathcal{Y}\}$ is the global dataset and $\mathcal{Y} = \{1, \ldots, C\}$ denotes the set of total C classes. In the learning system, each party i holds a local model f_i that is trained on $\mathcal{D}_i$. Since the classifier f_i is trained on only a partial set of $\mathcal{D}$, it may misclassify an unseen data sample x into a wrong class $y' \in \mathcal{Y}_i$, while its real class is $y \notin \mathcal{Y}_i$. Therefore, each party needs to learn a robust local model that can accurately classify not only unseen data belonging to its learned space, i.e., $y \in \mathcal{Y}_i$ but also data from unknown space, i.e., $y \notin \mathcal{Y}_i$. Particularly, for each party i, its classifier $f_i : \mathcal{X}_i \rightarrow \mathcal{Y}_i$ is obtained by training a local model $\mathcal{M}_i$ with model parameter θ_i, which can be different from those at other parties, e.g., linear regression, Support Vector Machine (SVM), or Artificial Neural Network (ANN). Given a data sample x, the output of f_i is

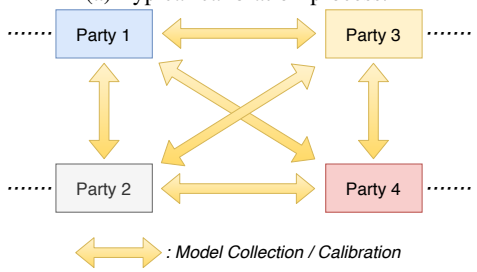
$$f_i(x) = \arg\max_{y \in \mathcal{Y}_i} h_i(\theta_i, x, y)$$

where $h_i(\theta_i, x, y)$ is a score function for the local model f_i that returns the predicted possibility (or confidence score) of y being the true label of x. Note that $h_i(\theta_i, x, y) = 0$ for $y \notin \mathcal{Y}_i$. In the next few sections, we slightly abuse the notation to use $h_i(x, y)$ or $h_i(\theta_i)$ instead when θ_i or (x, y) is given for brevity.

As shown in Fig. 2, a traditional multiparty learning system relies on a central server to coordinate the learning process and calibrate local models, which obviously becomes a single point of failure. Instead, we investigate a blockchain empowered decentralized multiparty learning system without any central server.



(a) Typical calibration process.



(b) Proposed decentralized calibration.

Figure 2: Typical vs. decentralized local model update processes.

A common model calibration method used for multiparty learning with a central server is called multiparty multiclass margin (MPMC-margin) [4]. Before we introduce it, we first present the following definition of the max model predictor.

Definition 1. Max-Model Predictor. Given a set of models $F = \{f_1, \ldots, f_N\}$, the max-model predictor f_F is defined as

$$f_F(x) = \arg\max_{y \in \mathcal{Y}, i \in [1,N]} h_i(x, y).$$

Definition 2. MPMC-margin. For a data sample (x, y), the MPMC-margin is defined as follows

$$\rho(x, y, y^-) = h_a(x, y) - h_b(x, y^-),$$

where y^- is a wrong class label for x, and

$$a = \arg\max_i h_i(x, y), y \in \mathcal{Y}_i,$$
$$b, y^- = \arg\max_{i, y'} h_i(x, y'), y' \in \mathcal{Y}_i \setminus \{y\}.$$

In the MPMC-margin scheme, every party in the learning system has knowledge of all the other parties' current learning models. In each iteration, each party exploits its own local data to find a valid calibration sample (x, y, y^-) such that $\rho \leq 0$, and sends the sample to both parties a and b found according to Definition 2. Through different model update methods, party a aims to amplify the importance of (x, y) by increasing $h_a(x, y)$, while party b decreases $h_b(x, y^-)$. After iterations of model calibration, all the parties can update their local models and finally employ the max-model predictor to make classifications.

B. Threat Model

One of the most common and challenging attacks in decentralized systems is Byzantine attack, where an attacker follows the system protocol but propagates arbitrary malicious information to benign system participants, aiming to degrade

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2020.2987843, IEEE Internet of Things Journal

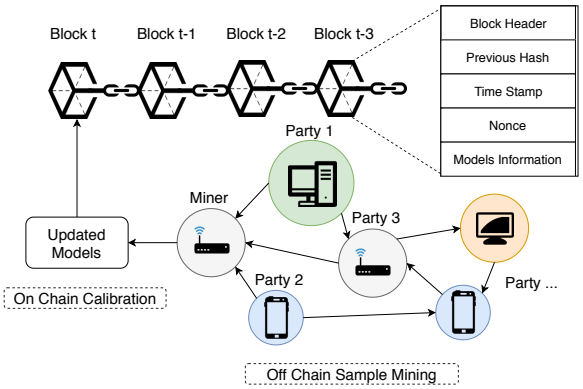


Figure 3: The architecture of blockchain-empowered secure multiparty learning.

the system performance and further mislead or control the system output. The main processes in the proposed system include model collection (i.e., collecting local models from other parties) and model update (i.e., sending calibration messages to particular parties if needed). Correspondingly there exist the following two types of Byzantine attacks.

In type I Byzantine attacks, a malicious party broadcasts a malicious local model to other parties in order to change the classification result of the max-model predictor. Particularly, consider a malicious party $\mathcal{B}$ with a model of $f^{\mathcal{B}}$, where for a sample (x_j, y_j), we have

$$h^{\mathcal{B}}(x_j, y^{\mathcal{B}}) > \max_{i \in \mathcal{H}} h_i(x_j, y_j), \ y^{\mathcal{B}} \in \mathcal{Y}_{\mathcal{B}}.$$

$\mathcal{H}$ is the set of benign parties. In this case, the benign parties that receive the model $f^{\mathcal{B}}$ from party $\mathcal{B}$ will misclassify x_j into a wrong class $y^{\mathcal{B}}$ when applying the max-model predictor.

In type II Byzantine attacks, an attacker $\mathcal{B}$ sends a malicious calibration message to particular parties in order to mislead their local model updating process. For example, suppose that the attacker $\mathcal{B}$ is able to find a class y^- based on its private sample (x, y), where

$$\rho(x, y^-, y) = h_a(x, y^-) - h_b(x, y) < 0, y^- \in \mathcal{Y}_a, y \in \mathcal{Y}. \quad (1)$$

Then, the attacker $\mathcal{B}$ can send a malicious calibration message (x, y^-, y) to both parties a and b, so that they will be misled to increase the prediction score on the wrong class and lower the prediction score on the right class, respectively.

III. Blockchain-Empowered Secure Multiparty Learning (BEMA)

Given the threat model described above, the main challenges in building decentralized multiparty learning are first, to verify the truthfulness of the broadcasted local models of each party, which can affect the max model predictor's output, and second, to verify the truthfulness of the calibration messages, which is critical for updating the local models. To address these challenges, we propose a blockchain-empowered secure multiparty learning system called BEMA, which is detailed in the following.

