Proceedings of Machine Learning Research vol 245:1–9, 2024 2024 International Conference on Machine Learning and Intelligent Computing

Decentralized Federated Learning Algorithm Based on Federated Groups and Secure Multiparty Computation

Fan Wu[*] WUFANGARY@163.COM
College of Information Engineering, Shanghai Maritime University, Shanghai, 201306, China

Maoting Gao
College of Information Engineering, Shanghai Maritime University, Shanghai, 201306, China

Editors: Nianyin Zeng and Ram Bilas Pachori

Abstract

To solve the problem that the centralized federal learning based on privacy protection relies on trusted central servers, has low resistance to malicious attacks, and is prone to privacy leakage, this paper proposes a decentralized federated learning algorithm based on federated groups and secure multiparty computation. By establishing a federated group mechanism based on model relevance, each client has its own federated group, and model parameters are only transmitted among federated group members, members outside the group are unable to access parameter information. Secret owners utilize secret sharing algorithms to split their model parameters into several secret shares, which are then transmitted to federated group members through secure channels. Federated group members then aggregate all transmitted secret shares by weighted averaging, and the secret owner receives the aggregated secret shares passed back from all federated group members, and then uses the secret recovery algorithms to recover secret, and obtains the updated parameter model. In the federated group, while a member becomes a Byzantine node, it is removed from the federated group, and another client is selected to join the group based on model relevance. So, each client participating in federated learning serves as both a data node and a computing node, federated learning eliminates reliance on servers and achieves decentralization. The good privacy performance of the proposed algorithm model is theoretically analyzed, and experiments on the FedML platform demonstrated that the algorithm has stronger resistance to attacks.

Keywords: federated learning, decentralization, privacy protection, federated groups, secure multiparty computation

1. Introduction

Federated learning (Rahman et al., 2021; McMahan et al., 2016a,b) is a method for multiple untrusted clients to collectively build a model aimed at obtaining a more accurate global model (Zhao et al., 2020). In this approach, clients use local data for model training and then transmit parameters to a central server for aggregation. However, the transmission process is susceptible to interception by attackers, leading to leakage of model parameters and potentially inferring the original data of clients (Mohassel and Zhang, 2017; LI et al., 2022). Even if attackers cannot directly access the parameters transmitted by clients, they can obtain the aggregated results from the central server . When encountering malicious central servers or black-box attacks, client data may be compromised.

To counter attackers' inference attacks, Lyu et al. (2022) proposed several common privacy protection strategies, such as homomorphic encryption, secure multiparty computation, and differential privacy. Sun et al. (2020) introduced the concept of Local Differential Privacy (LDP) to address the problem of information leakage in federated learning due to noise introduced in privacy protection being close to the original data. At the server-side, Gaussian noise is added to conceal the data of individual participants. Bourse et al. (2018) proposed the homomorphic evaluation framework FHE-DiNN based on fully homomorphic encryption, combining fully homomorphic encryption and discrete neural networks to provide a linear complexity neural network related to network depth. This framework improves the speed of fully homomorphic encryption and reduces complexity, but also reduces accuracy. Di et al. (2022), combining secure multiparty computation and differential privacy, perturb the local parameter model obtained by clients during local training and use Cramer's secret sharing scheme (Cramer et al., 2000) to share to multiple central servers. In the server, local parameter models are aggregated into a shared secret global model through multi-party computation protocols. This method can greatly prevent client data leakage. Since it depends on central servers, it cannot resist attackers' black-box attacks on servers. Moreover, when encountering Byzantine nodes, there may be a decrease in accuracy. Byzantine nodes refer to nodes whose behavior is arbitrary and may result in message loss, delay, disorder, or duplication. When a node becomes a Byzantine node, it is very likely to send incorrect messages or fail to transmit parameter shares within the specified time, leading to a decrease in overall accuracy.

