JUSTC

Article

http://justc.ustc.edu.cn

Received: August 25, 2022; Accepted: January 12, 2023

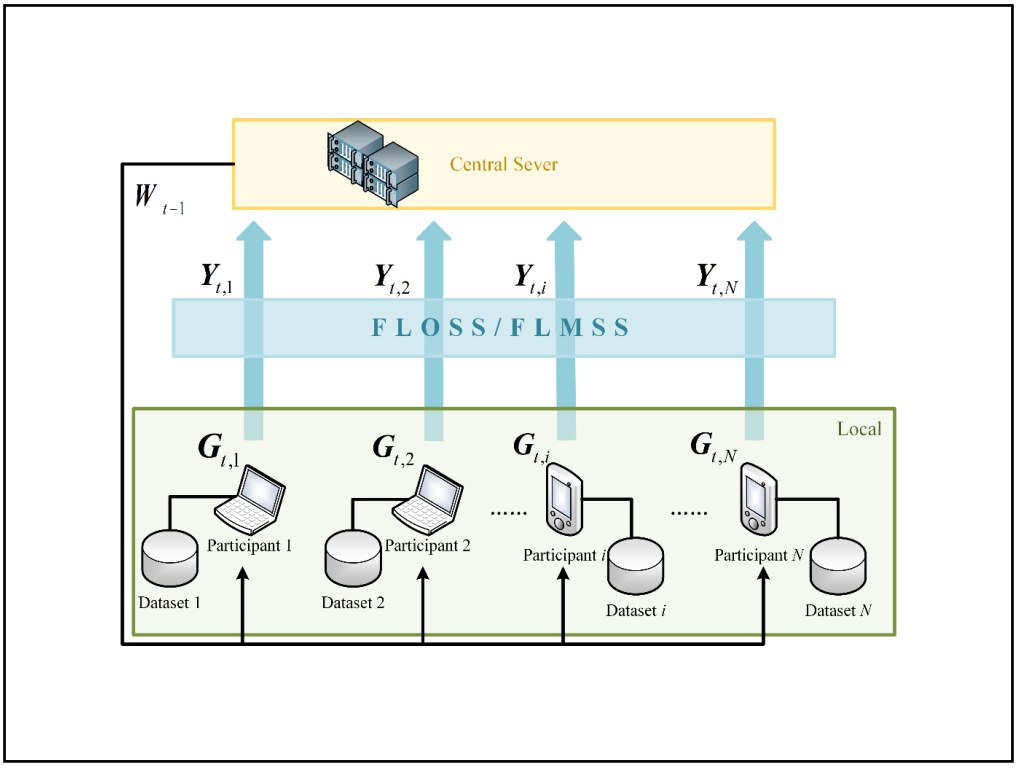
Efficient secure aggregation for privacy-preserving federated learning based on secret sharing

Xuan Jin[1], Yuanzhi Yao[2] ✉, and Nenghai Yu[1]

[1]*School of Cyber Science and Technology, University of Science and Technology of China, Hefei 230027, China;*
[2]*School of Computer Science and Information Engineering, Hefei University of Technology, Hefei 230601, China*

✉Correspondence: Yuanzhi Yao, E-mail: yaoyz@hfut.edu.cn

Graphical abstract



In our proposed privacy-preserving federated learning schemes, participants' local training data can be strongly protected with low cost.

Public summary

- The privacy-preserving federated learning scheme based on one-way secret sharing (FLOSS) is proposed to enable high privacy preservation while significantly reducing the communication cost by dynamically designing secretly shared content and objects.
- The privacy-preserving federated learning scheme based on multi-shot secret sharing (FLMSS) is proposed to further reduce the additional communication-computation cost and enhance the robustness of participant dropouts.
- Extensive security analysis and performance evaluations demonstrate the superiority of our proposed schemes in terms of model accuracy, privacy preservation, and cost reduction.