A. System Architecture

The main architecture of BEMA is shown in Fig. 3. In the system, each party could be an association/institution, or a personal device, like a smartphone or a personal computer. Each party holds a local model, which is trained on its private dataset. As the local models are heterogeneous, the blockchain stores model information, including model type, model parameter, and learned classes. In each time slot, each party can test the models on the chain and search for a valid calibration data sample (x, y, y^-) in its own dataset. If a valid sample is found, each party can broadcast it to the blockchain and win certain system rewards once a miner uses it to update models on the chain. This process is called "off-chain sample mining". The miners in the system collect the calibration samples broadcasted by the learning parties and check the validity of them based on our designed protocol. Once a sample is validated by the miner who wins the authority to write the next block, the miner uses it to update models on the chain, which is called the calibration process. Based on different predefined model updating protocols, the miner computes updated model information, constructs a legal block header, and creates a new block. All of the sample validation, calibration, and block construction processes are called "on-chain mining". Note that all the processes in on-chain mining are carefully designed, which will be described next. Furthermore, system rewards can be provided by system initializers, e.g., hospitals that would like to obtain better learning models for medical applications. Since the model update information stored in a block can be viewed and verified by each system participant, the miners, including both off-chain and on-chain ones, can obtain rewards as the new block is built.

B. System Entities

There are three types of participants in the system, which are the original party, regular party, and miner. The details of their roles are given as follows.

1) Original Party (OP): The OPs are the parties or associations that start the blockchain. Each OP has its local model and intends to work with each other to learn better local models without directly sharing all its local data. All the OPs are considered to be honest in the system.

2) Regular Party (RP): The RPs are the participants of the system after the initialization. The same as the OPs, each RP owns a local model trained on its own dataset. The number of OPs and their data amount could be limited. As more RPs join the system, they can help the system learn more robust local models for each party and enable all parties to perform more accurate classifications on the data from unseen classes. RPs are not trustworthy and can potentially launch attacks on the system.

3) Miner: A miner's main job is to build blocks on the blockchain based on the consensus protocol. Here, we adopt Proof-of-Work [6] as the system consensus protocol. A miner could be either an OP/RP with sufficient computing resources, or an edge device.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2020.2987843, IEEE Internet of Things Journal

C. Secure Multiparty Learning

BEMA mainly consists of system initialization, off-chain sample mining, and on-chain mining. In system initialization, the OPs register their identities (IDs) and model information on the chain. Afterward, the participating RPs can register their IDs and model information on the chain. The model calibration process is mostly the off-chain and on-chain mining. The former is to find eligible data samples for model calibration, while the latter is to calibrate specific local models based on the found samples and record them in the new blocks. More details are given as follows.

1) Initialization: For blockchain initialization, consider there are a total of $M(M < N)$ OPs, each of whom holds a local model $h_{Y_i}^{OP}$, where $i \in [1, M]$, $Y_i \subset \mathcal{Y}$. Here, h_{Y_i} represents the model trained on the data from classes of Y_i. Each OP i broadcasts its model information along with its ID, i.e., $\{h_{Y_i}^{OP}, Y_i, ID_{OP_i}\}$. Thus, the first block contains the model information from all OPs, which is $< \{h_{Y_1}^{OP}, Y_1, ID_{OP_1}\} \ldots, \{h_{Y_M}^{OP}, Y_M, ID_{OP_M}\} >$.

Denote the set of RPs by $\mathcal{R} = \{1, 2, \ldots, R\}$. Each RP needs to have its local model verified by the chain before it can register its model on the chain. Particularly, for a new participant RP j owning data from classes of $Y_j (j \in \mathcal{R})$ and has trained a local model $h_{Y_j}^{RP}$, any OP i with $Y_j \subset Y_i$ can validate RP j's model by checking if the following holds

$$\sum_{(x,y)\in\mathcal{D}_{OP_j}} |h_{Y_j}^{RP}(x, y) - h_{Y_i}^{OP}(x, y)| \leq \omega, \tag{2}$$

where ω is a pre-defined threshold. If the above condition holds, the OP will sign the approval for initializing the model $h_{Y_j}^{RP}$ for RP j on the chain. Then, a message $< h_{Y_j}^{RP}, ID_{RP_j}, Sign_{OP_i} >$ is broadcasted, which will be included in the new block on the chain. Otherwise, RP j's model is considered to be unqualified for classifying on Y_j, and cannot be registered in the system.

2) Off-Chain Sample Mining: In off-chain sample mining, each party in the system is encouraged to exploit the local models on the chain with their own dataset, aiming to find a data sample that can help calibrate some of the models. Different from the original MPMC-margin, we propose "secure MPMC-margin" for decentralized systems, which can help defend against Byzantine attacks.

Definition 3. Secure MPMC-margin. For a data sample (x, y), the secure MPMC-margin is defined as follows

$$\rho(x, y, y^-) = h_{a^*}(x, y) - h_{b^*}(x, y^-), y^- \in \mathcal{Y} \setminus \{y\}, \tag{3}$$

$$h_{a^*}(x, y) = \frac{1}{m} \sum_{k \xrightarrow{m} a} h_k(x, y), \tag{4}$$

$$h_{b^*}(x, y^-) = \frac{1}{m} \sum_{k \xrightarrow{m} b} h_k(x, y^-), \tag{5}$$

$$a = \arg\min_{j\in\mathcal{S}(y)} \sum_{k \xrightarrow{m} j} |h_k(x, y) - h_j(x, y)|, \tag{6}$$

$$y^-, b = \arg\max_{y^-\in\mathcal{Y}\setminus\{y\}} \arg\min_{j\in\mathcal{S}(y^-)} \sum_{k \xrightarrow{m} j} |h_k(x, y^-) - h_j(x, y^-)|. \tag{7}$$

Here, $k \xrightarrow{m} j$ means the m closest value to $h_j(\cdot, \cdot)$, where m equals to $\alpha|\mathcal{S}(y)|$ and $\alpha|\mathcal{S}(y^-)|$ in Eq. (6) and Eq. (7), respectively. $\mathcal{S}(y)$ and $\mathcal{S}(y^-)$ represent the set of parties that hold data in class y and in class y^-, respectively. $\alpha (0 < \alpha \leq 1)$ is a control parameter.

In the original MPMC-margin given in Definition 2, only two models, a and b, are selected to calculate the margin, which get further updated during the calibration process. However, this may not be appropriate in a decentralized system for two reasons. First, as there might be malicious parties in the system, only selecting two models may make the margin calculation process subject to attacks and compromise benign models. Second, based on the original MPMC-margin, only two models are updated in each calibration round, which would be inefficient in a large-scale system. In contrast, in secure MPMC-margin, we adopt the Krum function [7] in Eq. (6) and Eq. (7) to mitigate the influences from malicious models. Note that, the amount of samples needed for updating all the models is strictly limited as shown in [4], where a small budget (less than 1% of the global data) is sufficient to reach satisfactory results in the experiments.

