SCIENCE CHINA
Information Sciences

• LETTER •

May 2024, Vol. 67, Iss. 5, 159102:1–159102:2
https://doi.org/10.1007/s11432-023-3977-9

Efficient privacy-preserving federated learning under dishonest-majority setting

Yinbin MIAO[1], Da KUANG[1], Xinghua LI[1], Tao LENG[2,3,4*],
Ximeng LIU[5] & Jianfeng MA[1]

[1]*School of Cyber Engineering, Xidian University, Xi'an* 710071*, China;*
[2]*Intelligent Policing Key Laboratory of Sichuan Province, Sichuan Police College, Luzhou* 646000*, China;*
[3]*Institute of Information Engineering, Chinese Academy of Sciences, Beijing* 100085*, China;*
[4]*School of Cyber Security, University of Chinese Academy of Sciences, Beijing* 100049*, China;*
[5]*Key Laboratory of Information Security of Network Systems, Fuzhou University, Fuzhou* 350108*, China*

Received 24 March 2023/Revised 29 July 2023/Accepted 30 January 2024/Published online 24 April 2024

Federated learning (FL) is an emerging distributed learning paradigm that solves the problem of isolated data by jointly learning the global model through distributed clients. However, recent studies have shown that FL may not always guarantee sufficient privacy preservation, this is mainly because the model parameters (e.g., weights or gradients) may leak sensitive information to malicious adversaries. Thus, many privacy-preserving FL (PPFL) solutions have attracted much attention from both academic and industrial fields. However, there are still some issues to be solved. The first issue is that existing PPFL solutions incur high costs or low model accuracy. Although homomorphic encryption-based PPFL (HE-based PPFL) [1] can achieve secure aggregation, it still incurs high computation overhead due to complex operations over encrypted data. Secure multi-party computation-based PPFL (SMC-based PPFL) [2] eliminates time-consuming operations of HE, but it still brings high communication costs due to multiple interactions among clients. To avoid the high costs in above solutions, differential privacy-based PPFL (DP-based PPFL) [3] only adds noises to perturb local model parameters, thereby achieving privacy protection, but this method may lead to low model accuracy. Thus, the above PPFL solutions are still not well deployed in practical applications. The second issue is that existing PPFL still cannot provide an efficient defense mechanism under dishonest-majority setting. Some existing PPFL [4] eliminates discrepant malicious data by computing the cosine similarity between client-side model updates for comparison, thereby defending against model poisoning attacks. But this method cannot support the dishonest-majority setting, and incurs high costs as the operation of calculating the cosine similarity under the ciphertext is very complicated. More recently, PPFL [5] computes the similarity between model updates by comparing the Hamming distance between trust roots and model updates. It can support dishonest-majority setting, and its cost of calculating Hamming distance under ciphertext is less than that of calculating cosine, but this method will bring additional complex operations to support the calculation of Hamming distance, such as converting from boolean sharing to arithmetic sharing (Bit2A).

From above discussions, we propose an efficient PPFL with a defense mechanism to resist model poisoning attacks in dishonest-majority settings, balancing robustness with the efficiency and privacy protection required in practical applications. Specifically, we propose a sample-based secret sharing method to build efficient security modules. This method is designed according to the characteristics of Hamming distance calculation, which can avoid the complex operations brought by Hamming distance calculation under ciphertext, such as Bit2A. In addition, we use the root model update and adaptive learning rate to compensate for the lack of valid information caused by malicious clients, thereby improving the accuracy of the model. To reduce the computational overhead, we adopting MPC to build the privacy module. The specific algorithm details will be given in the Appendix A. Here we only introduce the overall algorithm flow. We divide the privacy framework into four parts, including the preprocessing stage, the stage for calculating Hamming distance and weight, the decoding and multiplication stage, and the model aggregation stage. Next, we describe its specific process in Algorithm 1.

