Do Not Perturb Me: A Secure Byzantine-Robust Mechanism for Machine Learning in IoT

Xindi MA*, Junying ZHANG†, Jianfeng MA*, Qi JIANG*, Sheng Gao‡ and Kang XIE§¶
*School of Cyber Engineering, Xidian University, Shaanxi, China
†College of Engineering, Peking University, Beijing, China
‡School of Information, Central University of Finance and Economics, Beijing, China
§Key Lab of Information Network Security, Ministry of Public Security, Shanghai, China

2020 International Conference on Networking and Network Applications (NaNA) | 978-1-7281-8954-3/20/$31.00 ©2020 IEEE | DOI: 10.1109/NANA51271.2020.00066

***Abstract*—With the development of Internet of Things (IoTs) and big data, collaborative machine learning has achieved many impressive successes in IoT to improve the system performance and provide more diversified services for people. Although one of the motivations for collaborative learning is privacy preservation, the adversary can still launch an inference attack through task nodes' shared information. What's worse, a few task nodes may perform as a Byzantine attacker to compromise the entire system. Many Byzantine-robust mechanisms have been proposed, but they relied on outsourcing the calculation on two non-colluding servers which were not realistic in practice or had privacy issues in one-server architecture. In this paper, we design a novel mechanism for secure Byzantine-robust collaborative machine learning, namely Omega, to allow IoT devices to achieve the collaborative model training without exposing their local data to the others. Specifically, we construct a single-server architecture to achieve the private aggregation of parameter gradients, which protects task nodes' local data even $n-1$ of n nodes colluded. A new secure Byzantine-robust protocol is also designed to resist the Byzantine attack and this protocol can be extended to any distance-based robust rule. Furthermore, we prove that Omega can ensure task nodes' privacy preservation. Finally, we conduct an experiment to evaluate Omega over real-world dataset and empirical results demonstrate that Omega can efficiently achieve the collaborative machine learning.**

I. Introduction

With the development of Internet of Thing (IoT) and big data, machine learning has achieved many impressive successes in IoT applications to improve the system performance and provide more diversified services for people. Nowadays, the architectures of machine learning tend to aggregate several data owners to achieve the collaborative learning, including the distributed learning and federated learning. Together with the greatly enhanced computation capability of IoT devices, collaborative machine learning (termed as federated learning as well) has emerged as a vastly-developed learning scheme in the real world IoT application. In collaborative learning, each task node owns its local training data and trains its local model. Then, all task nodes periodically exchange their trained gradients and update the global model with or without a central server.

Another important motivation for collaborative learning is the privacy preservation. Because the training data usually contains abundant sensitive and private information (e.g., environmental data, location, temperature, and so on), data owners are reluctant to contribute their data due to the privacy and confidentiality concerns. Hence, many collaborative learning architectures [1] [2] are specifically designed to achieve the privacy-preserving machine learning. However, although the training data is kept locally in collaborative machine learning, the adversary can still infer task nodes' private information from their shared knowledge [1] [3] [4], such as the gradients, empirical loss, model parameters, and so on.

While using the machine learning in IoT, there are many IoT devices that are uncontrollable. These devices may induce a higher risk of model training failures. These include crashes and computation errors, stalled processes, biases in the way the data samples are distributed among the processes, but also, in the worst case, attackers trying to compromise the entire system. The typical attack is Byzantine attack, i.e., completely arbitrary behaviors of some of the processes. A Byzantine participant or worker can behave arbitrarily malicious , e.g., sending arbitrary updates to the server. This poses great challenge to the most widely used aggregation rules, e.g., simple average, since a single Byzantine worker can compromise the results of aggregation. To resist this Byzantine attack, many robust mechanisms have been proposed. However, most of these system mechanisms are not easy or friendly for practical usage. The authors in [5] proposed a Byzantine tolerant gradient descent mechanism without considering the privacy preservation of data owners. Although He *et al.* [6] designed a secure Byzantine-robust machine learning scheme, they relied on outsourcing the calculation to two non-colluding servers which are not realistic in practice. Hence, how to achieve the practical and secure Byzantine-robust machine learning is still an unsolved challenge in collaborative machine learning.