3) On-Chain Mining: As mentioned above, each broadcasted sample (x, y, y^-) needs to be validated by the miner and then used for model calibration.

a) Sample Validation: For a received calibration sample (x, y, y^-), the miner checks its validity by

$$\rho(x, y, y^-) \leq 0, \tag{8}$$

$$h_{a^*}(x, y) \geq \delta, \tag{9}$$

$$\max_{k \xrightarrow{m} a} |h_k(x, y) - h_{a^*}(x, y)| \leq \sigma, \tag{10}$$

where δ and σ are predefined thresholds. Eq. (8) assures that (x, y, y^-) is eligible for calibration. Eq. (9) and Eq. (10) quantify the confidence level that the sample x belongs to class y. If all the above constraints are satisfied, the miner adopts the sample for model calibration.

b) Model Calibration: With a valid calibration sample (x, y, y^-), the miner updates the corresponding models, particularly their model parameters θ_i's, on the chain. The calibration process may vary for different models due to their heterogeneity. We assume that the models support online update, which is the case for most popular learning models such as linear regression, SVM, and ANN. We also assume that the score functions $h_i(\cdot, \cdot, \cdot) (i \in [1, N])$ are continuously differentiable (we use $h_i(\theta_i)$ in the following to simplify notations). The miner performs $enhance(x, y)$ and $impair(x, y^-)$ functions on $\mathcal{S}(y)$ and $\mathcal{S}(y^-)$ models that need to be updated by employing the gradient descent method [8], [9], [10]. Note that the same as in our previous work [10], we put a cap $G (G > 0)$ on the gradient to avoid the gradient exploding problem, i.e.,

$$\Delta_i(\theta_i^t) = \Delta_i(\theta_i^t) / \max(1, \frac{||\Delta_i(\theta_i^t)||_2}{G}).$$

Specifically, the $enhance(x, y)$ function is defined as

$$\forall h_i(x, y) < h_{a^*}(x, y), i \in \mathcal{S}(y),$$
$$\theta_i^{t+1} = \theta_i^t + \eta \Delta_i(\theta_i^t), \tag{11}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2020.2987843, IEEE Internet of Things Journal

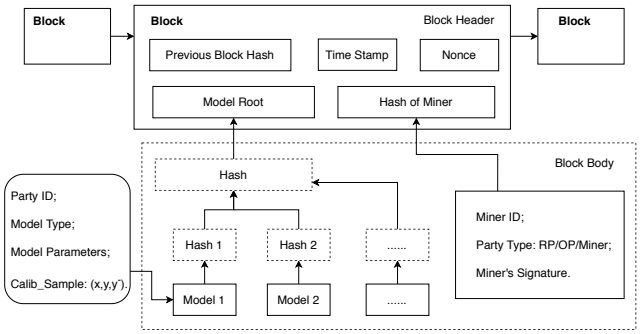


Figure 4: The structure of blocks.

where $\eta > 0$ is a learning rate, and $t+1$ and t represent after and before update, respectively. This function aims to increase the prediction score of (x, y) for the corresponding models.

Similarly, the $impair(x, y^-)$ function is defined as

$$\forall h_i(x, y^-) > h_{b^*}(x, y^-), i \in \mathcal{S}(y^-),$$
$$\theta_i^{t+1} = \theta_i^t - \eta \Delta_i(\theta_i^t), \tag{12}$$

which aims to decrease the prediction score of (x, y^-) for the corresponding models.

c) Block Creation: The structure of the blocks is shown in Fig. 4. For each received valid calibration sample, the miner updates the corresponding models and writes the updated model information in the created block, where the information includes the ID of the party that the model belongs to, the type of the model, model parameters θ, and the used calibration sample (x, y, y^-). The model information is stored in a Merkle tree [11] using a double SHA256 [12]. Besides, the miner also uploads its own information into the block, including its ID and party type (RP/OP/Miner), by using a double SHA-256. Thus, previous block's hash, time stamp, a nonce, the Merkle tree root, and the hash of the miner's information are included in the block header.

We summarize the main processes of the proposed BEMA system in Algorithm 1.

D. Krum based Model Predictor

After the above secure multiparty learning, each OP/RP can use the models on the chain to classify unseen data samples. A common method is to apply the max-model predictor given in Definition 1. However, as the max-model predictor finally only selects a single model for classification, such a regular max-model predictor may fail in the scenario that not every model in the system is trustworthy. Therefore, we propose the following Krum based model predictor, which exploits all the related models for an unseen data sample in order to give a reliable result.

Definition 4. Krum Based Model Predictor. Given a set of model $F = \{f_1, \ldots, f_N\}$, the Krum based model predictor f_{Krum} is defined as

$$f_{Krum}(x) = \arg\max_{y \in \mathcal{Y}} h_i(x, y), \tag{13}$$
$$i = \arg\min_{j \in \mathcal{S}(y)} \sum_{k \xrightarrow{m} j} |h_k(x, y) - h_j(x, y)|. \tag{14}$$

Algorithm 1 Blockchain-empowered Secure Multiparty Learning (BEMA)

1: **Model initialization:**
2: Each OP i broadcasts its model information along with its ID, i.e., $\{h_{Y_i}^{OP}, Y_i, ID_{OP_i}\}$ so as to build the first block.
3: Each RP j broadcasts its local model $h_{Y_j}^{RP}$ to OPs.
4: The OPs check the validity of received models based on Eq. (2), and broadcast approval messages $< h_{Y_j}^{RP}, ID_{RP_j}, Sign_{OP_i} >$ that will be included in the next block.
5: **Off-Chain mining:**
6: Each OP/RP finds a valid calibration sample (x, y, y^-) based on Eq. (3)-Eq. (7).
7: Each OP/RP broadcasts the found calibration sample (x, y, y^-).
8: **On-Chain mining:**
9: Sample validation.
10: The miner validates each received data sample (x, y, y^-) based on Eq. (8)-Eq. (10).
11: Model calibration.
12: The miner updates the corresponding models on the blockchain based on Eq. (11) and Eq. (12).
13: Block creation.
14: The miner builds a new block as shown in Fig. 4.
15: The miner broadcasts the new block.

E. System Analysis

1) Performance Analysis: We present the following theorem to provide insight into the efficacy of our secure multiparty learning process.

Theorem 1. *Assume that 1) each model $h_i(\theta_i)(i \in [1, N])$ is convex, continuously differentiable, and $c_i - lipschitz(c_i > 0)$, and 2) $p_i \leq |\nabla h_i(\theta_i)| \leq b_i$. The secure MPMC-margin improvement after sample calibration at time slot t is bounded by*

$$\frac{2\eta}{m} \min_{i \in [1,N]} \{p_i^2\} \leq \rho^{t+1} - \rho^t \leq$$
$$2\eta \max_{i \in [1,N]} \{c_i\} \min\{ \max_{i \in [1,N]} \{b_i\}, G\}. \tag{15}$$

Proof. For brevity, we rewrite secure MPMC-margin in Eq. (3) into the following form at time slot t

$$\rho^t = h_{a^*}^t - h_{b^*}^t.$$

The improvement after sample calibration at time slot t is

$$\rho^{t+1} - \rho^t = (h_{a^*}^{t+1} - h_{a^*}^t) + (h_{b^*}^t - h_{b^*}^{t+1}),$$

where the first term and the second term correspond to the $enhance$ and $impair$ processes, respectively.