This paper proposes a decentralized federated learning model based on federated groups and secure multiparty computation (FG-SMC-FL), aiming to address the vulnerability of traditional federated learning to inference attacks. This model combines two privacy protection methods: secure multiparty computation and differential privacy, by creating federated groups to replace central servers, effectively protecting the security of client data. Each client is assigned to a federated group based on model relevance, and parameter shares are transmitted within the federated group to prevent poisoning attacks by attackers. When members of a federated group become Byzantine nodes, the system removes them and selects new members to ensure the security of user privacy. This model not only eliminates the dependency on central servers but also transforms all clients into data nodes and computing nodes, making better use of node resources.

2. Secure Multiparty Computation

Secure Multiparty Computation (SMC) is a specialized computing protocol designed to enable collaborative computation among a group of mutually untrusting participants without the presence of a trusted third-party server. Its key techniques include secret sharing, oblivious transfer, and garbled circuits. Secret sharing distributes secret information among participants and allows for the reconstruction of the secret information using a secret reconstruction algorithm when enough shares are collected.

The task of secret sharing is to split a secret into shares and distribute the shares among several participants, where recipients need to collect a certain number of shares from participants who possess the secret to reconstruct it.

This paper employs an efficient secret sharing scheme proposed by Cramer et al. (2000), known as the (n, n)-threshold secret sharing scheme. Assuming the secret holder D needs to communicate the secret to a set of participants P $\{P_1, P_2, ..., P_\mathrm{n}\}$, where the nth secret share is represented by $s_n = s - \sum_{i=1}^{n-1} s_i \bmod 2^l$. Here, $s \in Z_{2^l}$ and 2^l refers to a sufficiently large integer greater than any coefficient, aiming to prevent attackers from inferring the secret through inference attacks.

To reconstruct the secret, any k members can combine their respective shares of the secret and use Lagrange interpolation formula to recover the secret s. Let the k members each possess a secret share; the original secret can be derived using $s = \sum_{i=1}^{n} s_i \bmod 2^l$.

3. Method Design

The model proposed, named FG-SMC-FL (Federated Group and Secure Multiparty Computation Federated Learning), aims to address the vulnerability of models like that proposed by Tang et al., which rely on multiple central servers and are prone to malicious attacks. To eliminate dependence on central servers, FG-SMC-FL employs federated groups based on model relevance and secure multiparty computation.

In FG-SMC-FL, secret owners use secret sharing algorithms to divide their model parameters into several secret shares. These shares are then distributed to members of federated groups through secure channels. Each member aggregates all received secret shares using weighted averaging to obtain aggregated parameters. The aggregated parameters are then transmitted back to the secret owners. The secret owners utilize secret recovery algorithms to reconstruct the secret, obtaining updated model parameters. This process iterates until model convergence.

If any member of a federated group becomes a Byzantine node, they are removed from the group, and a new client is selected based on model relevance. The structure of the FG-SMC-FL model is illustrated in Figure 1.