To tackle the above challenge, we propose a novel mechanism, namely Omega, to resist the Byzantine attack while achieving the privacy preservation. In Omega, task nodes' private local training data will not be revealed and the adversary cannot infer any private information from the intermediate results. Based on a distance-based robust aggregation rule, we achieve the Byzantine-robust machine learning without revealing the private information for task nodes and central servers. The main contributions of this paper are as follows:

¶Kang XIE is corresponding author. Email: xiekang@stars.org.cn

978-1-7281-8954-3/20/$31.00 ©2020 IEEE
DOI 10.1109/NaNA51271.2020.00066

- *Privacy-Preserving Gradient Aggregation:* With a central server managing model parameters, Omega achieves the collaborative machine learning with single-server architecture. Based on the homomorphic encryption, a new protocol is designed to aggregate the model parameters. The adversary cannot infer any private information about task nodes' local training data and intermediate results.
- *Defending the Byzantine Attack:* In the single-server architecture, we design a secure protocol to achieve the Byzantine-robust machine learning. The private distance between task nodes' gradients are still kept and the adversary, including the curious server and task nodes, cannot infer any private information from the calculation of Byzantine labels.
- *Privacy and Efficiency:* We conduct the analysis of Omega in both theory and practice. Both theoretical and experimental results show that Omega can achieve the Byzantine-robust machine learning effectively and efficiently.

II. Related Work

In the past few years, privacy issues have been gained significant interests and a lot of mechanisms were proposed to achieve the privacy preservation, such as social privacy [7]–[9], location privacy [10], [11], identity privacy [12]–[14], and so on. With the development of machine learning, the privacy issues in machine learning also received a lot of attention and many valuable works have been presented.

Based on the secure multi-party computation, many secure mechanisms have been designed to achieve the privacy-preserving model training in centralized learning. Wagh *et al.* [15] proposed an end-to-end 3-party protocol for fast and secure computation of deep learning algorithms on large networks. Mohassel *et al.* [16] designed several new and efficient protocols for privacy preserving machine learning for linear regression, logistic regression, and neural network. To verify the integrity or correctness of the aggregated results, Xu *et al.* [17] proposed the first privacy-preserving and verifiable federated learning framework based on the NO-hard problem. However, these works did not consider the collected data which typically comes from several data owners and is encrypted with different keys. To achieve the model training on multi-key encrypted data, Li *et al.* [18] and Ma *et al.* [19] introduced two privacy-preserving mechanisms based on the homomorphic encryption.

Recently, most research works move future steps to the distributed learning system. Xie *et al.* [20] proposed a novel differentially private proximal gradient algorithm to solve a general class of multi-task learning formulations, which trained low accuracy models because of the sanitized noise. To collaboratively train a Gaussian process regression model, Fenner and Pyzer-Knapp [21] designed a modular approach which applied fully homomorphic encryption to only the sensitive steps without either party gaining access to the other's data. However, the above works did not consider the Byzantine attack. Although He *et al.* [6] proposed a secure Byzantine-robust machine learning mechanism, it revealed the gradient distances to a computing server, which had a serious privacy leakage problem.

III. Preliminaries

In this section, we outline some preliminaries. Hereinafter, if all ciphertexts belong to several specific task nodes, we will use $[\![x]\!]$ instead of $[\![x]\!]_{pk_\Sigma}$.

A. Distributed Two Trapdoors Public-Key Cryptosystem

In Omega, we introduce a distributed two-trapdoor public-key cryptosystem (DT-PKC) [22] to achieve the secure model training. In detail, DT-PKC consists of the following subalgorithms (see [22] for detailed construction).

KetGen: Given two large primes, constructed parameter N and generate public key pk_i, corresponding weak private key sk_i, and two partial strong private key SK_i and SK_2.

Encryption ($\texttt{Enc}$): Input plaintext $x \in \mathbb{Z}_N$ and public key pk_i, output ciphertext $[\![x]\!]_{pk_i}$.

Partial Strong decryption step-I ($\texttt{PSD1}$): Input ciphertext $[\![x]\!]_{pk_i}$ and partial strong private key SK_1, output partial decrypted ciphertext sd_{i1}.

Partial Strong decryption step-II ($\texttt{PSD2}$): Input $[\![x]\!]_{pk_i}$, sd_{i1}, and key SK_2, output x.

Note that for ciphertexts $[\![x_1]\!]_{pk_i}$ and $[\![x_2]\!]_{pk_i}$ under the same key, the following properties are existed: 1) additive homomorphism: $[\![x_1]\!]_{pk_i} \cdot [\![x_2]\!]_{pk_i} = [\![x_1 + x_2]\!]_{pk_i}$; 2) scalar-multiplicative homomorphism: $([\![x_1]\!]_{pk_i})^{N-a} = [\![-a \times x_1]\!]_{pk_i}$, where $a \in \mathbb{Z}_N$ is a constant.