First, we consider the $enhance$ process. According to Eq. (11), for $h_i(\cdot)$ where $i \in \mathcal{S}(y)$, given a calibration sample (x, y, y^-), we have

$$h_i^{t+1} - h_i^t \geq \nabla h_i(\theta^t)(\theta_i^{t+1} - \theta_i^t)$$
$$= \eta(\nabla h_i(\theta^t))^2,$$

due to the convexity of $h_i(\cdot)$. Besides, since $h_i(\theta_i)$ is $c_i - Lipschitz$, we have

$$\begin{aligned} h_i^{t+1} - h_i^t \leq c_i|\theta_i^{t+1} - \theta_i^t| &= c_i\eta|\nabla h_i(\theta_i^t)| \\ &\leq c_i\eta \min_{i\in[1,N]}\{b_i, G\}. \end{aligned}$$

Similarly, for the $impair$ process, for $h_i(\cdot)$ where $i \in \mathcal{S}(y^-)$, we get

$$\begin{aligned} h_i^t - h_i^{t+1} &\geq \eta(\nabla h_i(\theta_i^t))^2 \\ h_i^t - h_i^{t+1} &\leq c_i\eta \min_{i\in[1,N]}\{b_i, G\}. \end{aligned}$$

Note that at least one model, say $h_j(\cdot)(j \in \mathcal{S}(y))$, gets enhanced based on the criterion in Eq. (11), and at least one model, say $h_k(\cdot)(k \in \mathcal{S}(y^-))$, gets impaired based on the criterion in Eq. (12). Therefore, we can obtain

$$h_{a^*}^{t+1} \geq \frac{1}{m}(\sum_{i\in M_a} h_i^t + \eta(\nabla h_j(\theta_j^t))^2) = h_{a^*}^t + \frac{\eta}{m}(\nabla h_j(\theta_j^t))^2,$$

$$h_{b^*}^{t+1} \leq \frac{1}{m}(\sum_{i\in M_b} h_i^t - \eta(\nabla h_k(\theta_k^t))^2) = h_{b^*}^t - \frac{\eta}{m}(\nabla h_k(\theta_k^t))^2,$$

where M_a and M_b represent the set of M models used for calculating $h_{a^*}^t$ and for calculating $h_{b^*}^t$, respectively.

Consequently, we get

$$\begin{aligned} \rho^{t+1} - \rho^t &\geq \frac{\eta}{m}(\nabla h_j(\theta_j^t))^2 + \frac{\eta}{m}(\nabla h_k(\theta_k^t))^2 \\ &\geq \frac{2\eta}{m} \min_{i\in[1,N]}\{p_i^2\}. \end{aligned}$$

Moreover, note that at most all models in $\mathcal{S}(y)$ get enhanced and at most all models in $\mathcal{S}(y^-)$ get impaired. Thus, we get

$$\begin{aligned} h_{a^*}^{t+1} &\leq \frac{1}{m}(\sum_{i\in M_a}(h_i^t + c_i\eta|\nabla h_i(\theta_i^t)|)) \\ &= h_{a^*}^t + \frac{\eta}{m}\sum_{i\in M_a} c_i|\nabla h_i(\theta_i^t)| \\ &\leq h_{a^*}^t + \frac{\eta}{m}\max_{i\in[1,N]}\{c_i\}\cdot m \cdot \min\{\max_{i\in[1,N]}\{b_i\}, G\} \\ &= h_{a^*}^t + \eta \max_{i\in[1,N]}\{c_i\}\min\{\max_{i\in[1,N]}\{b_i\}, G\}, \end{aligned}$$

$$\begin{aligned} h_{b^*}^{t+1} &\geq \frac{1}{m}(\sum_{i\in M_b}(h_i^t - c_i\eta|\nabla h_i(\theta_i^t)|)) \\ &\geq h_{b^*}^t - \eta \max_{i\in[1,N]}\{c_i\}\min\{\max_{i\in[1,N]}\{b_i\}, G\}. \end{aligned}$$

Consequently, we have

$$\rho^{t+1} - \rho^t \leq 2\eta \max_{i\in[1,N]}\{c_i\}\min\{\max_{i\in[1,N]}\{b_i\}, G\}.$$

□

2) Security Analysis: As we mentioned in Section II-B, there are two types of Byzantine attacks in decentralized multiparty learning. For type I Byzantine attacks, we take advantage of the OPs to validate each RP's model as shown in Eq. (2). The control parameter ω indicates the requirement on the model of RP j requesting to join the system. In particular, RP j's model, i.e., $h_{Y_j}^{RP}$, is required to perform similarly to the corresponding OP's local models that include Y_j in their learning spaces. Furthermore, with the designed blockchain based multiparty learning system, the model updating process is public and can be verified by all the participants on the blockchain. Thus, the system can ensure the validity of each model's updating process, even though the newly registered RP's model is, to some extent, malicious at the beginning.

Type II Byzantine attacks are another main security concern of the proposed system, where the bogus calibration sample may severely disrupt the model update process. We show that the proposed secure MPMC-margin has bounded performance under type II Byzantine attacks in following theorems.

Theorem 2. *Consider the set of models $\mathcal{S}(y) = \{h_1, \ldots, h_{|\mathcal{S}(y)|}\}$ that aim to classify data in class y. Let $|\mathcal{S}(y)| = K^{\mathcal{S}(y)} + f^{\mathcal{S}(y)}$, where $K^{\mathcal{S}(y)}$ and $f^{\mathcal{S}(y)}$ represent the number of honest models and that of malicious models, respectively. Assume $2f^{\mathcal{S}(y)} < m < K^{\mathcal{S}(y)}$, where m is defined in Definition 3. Given a data example (x, y), let $h_{min}(x, y)$ and $h_{max}(x, y)$ denote the minimum and maximum output of honest models, respectively. Then, $h_{a^*}(x, y)$ in Eq. (4) in the worst case is bounded by*

$$\begin{aligned} h_{min}(x,y) - \frac{f^{\mathcal{S}(y)}}{m}(h_{max}(x,y) - h_{min}(x,y)) &< h_{a^*}(x,y) \\ &< h_{max}(x,y) + \frac{f^{\mathcal{S}(y)}}{m}(h_{max}(x,y) - h_{min}(x,y)). \end{aligned} \tag{16}$$

Similarly, consider the set of models $\mathcal{S}(y^-)$ that aim to classify data in class y^-. Given data sample (x, y^-), $h_{b^*}(x, y)$ in Eq. (5) in the worst case is bounded by

$$\begin{aligned} h_{min}(x,y^-) - \frac{f^{\mathcal{S}(y^-)}}{m}(h_{max}(x,y^-) - h_{min}(x,y^-)) &\\ < h_{b^*}(x,y) &< \\ h_{max}(x,y^-) + \frac{f^{\mathcal{S}(y^-)}}{m}(h_{max}(x,y^-) - h_{min}(x,y^-)). \end{aligned} \tag{17}$$

Proof. We present the proof for Eq. (16) in the following, while the proof for Eq. (17) directly follows. To simplify the notations, we omit the subscripts and superscripts in $K^{\mathcal{S}(y)}$ and $f^{\mathcal{S}(y)}$.

Let $H^{\mathcal{B}} = \{h_1^{\mathcal{B}}, \ldots, h_f^{\mathcal{B}}\}$ denote all the byzantine models (malicious models) in $\mathcal{S}(y)$, where $h_l^{\mathcal{B}}(x, y) \leq h_{l+1}^{\mathcal{B}}(x, y), l \in [1, f-1]$, and $H^{\mathcal{O}} = \{h_1^{\mathcal{O}}, \ldots, h_K^{\mathcal{O}}\}$ denote all the honest models in $\mathcal{S}(y)$, where $h_l^{\mathcal{O}}(x, y) \leq h_{l+1}^{\mathcal{O}}(x, y), l \in [1, K-1]$. Thus, we have $\{h_{min}, h_{max}\} = \{h_1^{\mathcal{O}}, h_K^{\mathcal{O}}\}$.