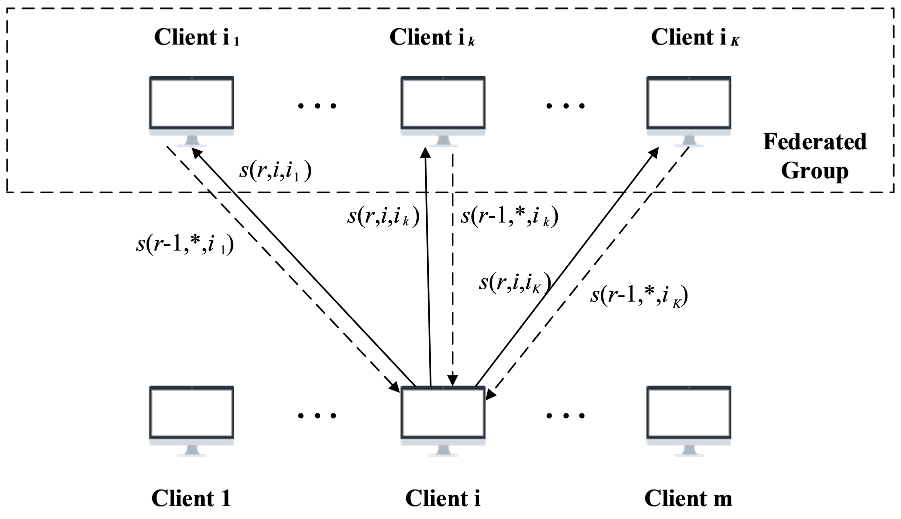

Figure 1: A federated learning framework based on federated groups and secure multiparty computation.

In Figure 1, r represents the current round, client i is the secret owner, and client i first selects K clients with the highest relevance based on model relevance as federated group members, and sequentially labels the clients as $\{i_1, i_2, ..., i_k, ..., i_K\}$. s represents the secret shares transmitted between clients, where $s(r,\ i,\ i_k)$ denotes the secret share transmitted from client i to client i_k in round r.

The model establishes a federated group based on model relevance, reducing the number of distributed secret shares to address communication overhead among multiple participants. Additionally, it replaces the server-side with clients to achieve decentralization, thereby mitigating threats from malicious server-side entities. Before introducing specific solutions, let's briefly introduce the symbols used, as shown in Table 1.

Table 1: Symbol Description Table.

Symbol	Description
N	The number of participants
$b_{i,j}$	The model relevance between participant i and participant j
α	Hyperparameter
w_i^r	The client of participant i in round r
r	The current round
s	Secret shares transmitted between clients
i	Client identifier
D_i	The dataset of client i
i_k	Member identifier of the federated group where client i belongs
$s(r, i, i_k)$	Secret share transmitted from client i to client i_k in round r
K	Upper limit of federated group members
s	Secret shares transmitted between clients

3.1. Federated Group Established Based on Model Relevance

If clients train their models in similar application contexts, they can learn more from the training process. Therefore, clients tend to participate in federated learning with other participants whose model applications are similar to theirs. To identify these participants similar to the current client, a federated group can be established based on model relevance. All participants first send their personalized parameters to all other participants. The secret owner selects K clients with the highest model relevance to form a federated group. The model relevance between participant i and participant j is calculated using the following equation:

$$b_{i,j} = e^{\alpha \cos(w_i^{r-1}, w_j^{r-1})} \tag{1}$$

where α is a hyperparameter, $\cos(w_i^{r-1}, w_j^{r-1})$ representing the cosine similarity between w_i^{r-1} and w_j^{r-1}, and higher values of $b_{i,j}$ indicate smaller differences between w_i^{r-1} and w_j^{r-1}. Participant i selects the K members with the highest $b_{i,j}$ values to form the federated group.

3.2. Federated Learning Model Based on Federated Groups and Secure Multiparty Computation

To address the issue of reliance on a trusted third-party central server in centralized federated learning, this paper proposes a federated learning model based on model relevance, which creates a federated group for each client. Clients establish federated groups by selecting *K* members based on model relevance. Each client receives initialized parameter shares from members within the federated group, then locally restores them, conducts local training, and obtains new parameters for the next round. Subsequently, using secret sharing algorithms, the client divides the secret into *K* secret shares and transmits them to other members of the group. Upon receiving all secret shares, federated group members aggregate them using weighted averaging to obtain aggregate parameters, which are then transmitted back to the secret owner. The secret owner employs secret recovery algorithms to restore the secret and obtain the updated parameter model, repeating these steps until achieving the desired objectives.

Compared to centralized federated learning, which requires a reliable central server, the decentralized federated learning framework based on federated groups and secure multiparty computation allows any client to act as a secret distributor or federated group member. This eliminates the reliance on a reliable central server and mitigates security risks associated with storing model parameters on a single server.