B. Secure Integer Computation Protocols

Given DT-PKC encrypted ciphertexts $[\![x]\!]$ and $[\![y]\!]$, we can conduct the following protocols:

Secure Multiplication Protocol ($\texttt{SMP}$): $\texttt{SMP}$ securely calculates the homomorphic multiplication and outputs $[\![x \times y]\!]$, denoted as $[\![x \times y]\!] \leftarrow \texttt{SMP}([\![x]\!], [\![y]\!])$. Specifically, if $x = y$, $[\![x^2]\!] \leftarrow \texttt{SMP}([\![x]\!])$.

Secure Maximum Selection Protocol ($\texttt{SMax}_n$): $SMax_n$ securely calculates the maximum in n encrypted samples, denoted as $[\![x_{max}]\!] \leftarrow \texttt{SMax}_n([\![x_1]\!], \cdots, [\![x_n]\!])$.

We refer the interested readers to [19] for a detailed description of the $\texttt{SMP}$ protocol and [22] for $\texttt{SMax}_n$ protocol.

C. Differential Privacy

In Omega, we introduce difference privacy to protect the distance of gradients which is used to achieve the robust defending for Byzantine attack. $\mathcal{M}_p$ is conducted as the random sanitized algorithm. Thus, it is difficult for the task node to recover other nodes' gradients.

Definition 1 (differential privacy): Algorithm $\mathcal{M}_p$ is ϵ-differential privacy if for any subset of outputs S:

$$\mathcal{P}(\mathcal{M}_p(\mathcal{D}) \in S) \leq e^{\epsilon} \times \mathcal{P}(\mathcal{M}_p(\mathcal{D}') \in S),$$

for any adjacent datasets $\mathcal{D}$ and $\mathcal{D}'$, where $\mathcal{M}_p(\mathcal{D})$ and $\mathcal{M}_p(\mathcal{D})$ are the outputs of the algorithm for inputs $\mathcal{D}$ and $\mathcal{D}'$ respectively, $\mathcal{P}$ is the randomness of the noise in the algorithm.

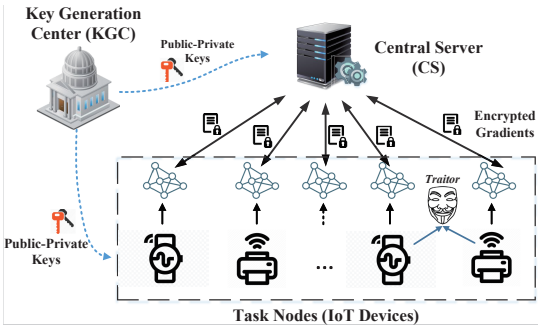


Fig. 1: System model of Omega

To measure the maximum change in $\mathcal{M}_p$ resulted by a single data point, we introduce the L_2-sensitivity:

Definition 2 (L_2**-sensitivity):** For algorithm $\mathcal{M}_p$, the L_2-sensitivity is defended as follows:

$$S(\mathcal{M}_p) \leftarrow \max_{\mathcal{D},\mathcal{D}',\|\mathcal{D}-\mathcal{D}'\|_1=1} \left\|\mathcal{M}_p(\mathcal{D}) - \mathcal{M}_p(\mathcal{D}')\right\|_2$$

where $\mathcal{D}$ and $\mathcal{D}'$ are adjacent datasets, $S(\mathcal{M}_p)$ is the maximum difference in the L_2 norm between the outputs of $\mathcal{M}_p$.

IV. System Overview of Omega

A. System Model

As shown in Fig. 1, our proposed Omega system comprises three parts: Key Generation Center (KGC), Central Server (CS), and several Task Nodes (TN).

- **KGC** is an indispensable entity that generates and distributes all the public and private keys in the system. It is trusted by all parts.
- **CS** manages the global model parameters and supplies some computational resources to update the parameters. After training, CS releases the trained model to task nodes. CS owns the partial strong private key SK_1 and all task nodes' union public key pk_Σ.
- **TN** owns its local training data and achieves the collaborative learning with the help of CS. To protect the privacy, all task nodes have different public-private key pairs, the same union public key pk_Σ[1], and another partial strong private key SK_2.