To analyze the bounds of Eq. (4) in the worst case, we consider the scenario where all the Byzantine models are included in the aggregation. So, Eq. (4) can be rewritten into

$$h_{a^*} = \frac{1}{m}(\sum^{f} h_i^{\mathcal{B}} + \sum^{m-f} h_j^{\mathcal{O}}).$$

Besides, the ground truth is $\frac{1}{K}\sum_{i=1}^{K} h_i^{\mathcal{O}} \in [h_{min}, h_{max}]$, while the second term in h_{a^*} is $\sum^{m-f} h_j^{\mathcal{O}} \in [(m-f)h_{min}, (m-f)h_{max}]$. Since we have $m > 2f$, although all malicious models are included in the aggregation, h_a is still located in $[h_{min}, h_{max}]$ according to Eq. (6). The proof of this is given in Lemma 1 in Appendix A.

The objective of attackers is to steer the aggregated value h_{a^*} as small or large as possible. Therefore, we discuss the attack in two worst cases.

Case I: All the malicious models aim to steer h_{a^*} smaller, where $h_f^{\mathcal{B}} < h_{min} \leq h_a$. From Eq. (6), we get

$$\begin{aligned} |h_1^{\mathcal{B}} - h_a| + \cdots + |h_f^{\mathcal{B}} - h_a| + |h_1^{\mathcal{O}} - h_a| + \ldots \\ + |h_{m-f}^{\mathcal{O}} - h_a| < \\ |h_{m-f+1}^{\mathcal{O}} - h_a| + \cdots + |h_m^{\mathcal{O}} - h_a| + |h_1^{\mathcal{O}} - h_a| + \ldots \\ + |h_{m-f}^{\mathcal{O}} - h_a|. \end{aligned}$$

Due to $h_f^{\mathcal{B}} < h_a < h_{m-f+1}^{\mathcal{O}}$, the above inequality can be rewritten into

$$\begin{aligned} & fh_a - \sum_{l=1}^{f} h_l^{\mathcal{B}} < \sum_{l=m-f+1}^{m} h_l^{\mathcal{O}} - fh_a \\ \Rightarrow & \sum_{l=1}^{f} h_l^{\mathcal{B}} > 2fh_a - \sum_{l=m-f+1}^{m} h_l^{\mathcal{O}} > 2fh_{min} - fh_{max} \\ \Rightarrow & \sum_{l=1}^{f} h_l^{\mathcal{B}} + \sum_{l=1}^{m-f} h_l^{\mathcal{O}} > 2fh_{min} - fh_{max} + (m-f)h_{min} \\ & = (m+f)h_{min} - fh_{max} \\ & \Rightarrow h_{a^*} > h_{min} - \frac{f}{m}(h_{max} - h_{min}). \end{aligned} \tag{18}$$

Case II: All the malicious models aim to steer h_{a^*} greater, where $h_1^{\mathcal{B}} > h_{max} \geq h_a$. Similarly, we get

$$\begin{aligned} |h_1^{\mathcal{B}} - h_a| + \cdots + |h_f^{\mathcal{B}} - h_a| + \\ |h_{K-m+f+1}^{\mathcal{O}} - h_a| + \cdots + |h_K^{\mathcal{O}} - h_a| < \\ |h_{K-m+1}^{\mathcal{O}} - h_a| + \cdots + |h_{K-m+f}^{\mathcal{O}} - h_a| + \\ |h_{K-m+f+1}^{\mathcal{O}} - h_a| + \cdots + |h_K^{\mathcal{O}} - h_a|. \end{aligned}$$

Because of $h_1^{\mathcal{B}} > h_a > h_{K-m+f}^{\mathcal{O}}$, we have

$$\begin{aligned} & \sum_{l=1}^{f} h_l^{\mathcal{B}} - fh_a < fh_a - \sum_{l=K-m+1}^{K-m+f} h_l^{\mathcal{O}} \\ \Rightarrow & \sum_{l=1}^{f} h_l^{\mathcal{B}} < 2fh_a - \sum_{l=K-m+1}^{K-m+f} h_l^{\mathcal{O}} < 2fh_{max} - fh_{min} \\ \Rightarrow & \sum_{l=1}^{f} h_l^{\mathcal{B}} + \sum_{l=K-m+f+1}^{K} h_l^{\mathcal{O}} < 2fh_{max} - fh_{min} \\ & + (m-f)h_{max} = (m+f)h_{max} - fh_{min} \\ & \Rightarrow h_{a^*} < h_{max} + \frac{f}{m}(h_{max} - h_{min}). \end{aligned} \tag{19}$$

Thus, we arrive at Eq. (16) and Eq. (17) directly follows. □

Theorem 3. *Under the same assumptions as in Theorem 2, the secure MPMC-margin $\rho(x, y, y^-)$ in Eq. (3) is bounded by*

$$\begin{aligned} & h_{min}(x,y) - h_{max}(x,y^-) - \frac{f^{\mathcal{S}(y)}}{m}(h_{max}(x,y) - h_{min}(x,y)) \\ & - \frac{f^{\mathcal{S}(y^-)}}{m}(h_{max}(x,y^-) - h_{min}(x,y^-)) < \rho(x,y,y^-) < \\ & h_{max}(x,y) - h_{min}(x,y^-) + \frac{f^{\mathcal{S}(y)}}{m}(h_{max}(x,y) - h_{min}(x,y)) \\ & + \frac{f^{\mathcal{S}(y^-)}}{m}(h_{max}(x,y^-) - h_{min}(x,y^-)). \end{aligned} \tag{20}$$

Proof. According to Eq. (16), Eq. (17), and Eq. (3), we arrive at Theorem 3. □

IV. Performance Evaluation

In this section, we evaluate our proposed BEMA system with real-world data. In particular, two baseline models are presented for comparison, which is regular centralized learning (single party) and distributed Heterogeneous Model Reused (HMR) method [4], respectively. With different experiment setup, we show that our decentralized system reaches competitive performance compared with the baseline models. Furthermore, we simulate attacks on calibration process and show the performance resilience. Finally, we give more analysis of simulation results and the attack resilience of the proposed system.

A. Data Processing

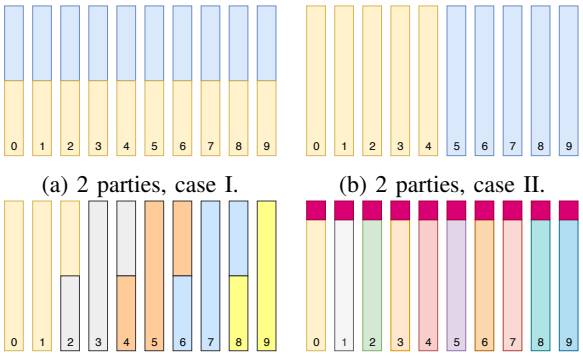
We conduct the simulation on MNIST, which is a popular machine learning dataset [13]. MNIST is an image dataset of handwritten digits, which has 60,000 training and 10,000 testing samples. Each image has a fixed-size of 28×28 pixels. To simulate the multiparty setting, we separate the dataset into different parties with different data distribution, according to Fig. 5. We extend a similar setting in [4], where four cases are formulated. These cases vary from 2 to 10 parties. Some of the cases are data unified distributed; some are skewed distributed, in which some parties may never see the data from some specific classes.

B. Baseline Models

1) Centralized Model (Single Party): We use a regular centralized model for the benchmark of simulation. In particular, the model is equipped with a classical CNN architecture, LeNet-5. The architecture of it is two sets of convolutional layers and average pooling layers, followed by a flattening convolutional layer, then two fully-connected layers, and finally, a softmax classifier [13]. The model is trained with the whole training dataset of MNIST.