Here is the detailed algorithm for the proposed model:

```
Input: Machine learning algorithm $M$, client set $C = \{C_1, \ldots, C_m\}$, local dataset $D = \{D_1, \ldots, D_m\}$, federated group A, differential privacy parameters $\varepsilon$ and $\delta$, minimum number of clients for parameter uploading $K$, total training rounds $R$, model pruning weight $P$.
Output: Trained model $S$.
1. for $r \leftarrow 1$ to $R$
2.   for $C_i \in C$ do
3.     if $r = 1$
4.         Initialize model parameters
5.         Select $t$ clients with high model relevance from $C$ to join federated group $A$, $A = \{A_1, \ldots, A_t\}$
6.         Download initial global model $s_0$ from $A_1$
7.     else
8.         Download shares $s(r-1, *, j)$ from $A_j$, $j \in \{1, \ldots, t\}$
9.         Restore $s_r - 1 \leftarrow$ SecRec($s(r-1, *, 1), \ldots, s(r-1, *, t)$)
10.      end if
11.      Local training $s^r{}_i \leftarrow M(s_{r-1}, Di)$
12.      Prune weights $s^r{}_i \leftarrow \max\,(1, \| s^r{}_i \| / P)$
13.      Add noise $s^r{}_i \leftarrow s^r{}_i$ + noise($\varepsilon$, $\delta$, $P$, $K$)
14.      Calculate shares $(s(r,*,1), \ldots, s(r,*,t)) \leftarrow$ SecShr$(s^r{}_i, t)$
15.      Send $s(r, i, j)$ to $s_j$, $j \in \{1, \ldots, t\}$
16.   end for
17.   for $A_j \in A$ do
18.      Wait until enough parameters $s(r, i_1, j), \ldots, s(r-1, i_K, j)$ are collected
19.      Aggregate parameter shares: $s(r, i_1, j) \leftarrow (s(r, i_1, j) + ... + s(r-1, i_K, j))/K$
20.   end for
21. end for
22. Download parameter shares and restore
23. Output final model S $\leftarrow s_R$
```

Given the instability of networks in practical application scenarios, the above algorithm can tolerate client disconnections to some extent. If a client disconnects, it will simply skip the current round of communication loop and not participate in the current round of training. Since the secret sharing algorithm distributes shares to all federation group members in parallel after splitting, when a client disconnects, it will not send parameter shares to group members. Additionally, since federated groups are established, if a member within the group becomes a Byzantine node, meaning there are delays in message reception or message sending failures, the system will remove that node and select a replacement based on model relevance from the participants. This ensures that data will not be leaked by corrupted user endpoints and that missing group members will not prevent the recovery of parameter shares.

3.3. Privacy Analysis

Currently, federated learning still relies on a central server and is susceptible to poisoning attacks and inference attacks. This section analyzes and demonstrates the privacy protection effectiveness of the proposed algorithm FG-SMC-FL and compares it with the algorithm proposed by Tang.

Since conventional federated learning relies on a reliable third-party central server, if the server is subjected to poisoning attacks and is compromised, the privacy of all client data participating in federated learning will become transparent. Some semi-honest participants may attempt to infer sensitive data of target clients by obtaining the gradient parameters uploaded by the target clients. To address this, the proposed algorithm provides a decentralized federated learning architecture where all participating clients can perform calculations without the need for any trusted central server. Each participant acts as both a data node and a computing node, ensuring that when the number of adversaries is within a certain range, client data participating in federated learning cannot be stolen.

Theorem 1: When the number of untrustworthy clients is less than t, where $0 \leq t \leq N - 1$ and N is the number of clients participating in federated learning, adversaries cannot obtain any data from any trustworthy client.