While transmitting the information through the network, we introduce the secure socket layer (SSL) or transport layer security (TLS) to secure all communication channels. The SSL/TLS protocol aims primarily to ensure the data integrity and authenticity between two communication entities.

B. Threat Model

In Omega, we consider KGC is trusted by all entities. It generates all the public-private keys for the system. We also consider CS and most of TNs are *honest-but-curious* (non-colluding) parties. Although CS and these curious TNs strictly follow the protocols, they are also interested to gather or learn other parties' private information during the training process. In Omega, we consider only a few TNs are Byzantine attackers. These TNs may contribute malformed messages and try to compromise the entire system in the worst case. Thus, we introduce an active adversary $\mathcal{A}$ which can decrypt the challenge CS's encrypted global parameters and infer the challenge TNs' local training data with the following capabilities:

- $\mathcal{A}$ could *eavesdrop* all communications to obtain the transmitted information and launch an active attack to infer and forge the intercepted messages.
- $\mathcal{A}$ could *compromise* CS to guess the plaintext of all encrypted information and infer TNs' local training data.
- $\mathcal{A}$ could *compromise* one or more curious TNs, except for challenge TNs, to obtain access to their decryption abilities and infer other TNs' local training data through the intermediate results.
- $\mathcal{A}$ could *compromise* one or more Byzantine TNs to construct malformed information to subvert the model training.

However, the adversary $\mathcal{A}$ is not allowed to compromise CS and TNs concurrently. We remark that such restrictions are typical in cryptographic protocols [22].

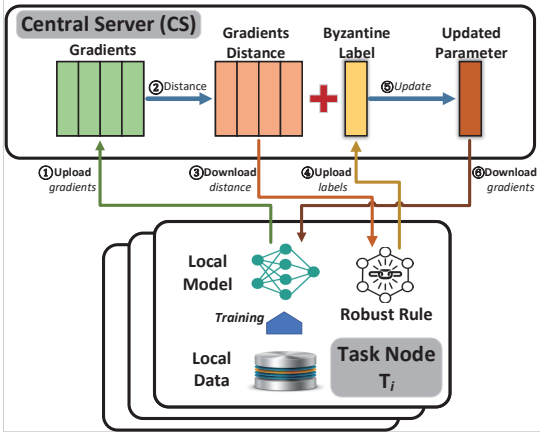


Fig. 2: High-level Architecture of Omega

V. Construction of Omega

In this section, we present the details of Omega and the architecture of Omega is shown in Fig. 2. In each training epoch, task nodes train the model on their local training data and upload the parameter gradients to CS to update the global model. Because all task nodes perform the same training process, we take task node T_i as an example to train the model in the following subsections. After receiving the uploaded gradients, CS aggregates all the gradients as follows and then uses the aggregated result to update the global model:

$$\mathcal{G} \leftarrow \sum_{i=1}^{n} g_i,$$

where n is the number of task nodes, g_i is the uploaded gradients from T_i.

[1]The union public key is constructed will all participants' public keys $pk_i (i = 1, \cdots, n)$

Thus, to achieve the privacy preservation, T_i needs to encrypt its generated gradients before uploading to CS. However, we also consider that a few Byzantine nodes are existed in the system. So we should design a secure robust aggregation protocol to ensure the model training normally. In Omega, we adopt a distance-based robust aggregation rule, such as Multi-Krum [5], which relies on computing $\|g_i - g_j\|^2$ for all i, j node pairs to defend the Byzantine attack.

Step-I (@T_i): T_i trains its local model on its local training data and obtains the parameter gradients g_i. Before uploading g_i to CS, T_i encrypts g_i with the union public key pk_Σ and uploads the ciphertext cg_i to CS:

$$cg_i \leftarrow \texttt{Enc}(g_i, pk_\Sigma) = [\![g_i]\!].$$

Step-II (@CS): At each epoch, CS randomly selects several task nodes to upload the parameter gradients and updates the global model based on the robust aggregation protocol. After receiving the parameter gradients, CS firstly calculates the square of the pairwise Euclidean distances:

$$d_{ij} \leftarrow [\![g_i]\!] \cdot [\![g_j]\!]^{N-1} = [\![g_i - g_j]\!],$$
$$\xi_{ij} \leftarrow \texttt{SMP}(d_{ij}) = [\![(g_i - g_j)^2]\!],$$

where $[\![g_i]\!]$ and $[\![g_j]\!]$ are the received parameter gradients from task node T_i and T_j.