2) HMR (Distributed Multiparty Learning): We also adopt a popular distributed multiparty learning method, HMR, as a baseline model [4]. In HMR, a center trusted server collects the local data and models from all parties. The server checks models over the union of all the data to find the calibration samples by using MPMC-margin, in Definition 2, and generates final system output by using the max-model predictor, which is introduced in Definition 1.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2020.2987843, IEEE Internet of Things Journal



(a) 2 parties, case I. (b) 2 parties, case II.

(c) 5 parties. (d) 10 parties.

Figure 5: Data distribution in different simulation cases. Each color represents the local data on a party. There is no data overlap between different parties. In Fig. 5d, the red color denotes the shared public data that every party can access.

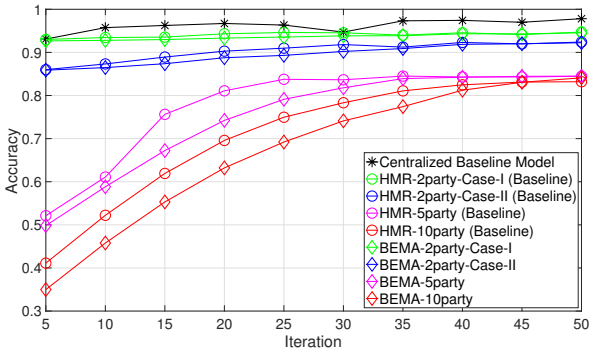


Figure 6: System prediction accuracy. The comparison results between BEMA and the baseline models.

C. System Performance and Discussion

1) Simulation Setup: In our decentralized system, each party holds a LeNet-5, which has the same structure as the model in the centralized baseline model. Besides, we apply the heterogeneous model setting in the case of Fig. 5d, where both convolutional networks (CNN) and a fully connected neural network (ANN) are adopted. Specifically, party 1, 2, and 3 hold model of LaNet-5; party 4, 5, and 6 hold model of LaNet-4 [13]; each of the rest parties holds a regular ANN, which consists of a two-hidden-layer fully connected neural nets. The setup of the calibration process is as follows. Given a calibration sample (x, y, y^-), the miner updates the corresponding local models based on Eq. (11) and Eq. (12) for $enhance$ and $impair$, respectively. To make a comparison with the centralized baseline model, we define one iteration as a calibration period that 200 calibration samples are used for model calibration. This can also be called as a calibration budget of 200, which follows the same setting with the work in [4]. We apply the proposed secure MPMC-margin to find valid calibration samples and the Krum based model predictor for the final system output. Besides, δ in Eq. (9) and σ in Eq. (10) are set as 0.6 and 0.2, respectively.

2) Prediction Accuracy: We show the system prediction accuracy with the cases listed in Fig. 5. The comparison with the baseline models is given in each case. The results are shown in Fig. 6. In all the simulated cases, it can be found that after sufficient iterations, the performance of our decentralized multiparty learning method entangles with the baseline distributed multiparty learning, HMR model. Although the centralized baseline model remains the highest performance, our decentralized multiparty learning method still reaches competitive accuracy compared with the existing distributed multiparty learning. In the cases of 5-party and 10-party, the convergence speed of HMR is higher than the proposed method. However, as the proposed method focus on the decentralized system, and we applied the secure MPMC-margin to ensure the security of model calibration, it is reasonable to see the convergence speed of the proposed method is limited. In spite of this, we are still confident to conclude that the proposed decentralized multiparty learning is a decent alternative model to the existing distributed one.

3) Attack Resilience: In this part, we further test the system to attack resilience with the designed attacks. As we mentioned in Section II-B, there are two types of critical attacks in the system. Type I attack is the broadcast of malicious local models, and type II is the broadcast of bogus calibration messages. In type I attack, the malicious party broadcasts the arbitrary model information, e.g., the attacker professes the model is trained with a class of y, while which is actually trained on the class of y'. There are two mechanisms designed in the proposed system to defend such an attack. The first is to set up restrictions when a new participant wants to add his local model in the system. From Eq. (2), the newly added models have to be performing fairly close to the existing OP's models and obtain the approval from the OP before they can join the system legally. Thus, the restriction helps prevent malicious local models to be joined into the system. Second, all the model calibration processes are public and can be verified by every participant. Thus, as long as the calibration messages are valid, the models on the blockchain cannot be tampered arbitrarily and will be updated in the right direction. To validate the calibration messages, the secure MPMC-margin and Krum based Model Predictor are proposed. In the simulation, we launch type II attack, where bogus calibration messages are generated and test the attack resilience of the proposed method.

In particular, three cases are simulated. In the first case, we continue to use the setup in Fig. 5d, where there are a total of 10 parties in the system. A certain percentage of bogus calibration messages are broadcasted in the system, where the bogus messages are generated by Eq. (1). In other words, for any calibration sample (x, y, y^-) that makes $\rho(x, y, y^-) > 0$, the attacker can maliciously broadcast calibration sample as (x, y^-, y), which makes $\rho(x, y^-, y) < 0$. Thus the bogus message is generated. If any model is calibrated by such a sample, it is maliciously tampered. In the second case, we further divide each party in the first case into two parties. The local dataset of each party is also averagely separated into two parties. Similar to the third case, we divide each party in the first case into five. The results of system prediction accuracy under the above three cases are shown in Table I.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2020.2987843, IEEE Internet of Things Journal

Table I: System prediction accuracy under attacks. In HMR, MPMC-margin and Max-Model Predictor are used for model calibration and generating system output. In BEMA, the proposed secure MPMC-margin and Krum based Model Predictor are used.

Percentage of Bogus Messages	**0%**	**10%**	**20%**	**50%**
HMR (10-Party)	0.8320	0.7926	0.7397	0.5849
BEMA (10-Party)	**0.8401**	**0.8372**	**0.8134**	**0.7692**
HMR (20-Party)	**0.8178**	0.7631	0.6959	0.5281
BEMA (20-Party)	0.8023	**0.7895**	**0.7710**	**0.7395**
HMR (50-Party)	**0.7587**	0.6945	0.6120	0.4162
BEMA (50-Party)	0.7316	**0.7228**	**0.7012**	**0.6804**

For each case, we make a comparison with HMR method, where the simulation setting is the same. All the results are the system prediction accuracy after 50 iterations of model calibration. From the table, it can be found the proposed method outperforms HMR under type II attack. Under 10% and 20% of bogus messages, BEMA still outputs competitive performance. Under 50% of bogus messages, the performance of HMR decays obviously, while the influence on the proposed method is limited. The reason for the results mainly comes from two aspects. First, regarding the calibration samples validation, we have constraints in Eq. (9) and (10) to help verify the validity of broadcasted calibration message. Thus some of the bogus calibration messages are completely ignored by the miner in the system, while HMR applies all of them. Besides, as we analyzed in Theorem 3, the influence of the attacks in each calibration iteration is bounded by applying the secure MPMC-margin. Second, regarding the system predictor, HMR uses the max-model predictor, which only selects a single model to output the prediction score on a certain data sample. It is obvious that the prediction score is easily corrupted once a malicious model is selected. By applying the proposed Krum based model predictor, our BEMA aggregates the output of multiple local models to generate the prediction on a certain data sample. Thus, the attack on the system prediction accuracy is further mitigated.