Proof: For any trustworthy client C_i, considering that corrupted clients can only receive parameter shares $s(r, i, j)$ updated from C_i each round; when the number of corrupted clients is less than t, adversaries cannot recover any parameter $s(r, i, j)$, and thus cannot obtain any useful information from the parameter shares. Moreover, since adversaries cannot obtain parameters, they cannot aggregate all collected parameter shares or send aggregated parameters to the secret owner, resulting in being identified as Byzantine nodes and removed from the federated group. Compared to the algorithm proposed by Tang, which needs to consider whether the server is corrupted, the overall privacy of the proposed algorithm is higher.

4. Experimental Results and Analysis

To verify the privacy protection performance of the proposed algorithm, we analyze and evaluate it from three aspects: privacy, resilience to attacks, and efficiency. Section 2.3 has already demonstrated the privacy of the proposed algorithm model, and this section mainly tests the accuracy of the algorithm through experiments.

The experiments were conducted on the FedML platform, an excellent federated learning framework. Cloud servers were used to simulate communication between multiple client nodes on the FedML platform. Python was used to implement local model training, creation of federated groups, secure multiparty computation, and communication between nodes after adding differential privacy.

The MNIST dataset was used for the experiments, and a convolutional neural network was employed for training. The neural network consists of two convolutional layers with kernel sizes of 5∗5 and strides of 10 and 20.

4.1. Experiment 1: Resilience to Attacks

To analyze and verify the resilience to attacks of the proposed algorithm, we compared the training results after 100 communication rounds between the proposed algorithm and the algorithm proposed by Tang Lingtao et al., both with and without Byzantine nodes. The accuracy under different training methods is shown in Table 2.

Table 2: Accuracy under different training method.

	Scenario	Method Accuracy
No Byzantine Nodes	TANG	96.12
	FG-SMC-FL	96.20
With Byzantine Nodes	TANG	90.58
	FG-SMC-FL	95.12

From the table, it can be observed that in the absence of Byzantine nodes, the accuracy of the proposed algorithm and Tang's algorithm is very similar, demonstrating that the proposed algorithm can replace the central server functionality in Tang's algorithm. However, Tang's algorithm has lower accuracy when facing Byzantine nodes. This is because although Tang's algorithm can tolerate client disconnections to some extent, persistent disconnection of a client will affect the accuracy of the overall results. In contrast, the proposed algorithm replaces the disconnected Byzantine node with a new client based on model relevance, resulting in higher accuracy.

4.2. Experiment 2: Efficiency Analysis

In this experiment, we compared the proposed algorithm with Tang Lingtao's algorithm to test the efficiency of the model. The total time taken for the proposed model and Tang's algorithm under different data volumes (3000 and 6000) is shown in the Figure 2.

From the graph, it can be seen that the proposed solution is very similar to Tang's algorithm. The reasons for this are as follows:

1. In the proposed model, communication occurs only between the secret holder and federated group members each round, and communication between members outside the federated group is not involved. Additionally, communication between the secret holder and federated group members is parallel. Therefore, compared to Tang's algorithm, there is no significant increase in communication overhead.

2. The amount of communication between nodes remains unchanged. Both Tang's algorithm and the proposed algorithm train data locally, so only parameters are transmitted between nodes without additional data.

3. The time required for aggregation remains unchanged. Both algorithms average the parameters locally on the client side without the need for interaction.