Step-III (@CS): When filtering out the Byzantine nodes, we need to obtain the plaintext of the pairwise distance. Thus, to resist the inference attack launched by the task nodes, we introduce the difference privacy to sanitize the ciphertexts of Euclidean distance. Because we directly perturb the distances, we choose the maximum value of the pairwise distances as the sensitivity. Hence, we use the $\texttt{SMax}_n$ protocol to calculate the maximum ciphertext in calculated distance:

$$d_{max} \leftarrow \texttt{SMax}_n(\xi_{11}, \xi_{12}, \cdots, \xi_{ij}, \cdots, \xi_{nn})$$

where n is the number of chose task nodes and $i, j \in [1, n]$.

Step-IV (@CS): After calculating the sensitivity of differential privacy, we generate the sanitized noise based on the Laplace mechanism. Then, based on the homomorphic additive property, we perturb the pairwise Euclidean distances as follows:

$$\xi'_{ij} \leftarrow \xi_{ij} \cdot \texttt{Enc}(\texttt{Lap}(\frac{d_{max}}{\epsilon}), pk_\Sigma) = [\![(g_i - g_j)^2 + \texttt{Lap}(\frac{d_{max}}{\epsilon})]\!],$$

where $\texttt{Lap}(\cdot)$ is the Laplace noise, d_{max} is the sensitivity, ϵ is privacy budget. Then, CS partially decrypts the sanitized distances:

$$sd_{ij} \leftarrow \texttt{PSD1}(\xi'_{ij}, SK_1).$$

After that, CS disturbs the orders of partial decrypted distance matrix and maps it to another matrix. Then, CS randomly chooses a task node T_g which is not chosen to upload gradients and sends the disturbed matrix to T_g.

Step-V (@T_g): After receiving the disturbed matrix, T_g partially decrypts it with SK_2 and obtains the plaintexts of the square of the pairwise Euclidean distances. Then, T_g feeds these distances to the distance-based robust aggregation rule and obtains a weight vector $\mathbf{u} = \{u_i\}_{i=1}^n$, we call it Byzantine labels, which is a vector of binary values and indicates that the selected nodes' ($u_i = 1$) gradients will be used to update the global model. T_g encrypts the Byzantine labels $\mathbf{u}$ with the union public key pk_Σ and sends the ciphertext to CS.

Step-VI (@CS): Based on the mapping relationship, CS recovers the normal orders of Byzantine labels $[\![\mathbf{u}]\!]$, represented as $[\![\mathbf{u}']\!]$ and calculates the products with the received gradients. Then, CS aggregates the products based on the homomorphic additive property:

$$[\![\mathcal{G}]\!] \leftarrow \prod_{i=1}^{n} \texttt{SMP}([\![g_i]\!], [\![u'_i]\!]) = [\![\sum_{i=1}^{n} g_i \times u'_i]\!],$$

where $u'_i \in \mathbf{u}'$, $[\![g_i]\!]$ is the received parameter gradients from task node T_i.

Step-VII (@CS): After that, CS updates the model parameters based on the aggregated parameter gradients:

$$[\![w_z]\!] \leftarrow [\![w_z]\!] \cdot [\![\theta_z]\!]^{N-\eta} = [\![w_z - \eta \times \theta_z]\!],$$

where w_z is the model parameters, η is the learning rate for model updating, $\theta_z \in \mathcal{G}$. We encrypt the model parameters and send the ciphertext to CS, because the encryption can resist the inference attack launched by CS. After updating the model parameters, CS partially decrypts the parameters and sends the partially decrypted ciphertexts to T_i.

Step-VIII (@T_i): T_i continues to decrypt the model parameters with partial strong private key SK_2 and obtains the plaintexts of parameters. Then, T_i performs the model training in the next iteration.

We summarize the secure Byzantine-robust mechanism in Algorithm 1.

VI. Theoretical Analysis

In this paper, we adopt a security model that is usually used to prove the security of multi-party protocols [23] [24] to prove the privacy preservation of the model training in Omega. Considering two parties: CS and T_i, we conduct two simulators $(S_{CS}, S_{T_i}, S_{T_g})$ against two types of attackers $(\mathcal{A}_{CS}, \mathcal{A}_{T_i}, \mathcal{A}_{T_g})$ that corrupt CS, T_i, and T_g respectively.