Preprocessing. Each client C_i loads the global model w^τ of round τ and uses the local dataset to train the local model w_i^τ, and then calculates the model update $\nabla w_i^\tau = w^\tau - w_i^\tau$. Next we quantify ∇w_i^τ to obtain the model update ∇w_i^q. In addition to facilitating the calculation of Hamming distance, C_i needs to code the ∇w_i^q by $\nabla w_i^e = \lceil (1 - \nabla w_i^q)/2 \rceil$. Finally, as the secret share distributor, C_i first generates the secret shares $[\![\nabla w_i^e]\!]_0, [\![\nabla w_i^e]\!]_1$ locally, and sends them to the secret share holders $\mathcal{S}_0$ and $\mathcal{S}_1$ respectively. Similarly, the server $\mathcal{S}_0$ holds the root of data set, and performs the preprocessing process as described above. Specifically, $\mathcal{S}_0$ performs training based on the root data set, and calculates the local model update ∇w_s^τ, and then performs quantization and encoding to obtain ∇w_s^e. And as the holder and distributor of ∇w_s^e, $\mathcal{S}_0$ generates the secret shares $[\![\nabla w_s^e]\!]_0, [\![\nabla w_s^e]\!]_1$, and keeps $[\![\nabla w_s^e]\!]_0$, and sends $[\![\nabla w_s^e]\!]_1$ to $\mathcal{S}_1$. Specific details will

* Corresponding author (email: lengtao@iie.ac.cn)

be described in the algorithm PO$(\cdot)$ in Appendix A.1.

Algorithm 1 FESD

```
Input: $\alpha, w^0$, number of clients $n$, number of communication rounds T.
Output: Global model $w^\tau$.
1: Servers compute $[\![a]\!], [\![b]\!], [\![c]\!] \leftarrow$ F.GMT $(\mathcal{S}_0, \mathcal{S}_1)$ //Generate multiplicative triples in the offline stage;
2: for $\tau = 0, 1, \ldots, T-1$ do
3:     $\mathcal{S}_0$ selects clients $\mathcal{C}^t = \{C_0, \ldots, C_{n-1}\}$ and then computes $[\![\nabla w_s^e]\!]_0, [\![\nabla w_s^e]\!]_1 \leftarrow$ F.PO$(w^\tau)$;
4:     for $i = 0, \ldots, n-1$ do
5:         $C_i \in \mathcal{C}^t$ computes $[\![\nabla w_i^e]\!]_0, [\![\nabla w_i^e]\!]_1 \leftarrow$ F.PO$(w^\tau)$;
6:         Calculate Hamming distance:
7:         $\mathcal{S}_0$ and $\mathcal{S}_1$ separately compute $[\![Hd]\!]_b \leftarrow H([\![\nabla w_i^e]\!]_b \oplus [\![\nabla w_s^e]\!]_b), b \in \{0,1\}$;
8:         Calculate client weight:
9:         $[\![v]\!]_0, [\![v]\!]_1 \leftarrow$ GMW$(\mu - [\![Hd_i]\!]_0, -[\![Hd_i]\!]_1)$;
10:        Decoding:
11:        $[\![\nabla w_*^d]\!]_b = b - 2 \times [\![\nabla w_*^e]\!]_b, b \in \{0,1\}$;
12:        Multiplication:
13:        $[\![\nabla w_i^\tau]\!]_0, [\![\nabla w_i^\tau]\!]_1 \leftarrow$ SecMul$([\![v_i]\!], [\![\nabla w_i^d]\!])$;
14:        Model aggregation:
15:        $[\![w^{\tau+1}]\!]_b = [\![w^\tau]\!]_b - \alpha \times \left(\sum_i [\![\nabla w_i^\tau]\!]_b + [\![\nabla w_s^d]\!]_b\right), b \in \{0,1\}$;
16:    end for
17: end for
18: return $w^T$.
```

Calculate Hamming distance and weight. To detect malicious parameters, we use Hamming distance to measure the similarity between the trusted root $[\![\nabla w_s^e]\!]$ and $[\![\nabla w_i^e]\!]$ uploaded by C_i. If the calculated Hamming distance is greater than the parameter μ, C_i is judged to be corrupted. Then we calculate the weight value of C_i for security aggregation.

Specifically, $\mathcal{S}_0$ and $\mathcal{S}_1$ respectively compute the secret shares $[\![Hd]\!]_0, [\![Hd]\!]_1$ of the Hamming distance between $[\![\nabla w_s^e]\!]$ and $[\![\nabla w_i^e]\!]$ through XOR and $H(\cdot)$, where $H(\cdot)$ is the number of 1 in the binary stream. In the entire process, operations are performed locally, which can ensure the privacy and security of the data (Line 7 in Algorithm 1). Aiming to guarantee that servers do not disclose their respective secret shares while calculating the client's weight value, we use GMW to calculate the secret share $[\![v_i]\!]_0, [\![v_i]\!]_1$ of the client weight value (Line 9 in Algorithm 1). The structure of circuit is shown in Appendix A.2.