4) Discussion: In the simulation above, we first show that BEMA can reach competitive performance compared with HMR, an existing distributed multiparty learning method. Then we simulate the attacks to show the attack resilience of the proposed method. Besides, we also find the setup of system parameters, i.e., δ, and σ, influences the efficiency of the system prediction accuracy and attack resilience. δ and σ in Eq. (9) and (10) are used for verifying the calibration sample. A lower δ or a higher σ reflects that the calibration sample becomes more easily to pass the verification. As more samples pass the verification in each iteration, the convergence speed of system prediction accuracy gets higher. However, if the attacks are launched in the samples, the system is more likely to be misled as the samples are easy to be trusted. Conversely, a higher δ or a lower σ reflects a lower convergence speed, yet a higher attack resilience level. All in all, we conclude that BEMA is a decentralized multiparty learning system that is reliable on both the system performance and the attack resilience.

V. Related Work

Multiparty learning as an emerging topic, many of the related frameworks and applications are proposed. In this section, we explore the extent of these frameworks and technologies.

Yang et al. [14] provide a comprehensive survey of existing works on a secure federated learning framework. Bonawitz et al. [15] build a scalable production system for Federated Learning in the domain of mobile devices. Konečnỳ et al. [16] propose ways to reduce communication costs in federated learning. Nishio and Yonetani [17] propose a new Federated Learning protocol, FedCS, which can actively manage computing workers based on their resource conditions. Zhao et al. [18] notice that conventional federated learning fails on learning non-IID data and propose a strategy to improve training on non-IID data by creating a small subset of data which is globally shared between all the edge devices. Smith et al. [19] propose federated multi-task learning, which is a novel systems-aware optimization method, MOCHA. Besides frameworks, there are many multiparty learning-based applications proposed. De Cock et al. [20] propose a privacy-preserving solution for text classification based on secure multiparty computation, where neither the application nor the text owners learn anything about each other's contents. Shi et al. [21] present a deep sequential model for parsing discourse dependency structures of multi-party dialogues. Ma et al. [22] and Liu et al. [23] consider the traffic trajectories prediction and transportation recommendation based on multiparty learning.

In conventional multiparty learning, a master or a global model is trained based on provided data sources from each party. However, in fact, as learning is becoming more common, each party may be able to train a model (local model) based on their own local data. With conventional multi-party learning, the local models might be ignored, which is a tremendous waste of resources. Targeting this problem, some works are proposed. McMahan et al. [2] advocate to learn a shared model by aggregating local model updates in federated learning, which is proved to be robust to the unbalanced and non-IID data distributions. Shen et al. [24], inspired by Rusu [25], proposed ensemble method (MEAL) distill a variety of trained deep neural networks (DNNs) and transfer to a single DNN. Li et al. [3] propose a byzantine robust stochastic aggregation, which changes the gradient aggregation problem into model aggregation and propose a regularization term in the objective function. Ma et al. [26] propose a protocol for secure multiparty learning (SML) from the aggregation of locally trained models, by using homomorphic proxy re-encryption and aggregate signature techniques. Hamm et al. [27] build an accurate and differentially private global classifier by combining locally-trained classifiers from different parties, without access to any party's private data. Although the above works utilize the local models in multiparty learning, most of them aim to aggregate the local models to a global model and require the local models to be homogeneous. The study on multiparty learning over heterogeneous local models

is still limited, where such the case might be more practical in the real application.

Besides, a more general and practical scenario of multiparty learning should be concerned, which is to eliminate the supervision from a central trusted operator. That is to extend multiparty learning to a decentralized network structure. By eliminating the central operator in multiparty learning, the system is resilient from the single point failure, which increases the robustness of multiparty learning. Lalitha et al. [5] consider the problem of training a machine learning model over a network of users in a fully decentralized framework. They propose a decentralized learning algorithm in which users update their belief by aggregate information from their one-hop neighbors to learn a model that best fits the observations over the entire network. Omidshafiei et al. [28] introduce multi-agent reinforcement learning into a decentralized network, where each agent's model variables are distilled into a generalized network.

In decentralized multiparty learning, security and privacy are still the main concern besides the system performance, which is still remaining exploration. Byzantine attacker, who personate honest user and send arbitrary values to others to bias the ultimate system performance, is one of the most common attackers and severely threaten the system reliability. To defend the Byzantine attacks, many works are proposed. Blanchard et al. [7] focus on Byzantine tolerant gradient descent and propose Krum, an aggregation rule that satisfies the resilience property. Wang et al. [29] propose Mean-Around-Krum for secure aggregation in a decentralized traffic system. Chen et al. [10] propose an l-nearest algorithm to defend Byzantine attacks in decentralized Stochastic Gradient Descent (SGD). El-Mhamdi et al. [30] propose GUANYU, a theoretically proved Byzantine tolerant algorithm on asynchronous networks. Xie et al. [31] explore different aggregation rules for distributed synchronous SGD under a general Byzantine failure model. Regarding security in multiparty learning, several works explore the vulnerabilities of multiparty learning. Mahloujifar et al. [32] show that in a multiparty learning scenario, there always exists an attack (universal poisoning attack) that can increase the probabilities of failure in the master model. Hayes and Ohrimenko [33] show the contamination attacks can taint the model in multiparty learning and present an adversarial training method to against such attacks. Baruch et al. [34] present small but well-crafted changes, called non-omniscient attacks, which are sufficient to attack the central model in distributed learning. Bagdasaryan et al. [35] present a model-poisoning attack that significantly biases the shared model in federated learning. To strengthen the robustness of federated learning, Fung et al. [36], Bonawitz et al. [37], and Liu et al. [38] propose Sybil attack resilient federated learning, secure aggregation protocol in federated learning, and secure federated transfer learning, respectively. Besides, Kairouz et al. [39] study the problem of differential privacy in interactive function computation by multiple parties, where each party wants to compute a function. Although security and privacy problem have been widely studied in multiparty learning, few of them consider the security problem in decentralized multiparty learning with heterogeneous local models. Our work fits in the paradigm of decentralized multiparty learning with heterogeneous models, and further is attack-resilient.

VI. Conclusion

This paper focuses on an application of edge computing. In particular, we propose a blockchain-empowered secure multiparty learning system over edge devices, which is called BEMA. Different from the existing multiparty learning methods, the superiority of BEMA is twofold. First, our system extends the regular distributed multiparty learning into a fully decentralized structure, where each party is allowed to hold a heterogeneous model. Second, we carefully concern security issues in the decentralized system, where two types of Byzantine attacks are formulated. We propose "off-chain" and "on-chain" mining schemes for attack defending. Furthermore, theoretical analysis is given to show the performance of the proposed system and the system resilience under Byzantine attacks. Simulation results on the well-known machine learning data show that the efficacy and reliability of BEMA.

Appendix A

Lemma 1. *Under the same conditions as in Theorem 2, in the worst case that all Byzantine reports are included in the aggreggation, we have* $h_a \in [h_{min}, h_{max}]$.

Proof. Note that $h_{min} = h_1^{\mathcal{O}}$ and $h_{max} = h_K^{\mathcal{O}}$. We first consider the worst case that the Byzantine attackers aim to maliciously make h_a very small, where $h_f^{\mathcal{B}} < h_{min}$. Thus, the report set of "$k \xrightarrow{m} j$" in Eq. (6) can be written as $\mathbf{h} = \{h_1, h_2, \ldots, h_m\} = \{h_1^{\mathcal{B}}, ..., h_f^{\mathcal{B}}, h_1^{\mathcal{O}}, ..., h_{(m-f)}^{\mathcal{O}}\}$.