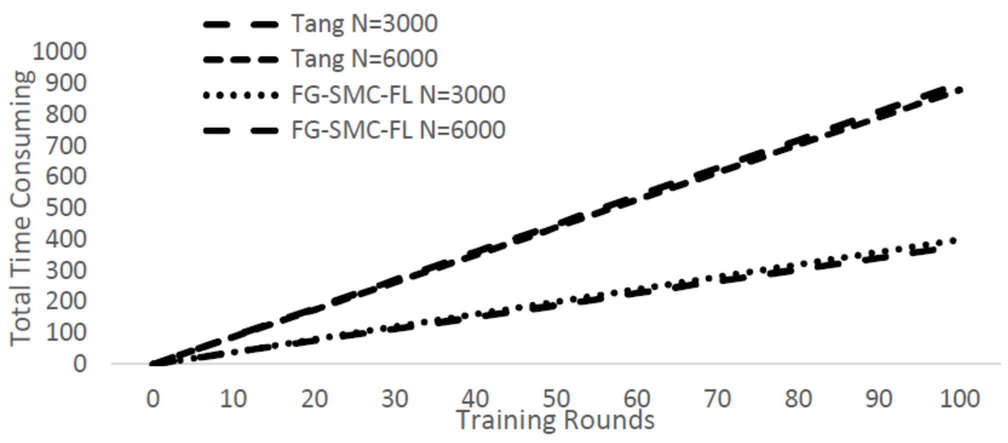

Figure 2: Total training time under different data volumes.

5. Conclusion

Through the study of existing federated learning and privacy protection methods, a decentralized federated learning method based on federated groups and secure multiparty computation was proposed. Secure multiparty computation is used to secret share client model parameters, and the client's model parameters are split into several secret shares to prevent inference attacks. A federated group scheme based on model relevance was designed to replace the central server with participating clients, achieving decentralization. The model currently considers semi-honest clients, assuming that the original parameter model is reliable during secret transmission. However, dealing with malicious clients, who may transmit maliciously tampered data during secret sharing, will be a future challenge.

References

Florian Bourse, Michele Minelli, Matthias Minihold, and Pascal Paillier. Fast homomorphic evaluation of deep discretized neural networks. In *Advances in Cryptology–CRYPTO 2018: 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19–23, 2018, Proceedings, Part III 38*, pages 483–512. Springer International Publishing, 2018.

Ronald Cramer, Ivan Damgård, and Ueli Maurer. General secure multi-party computation from any linear secret-sharing scheme. In *International Conference on the Theory and Applications of Cryptographic Techniques, Berlin, Heidelberg: Springer Berlin Heidelberg*, pages 316–334, 2000.

WANG Di, ZHANG Lufei, et al. Federated learning scheme based on secure multi-party computation and differential privacy. *Computer Science*, 49(9):297–305, 2022.

Shaobo LI, Lei YANG, Chuanjiang LI, Ansi ZHANG, and Ruishi LUO. Overview of federated learning: Technology, applications and future. *Computer Integrated Manufacturing System*, 28 (7):2119, 2022.

Lingjuan Lyu, Han Yu, Xingjun Ma, Chen Chen, Lichao Sun, Jun Zhao, Qiang Yang, and S Yu Philip. Privacy and robustness in federated learning: Attacks and defenses. *IEEE transactions on neural networks and learning systems*, 2022.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282, 2016a.

H Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Agüera y Arcas. Federated learning of deep networks using model averaging. *arXiv preprint arXiv:1602.05629*, 2(2), 2016b.

Payman Mohassel and Yupeng Zhang. Secureml: A system for scalable privacy-preserving machine learning. In *2017 IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA*, pages 19–38, 2017.

KM Jawadur Rahman, Faisal Ahmed, Nazma Akhter, Mohammad Hasan, Ruhul Amin, Kazi Ehsan Aziz, AKM Muzahidul Islam, Md Saddam Hossain Mukta, and AKM Najmul Islam. Challenges, applications and design aspects of federated learning: A survey. *IEEE Access*, 9:124682–124700, 2021.

Lichao Sun, Jianwei Qian, and Xun Chen. Ldp-fl: Practical private aggregation in federated learning with local differential privacy. In *The Thirtieth International JointConference on Artificial Intelligence, Montreal, Canada*, pages 1571–1578, 2020.

Qi Zhao, Chuan Zhao, Shujie Cui, Shan Jing, and Zhenxiang Chen. Privatedl: privacy-preserving collaborative deep learning against leakage from gradient sharing. *International Journal of Intelligent Systems*, 35(8):1262–1279, 2020.