Theorem 1: The aggregation of gradients in Omega is secure against the adversary $\mathcal{A}$ defined in the threat model.

Proof: At the beginning of gradient aggregation, T_i encrypts the gradients with DT-PKC and sends the ciphertext to CS. Thus, based on the security of DT-PKC, $\mathcal{A}_{CS}$ cannot infer any private information from T_i's gradients. To resist the Byzantine attack, CS calculates the square of the pairwise Euclidean distances based on the $\texttt{SMP}$ protocol and the system does not reveal the plaintext to CS. Before calculating the Byzantine labels, CS perturbs the distances and reveals the differentially private distances to T_g. Hence, $\mathcal{A}_{CS}$ and $\mathcal{A}_{T_g}$ also cannot distinguish the real and the ideal executions in this phase. After obtaining the Byzantine labels, T_g encrypts the labels and sends the ciphertexts to CS. Without the partial strong private key SK_2, CS cannot decrypt the encrypted labels. Then, CS aggregates the parameter gradients based on

Algorithm 1: Secure Byzantine-Robust Model training

Data: Training datasets $\mathcal{D}_i$ and sensitive property p.
Result: Participant T_i's personalized model.

1 (@T_i): encrypt the calculated gradients and send the results to CS;
2 (@CS): calculate the square of the pairwise Euclidean distances:

$$\xi_{ij} \leftarrow [\![(g_i - g_j)^2]\!];$$

3 (@CS): calculate the differential privacy d_{max} and generate the Laplace noise;
4 (@CS): perturb the distances with differentially private noise:

$$\xi'_{ij} \leftarrow [\![(g_i - g_j)^2 + \texttt{Lap}(\frac{d_{max}}{\epsilon})]\!];$$

5 (@CS): partially decrypt the sanitized distances and send the results to T_g;
6 (@T_g): partially decrypt sd_{ij}, feed these distances to the distance-based robust aggregation rule, and obtain **u**;
7 (@CS): aggregate the parameter gradients based on the Byzantine labels:

$$[\![\mathcal{G}]\!] \leftarrow [\![\sum_{i=1}^{n} g_i \times u'_i]\!];$$

8 (@CS): update model parameters based on the homomorphic additive property:

$$[\![w_z]\!] \leftarrow [\![w_z - \eta \times \theta_z]\!];$$

9 (@T_i): train the model on the updated model parameters.

the SMP protocol and updates the model parameters based on the homomorphic property. The views of $\mathcal{A}_{CS}$ in the real and the ideal executions are also indistinguishable in this process. ■

VII. Experimental Evaluation

In this section, we evaluate Omega over a medical dataset, ADNI, which was collected by the Alzheimers Disease Neuroimaging Initiative. ADNI researchers collect several types of data, including clinical, genetic, MRI image, PET image, biospecimen, from study volunteers throughout their participation in the study, using a standard set of protocols and procedures to eliminate inconsistencies. There are a total of 79 tasks in ADNI. To evaluate Omega, we just select 20 tasks' datasets as our training data. Thus, we have 20 task nodes in our system. We conduct the experiments over a machine with a 2.6 GHz 32-core processor and 128 GB RAM.

We mainly evaluate the efficiency and model training accuracy of Omega in this section. When discussing the efficiency, we focus on the running time of three stages, including encrypting the gradients (@T_i), calculating the Byzantine labels (@T_g), and updating the global model parameters (@CS), by varying the number of task nodes. It should be noted that all