Decoding and multiplication. Before updating the global model, the gradient needs to be decoded for weighting. Specifically, $\mathcal{S}_0$ and $\mathcal{S}_1$ decode secret shares $[\![\nabla w_*^e]\!]_0, [\![\nabla w_*^e]\!]_1$ by (1) locally, where $* \in \{i, s\}$. Then servers calculate the weighted model update $[\![\nabla w_i^\tau]\!] = [\![v_i]\!] \times [\![\nabla w_i^d]\!]$ (Lines 10–13 in Algorithm 1). To calculate multiplication under secret shares, $\mathcal{S}_0$ needs to generate multiplicative triples $[\![a]\!], [\![b]\!], [\![c]\!]$ according to Algorithm GMT$(\cdot)$ in Appendix A.3, which can be performed in the offline phase. Finally, we securely calculate $[\![\nabla w_i^\tau]\!]$ according to Algorithm SecMul$(\cdot)$ in Appendix A.4.

$$[\![\nabla w_*^d]\!]_b = b - 2 \cdot [\![\nabla w_*^e]\!]_b, \quad b \in \{0,1\}. \tag{1}$$

Model aggregation. $\mathcal{S}_0$ and $\mathcal{S}_1$ respectively calculate secret shares $[\![w^{\tau+1}]\!]_0, [\![w^{\tau+1}]\!]_1$ by (2) and then send $[\![w^{\tau+1}]\!]_0$, $[\![w^{\tau+1}]\!]_1$ to each client C_i.

$$[\![w^{\tau+1}]\!]_b = [\![w^\tau]\!]_b - \alpha \times \left(\sum\nolimits_i [\![\nabla w_i^\tau]\!]_b + [\![\nabla w_s^d]\!]_b\right), b \in \{0,1\}, \tag{2}$$

where α is the global learning rate. In particular, we adopt an adaptive global learning rate to better improve the model performance, we set $\alpha = \alpha/(1-\delta)$, where δ is the fraction of corrupt clients detected in this round of communication and δ can be estimated by whether v_i is equal to 0.

As shown in Appendixes A–C, the algorithm details, security analysis, performance analysis are described respectively.

Conclusion and future work. We propose an efficient PPFL (FESD), which can defend against model poisoning attacks in the dishonest-majority setting and guarantees that the privacy will not be compromised. Along the way, we also propose a sampling-based secret sharing method that can completely avoid Bit2A, thus greatly reducing the computational overhead. In future research, we will focus on combining the latest security multi-party technologies to improve the efficiency of our framework, such as adopting OT-Extension.

Acknowledgements This work was supported by National Natural Science Foundation of China (Grant Nos. 62072361, 62125205, U23A20303), Key Research and Development Program of Shaanxi (Grant No. 2022GY-019), Shaanxi Fundamental Science Research Project for Mathematics and Physics (Grant No. 22JSY019), Opening Project of Intelligent Policing Key Laboratory of Sichuan Province (Grant No. ZNJW2023KFMS002), and Open Fund of Key Laboratory of Computing Power Network and Information Security (Grant No. 2023ZD020).

Supporting information Appendixes A–C. The supporting information is available online at info.scichina.com and link.springer.com. The supporting materials are published as submitted, without typesetting or editing. The responsibility for scientific accuracy and content remains entirely with the authors.

References

1 Phong L T, Aono Y, Hayashi T, et al. Privacy-preserving deep learning via additively homomorphic encryption. IEEE TransInformForensic Secur, 2018, 13: 1333–1345
2 Sharma S, Xing C, Liu Y, et al. Secure and efficient federated transfer learning. In: Proceedings of IEEE International Conference on Big Data, 2019. 2569–2576
3 Girgis A M, Data D, Diggavi S, et al. Shuffled model of federated learning: privacy, accuracy and communication trade-offs. IEEE J Sel Areas Inf Theor, 2021, 2: 464–478
4 Nguyen T D, Rieger P, Yalame H, et al. Flguard: secure and private federated learning. 2021. ArXiv:2101.02281
5 Dong Y, Chen X, Li K, et al. Flod: oblivious defender for private byzantine-robust federated learning with dishonest-majority. In: Proceedings of European Symposium on Research in Computer Security (ESORICS'21), 2021. 497–518