Define $KR(j) = \sum_{k \xrightarrow{m} j} |h_k - h_j|$ and $\Delta_{ij} = |h_i - h_j|$. Then, we have

$$\begin{aligned} KR(s) =& \Delta_{1s} + \cdots + \Delta_{(s-1)s} + \\ & \Delta_{(s+1)s} + \cdots + \Delta_{ms}, \\ KR(s+1) =& \Delta_{1(s+1)} + \cdots + \Delta_{s(s+1)} + \\ & \Delta_{(s+2)(s+1)} + \cdots + \Delta_{m(s+1)}. \end{aligned}$$

Note that $\Delta_{i(s+1)} = \Delta_{is} + \Delta_{s(s+1)}$ for $1 \le i < s$ and $\Delta_{i(s+1)} = \Delta_{is} - \Delta_{s(s+1)}$ for $s+1 < i \le m$. Thus, we can obtain

$$\begin{aligned} KR(s+1) =& \Delta_{1s} + \Delta_{2s} + \cdots + \Delta_{(s-1)s} + \\ & (s-1)\Delta_{s(s+1)} + \Delta_{s(s+1)} + \Delta_{(s+2)s} \\ & + \cdots + \Delta_{ms} - (m-s-1)\Delta_{s(s+1)} \\ =& KR(s) + (2s-m)\Delta_{s(s+1)}. \end{aligned}$$

Due to $m \ge 2f$, we have $KR(s+1) < KR(s)$ for $1 \le s \le f$, which means $KR(f+1) < KR(f)$. Consequently, we get $h_a \ge h_{min}$.

Similarly, in the worst case that Byzantine attackers aim to maliciously make h_a very large, where $h_1^{\mathcal{B}} > h_{max}$, we can prove that $h_a \le h_{max}$.

Therefore, Lemma 1 follows. □

REFERENCES

[1] X. Wang, Y. Han, C. Wang, Q. Zhao, X. Chen, and M. Chen, "In-edge ai: Intelligentizing mobile edge computing, caching and communication by federated learning," *IEEE Network*, vol. 33, no. 5, pp. 156–165, 2019.

[2] H. B. McMahan, E. Moore, D. Ramage, and B. A. y Arcas, "Federated learning of deep networks using model averaging," 2016.

[3] L. Li, W. Xu, T. Chen, G. B. Giannakis, and Q. Ling, "Rsa: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 1544–1551.

[4] X.-Z. Wu, S. Liu, and Z.-H. Zhou, "Heterogeneous model reuse via optimizing multiparty multiclass margin," in *International Conference on Machine Learning*, 2019, pp. 6840–6849.

[5] A. Lalitha, S. Shekhar, T. Javidi, and F. Koushanfar, "Fully decentralized federated learning," in *Proceedings of third workshop on Bayesian Deep Learning (NeurIPS)*, 2018.

[6] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun, "On the security and performance of proof of work blockchains," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. ACM, 2016, pp. 3–16.

[7] 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.

[8] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[9] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*. Springer, 2010, pp. 177–186.

[10] X. Chen, J. Ji, C. Luo, W. Liao, and P. Li, "When machine learning meets blockchain: A decentralized, privacy-preserving and secure design," in *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2018, pp. 1178–1187.

[11] G. Becker, "Merkle signature schemes, merkle trees and their cryptanalysis," *Ruhr-University Bochum, Tech. Rep*, 2008.

[12] H. Gilbert and H. Handschuh, "Security analysis of sha-256 and sisters," in *International workshop on selected areas in cryptography*. Springer, 2003, pp. 175–193.

[13] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[14] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, p. 12, 2019.

[15] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konecny, S. Mazzocchi, H. B. McMahan *et al.*, "Towards federated learning at scale: System design," *arXiv preprint arXiv:1902.01046*, 2019.

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

[17] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE, 2019, pp. 1–7.

[18] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.

[19] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 4424–4434.

[20] M. De Cock, R. Dowsley, A. C. Nascimento, D. Reich, and A. Todoki, "Privacy-preserving classification of personal text messages with secure multi-party computation: An application to hate-speech detection," *arXiv preprint arXiv:1906.02325*, 2019.

[21] Z. Shi and M. Huang, "A deep sequential model for discourse parsing on multi-party dialogues," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 7007–7014.

[22] Y. Ma, X. Zhu, S. Zhang, R. Yang, W. Wang, and D. Manocha, "Trafficpredict: Trajectory prediction for heterogeneous traffic-agents," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 6120–6127.

[23] H. Liu, T. Li, R. Hu, Y. Fu, J. Gu, and H. Xiong, "Joint representation learning for multi-modal transportation recommendation," *AAAI, to appear*, 2019.

[24] Z. Shen, Z. He, and X. Xue, "Meal: Multi-model ensemble via adversarial learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 4886–4893.

[25] A. A. Rusu, S. G. Colmenarejo, C. Gulcehre, G. Desjardins, J. Kirkpatrick, R. Pascanu, V. Mnih, K. Kavukcuoglu, and R. Hadsell, "Policy distillation," *arXiv preprint arXiv:1511.06295*, 2015.

[26] X. Ma, C. Ji, X. Zhang, J. Wang, J. Li, and K.-C. Li, "Secure multiparty learning from aggregation of locally trained models," in *International Conference on Machine Learning for Cyber Security*. Springer, 2019, pp. 173–182.

[27] J. Hamm, Y. Cao, and M. Belkin, "Learning privately from multiparty data," in *International Conference on Machine Learning*, 2016, pp. 555–563.

[28] S. Omidshafiei, J. Pazis, C. Amato, J. P. How, and J. Vian, "Deep decentralized multi-task multi-agent reinforcement learning under partial observability," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 2681–2690.

[29] Q. Wang, T. Ji, Y. Guo, L. Yu, X. Chen, and P. Li, "Trafficchain: A blockchain based secure and privacy-preserving traffic map," *IEEE Access*, 2020.

[30] E.-M. El-Mhamdi, R. Guerraoui, A. Guirguis, and S. Rouault, "Sgd: Decentralized byzantine resilience," *arXiv preprint arXiv:1905.03853*, 2019.

[31] C. Xie, O. Koyejo, and I. Gupta, "Generalized byzantine-tolerant sgd," *arXiv preprint arXiv:1802.10116*, 2018.

[32] S. Mahloujifar, M. Mahmoody, and A. Mohammed, "Universal multi-party poisoning attacks."

[33] J. Hayes and O. Ohrimenko, "Contamination attacks and mitigation in multi-party machine learning," in *Advances in Neural Information Processing Systems*, 2018, pp. 6604–6615.

[34] M. Baruch, G. Baruch, and Y. Goldberg, "A little is enough: Circumventing defenses for distributed learning," *arXiv preprint arXiv:1902.06156*, 2019.

[35] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," *arXiv preprint arXiv:1807.00459*, 2018.

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

[37] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for federated learning on user-held data," *arXiv preprint arXiv:1611.04482*, 2016.

[38] Y. Liu, T. Chen, and Q. Yang, "Secure federated transfer learning," *arXiv preprint arXiv:1812.03337*, 2018.

[39] P. Kairouz, S. Oh, and P. Viswanath, "Secure multi-party differential privacy," in *Advances in neural information processing systems*, 2015, pp. 2008–2016.