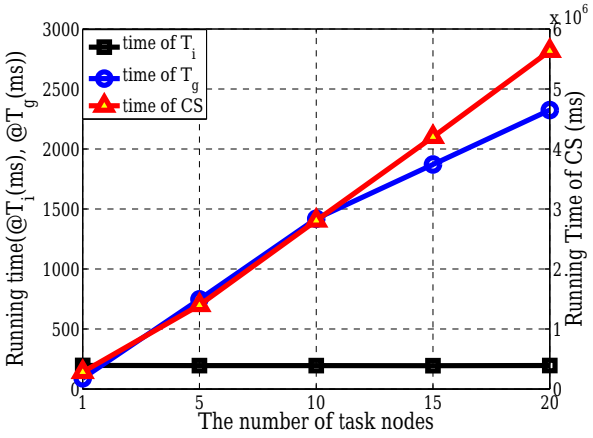


Fig. 3: Efficiency evaluation of Omega

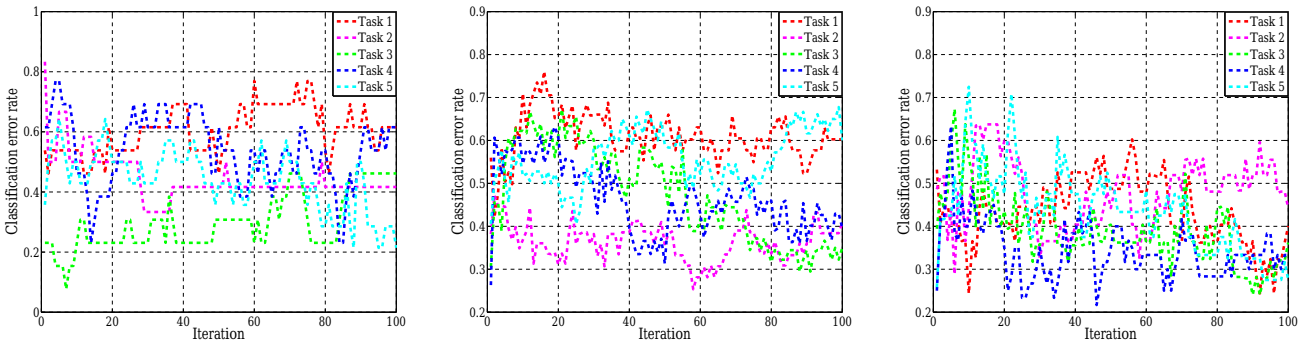
the reported time is counted over 100 iterations and averaged over 10 runs.

The simulation results are shown in Fig. 3. By varying the number of task nodes, CS takes more time to achieve the global model updating. As the number of task nodes increases, CS needs to aggregate more uploaded gradients and calculates the products between gradients and Byzantine labels. Thus, CS spends more time on updating the model parameters. We also find that the calculating time of Byzantine labels by T_g also increases. The reason for this is that T_g spends more time to filter the Byzantine attackers and encrypts more Byzantine labels to send to CS. So T_g also spends more time with the increasing of task nodes. However, the encrypting time spent by T_i is stable. Although the number of task nodes increases, the task nodes do not interfere with each other. Thus, the increasing of task nodes number does not have any influence on T_i's encrypting time for parameter gradients.

While evaluating the model training accuracy, we use the classification error rate to estimate the trained model accuracy. The simulation results are shown in Fig. 4. We respectively choose $5, 10, 15$ task nodes to train the model. In Fig. 4, we find that the classification error decreases with the number of task nodes, that is, more task nodes train a more accurate model. As the training progresses, each task node uploads the calculated gradients to central server to achieve the collaborative learning. So each task node benefits from the shared knowledge trained by other tasks and the trained model will be more accurate. Although there are one or more Byzantine participants in the task nodes, we find that Omega can still train an accurate model.

VIII. Conclusion

In this paper, we proposed a novel protocol, namely Omega, to address the grand challenges in secure Byzantine-robust collaborative learning. Specifically, we considered that all task nodes (IoT devices) train a collaborative model without sharing their local training data and a few task nodes are Byzantine attackers. These attackers might launch the active attack and tried to compromise the entire system. Thus, based



(a) Number of task node $n = 5$ (b) Number of task node $n = 10$ (c) Number of task node $n = 15$

Fig. 4: Classification error rate for different task nodes

on the homomorphic encryption, we design a robust gradient aggregation protocol and combined an existed distance-based Byzantine-robust rule to achieve the secure Byzantine-robust collaborative machine learning. Moreover, we evaluated the effectiveness and performance of Omega and the results justified that Omega was effective and efficient. As part of our future work, we will consider the reputation of task nodes and design a reputation-based mechanism to resist the Byzantine attack.

ACKNOWLEDGMENT

This work was supported by the National Natural Science Foundation of China (Grant Nos. 61902290, 61902291, 61672413, 61872283),Key Research and Development Program of Shaanxi(Grant Nos. 2020ZDLGY09-06, 2019ZDLGY12-04), Natural Science Foundation of Shaanxi Province (Grant Nos. 2019JM-109, 2019JM-425), Key Lab of Information Network Security, Ministry of Public Security (Grant No. C19604), Scientific Research Program Funded by Shaanxi Provincial Education Department (Grant No. 20JY016), Fundamental Research Funds for the Central Universities (Grant Nos. JB191508, JB191507).

REFERENCES

[1] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015, pp. 1310–1321.

[2] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS, FL, USA*, vol. 54, 2017, pp. 1273–1282.

[3] M. Abadi, A. Chu, I. J. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 308–318.

[4] L. Melis, C. Song, E. D. Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *Proceedings of IEEE Symposium on Security and Privacy, San Francisco, CA, USA*. IEEE, 2019, pp. 691–706.

[5] 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.

[6] L. He, S. P. Karimireddy, and M. Jaggi, "Secure byzantine-robust machine learning," *arXiv preprint arXiv:2006.04747*, 2020.

[7] X. Ma, J. Ma, H. Li, Q. Jiang, and S. Gao, "Armor: A trust-based privacy-preserving framework for decentralized friend recommendation in online social networks," *Future Generation Computer Systems*, vol. 79, pp. 82–94, 2018.

[8] D. Wu, S. Si, S. Wu, and R. Wang, "Dynamic trust relationships aware data privacy protection in mobile crowd-sensing," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2958–2970, 2018.

[9] K. Gu, L. Wang, and B. Yin, "Social community detection and message propagation scheme based on personal willingness in social network," *Soft Computing*, vol. 23, no. 15, pp. 6267–6285, 2019.

[10] X. Ma, H. Li, J. Ma, Q. Jiang, S. Gao, N. Xi, and D. Lu, "Applet: a privacy-preserving framework for location-aware recommender system," *Science China Information Sciences*, vol. 60, no. 9, p. 092101, 2017.

[11] S. Gao, J. Ma, W. Shi, G. Zhan, and C. Sun, "Trpf: A trajectory privacy-preserving framework for participatory sensing," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 6, pp. 874–887, 2013.

[12] Q. Jiang, N. Zhang, J. Ni, J. Ma, X. Ma, and K.-K. R. Choo, "Unified biometric privacy preserving three-factor authentication and key agreement for cloud-assisted autonomous vehicles," *IEEE Transactions on Vehicular Technology*, 2020.

[13] Q. Jiang, Z. Chen, J. Ma, X. Ma, J. Shen, and D. Wu, "Optimized fuzzy commitment based key agreement protocol for wireless body area network," *IEEE Transactions on Emerging Topics in Computing*, 2019.

[14] X. Guo, H. Lin, Z. Li, and M. Peng, "Deep reinforcement learning based qos-aware secure routing for sdn-iot," *IEEE Internet of Things Journal*, 2019.

[15] S. Wagh, S. Tople, F. Benhamouda, E. Kushilevitz, P. Mittal, and T. Rabin, "Falcon: Honest-majority maliciously secure framework for private deep learning," *arXiv preprint arXiv:2004.02229*, 2020.

[16] P. Mohassel and Y. Zhang, "Secureml: A system for scalable privacy-preserving machine learning," in *IEEE Symposium on Security and Privacy*, 2017, pp. 19–38.

[17] G. Xu, H. Li, S. Liu, K. Yang, and X. Lin, "Verifynet: Secure and verifiable federated learning," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 911–926, 2019.

[18] P. Li, J. Li, Z. Huang, T. Li, C. Gao, S. Yiu, and K. Chen, "Multi-key privacy-preserving deep learning in cloud computing," *Future Generation Computer Systems*, vol. 74, pp. 76–85, 2017.

[19] X. Ma, J. Ma, H. Li, Q. Jiang, and S. Gao, "Pdlm: Privacy-preserving deep learning model on cloud with multiple keys," *IEEE Transactions on Services Computing*, 2018.

[20] L. Xie, I. M. Baytas, K. Lin, and J. Zhou, "Privacy-preserving distributed multi-task learning with asynchronous updates," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 1195–1204.

[21] P. Fenner and E. O. Pyzer-Knapp, "Privacy-preserving gaussian process regression–a modular approach to the application of homomorphic encryption," *arXiv preprint arXiv:2001.10893*, 2020.

[22] X. Liu, R. H. Deng, K.-K. R. Choo, and J. Weng, "An efficient privacy-preserving outsourced calculation toolkit with multiple keys," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 11, pp. 2401–2414, 2016.

[23] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser, "Machine learning classification over encrypted data," in *Proceedings of the 22nd Annual Network and Distributed System Security Symposium*, 2015, pp. 1–14.
[24] O. Goldreich, *The Foundations of Cryptography - Volume 2, Basic Applications*. Cambridge University Press, 2004.