Citation: Jin X, Yao Y Z, Yu N H. Efficient secure aggregation for privacy-preserving federated learning based on secret sharing. *JUSTC*, **2024**, 54(1): 0104. DOI: 10.52396/JUSTC-2022-0116

JUSTC

Article

http://justc.ustc.edu.cn

Received: August 25, 2022; Accepted: January 12, 2023

Efficient secure aggregation for privacy-preserving federated learning based on secret sharing

Xuan Jin[1], Yuanzhi Yao[2] ✉, and Nenghai Yu[1]

[1]*School of Cyber Science and Technology, University of Science and Technology of China, Hefei 230027, China;*
[2]*School of Computer Science and Information Engineering, Hefei University of Technology, Hefei 230601, China*

✉Correspondence: Yuanzhi Yao, E-mail: yaoyz@hfut.edu.cn

Cite This: *JUSTC*, **2024**, 54(1): 0104 (15pp)

Read Online

Abstract: Federated learning allows multiple mobile participants to jointly train a global model without revealing their local private data. Communication-computation cost and privacy preservation are key fundamental issues in federated learning. Existing secret sharing-based secure aggregation mechanisms for federated learning still suffer from significant additional costs, insufficient privacy preservation, and vulnerability to participant dropouts. In this paper, we aim to solve these issues by introducing flexible and effective secret sharing mechanisms into federated learning. We propose two novel privacy-preserving federated learning schemes: federated learning based on one-way secret sharing (FLOSS) and federated learning based on multi-shot secret sharing (FLMSS). Compared with the state-of-the-art works, FLOSS enables high privacy preservation while significantly reducing the communication cost by dynamically designing secretly shared content and objects. Meanwhile, FLMSS further reduces the additional cost and has the ability to efficiently enhance the robustness of participant dropouts in federated learning. Foremost, FLMSS achieves a satisfactory tradeoff between privacy preservation and communication-computation cost. Security analysis and performance evaluations on real datasets demonstrate the superiority of our proposed schemes in terms of model accuracy, privacy preservation, and cost reduction.

Keywords: federated learning; privacy preservation; secret sharing; secure aggregation

CLC number: TP309.2 **Document code:** A

1 Introduction

Computational models generated by deep learning are able to learn valuable representations of data[1]. Deep learning has dramatically improved artificial intelligence in many domains, such as object recognition[2], object classification[3], and brain science[4]. A relative consensus is that stronger hardware foundation, better algorithm design, and larger available data are the main driving forces for the further development of deep learning. Emerging deep physical neural networks[5] provide new ways of learning at the hardware level. Ingenious algorithms and model designs, such as GoogLeNet[6] and ResNet[7], have also been successful in advancing deep learning. In the meantime, how to obtain efficient data for deep learning is a key issue.

With the proliferation of mobile communication terminals and the rapid spread of sensory engineering, such as the Internet of Things, mankind has stepped into the era of big data. However, due to a series of barriers, such as personal privacy and industry secrecy, it is difficult to share data securely, and it is harder for deep learning to gain access to data on a mass of terminals. To this end, Google has proposed a federated learning[8] framework for mobile communication terminals, which adopts a distributed deep learning approach of local training and parameter aggregation. In federated learning, local training is performed at the participant side to update local model parameters, and parameter aggregation is conducted for the global model update by transmitting model parameters (e.g., gradients). However, many studies[9–12] in recent years have shown that federated learning also faces serious privacy threats, such as membership inference attacks, GAN-based attacks, and reconstruction attacks. Training data can be recovered with high quality from the model parameters shared in federated learning[11]. Protecting local model parameters transmitted in federated learning is a considerable problem[13].

Privacy-preserving strategies for federated learning are usually divided into two categories[14]: perturbation strategies based on differential privacy (DP), encryption strategies based on homomorphic encryption (HE), and secure multi-party computation (SMPC). DP-based preservation[15] inevitably reduces model utility, although privacy guarantees can be obtained. In addition, the applications[16, 17] of HE for high-dimensional gradients impose an unbearable computational load on the mobile terminal. It is worth noting that the general SMPC model using secret sharing fits well with the scenario of distributed learning and edge computing. The number of participants in secret sharing directly affects privacy-preserving intensity and system overhead but is solidified into two-party or all-party in previously proposed deep learning frameworks. The two-party computation (2-PC) setting is often used to implement a lightweight privacy-preserving deep learning framework[18, 19]. However, the security of the 2-PC setting is based on two noncolluding servers, and this

DOI: 10.52396/JUSTC-2022-0116
JUSTC, **2024**, 54(1): 0104

setting is usually achieved by using servers from different service providers[20], which is fragile and has no theoretical guarantee. All-party computation (All-PC) secret sharing-based secure aggregation is widely utilized in privacy-preserving federated learning[21–23]. However, a large number of participants in federated learning all join in the fully connected secret sharing computing, which will inevitably lead to huge computing and communication costs in the process of encrypting and handling dropout. 2-PC, All-PC, lightweight, and full connection are not secret sharing schemes that can better balance the strength and cost of privacy protection. Therefore, secret sharing-based privacy-preserving federated learning schemes with low communication-computation cost, sufficient privacy preservation, and robustness of participant dropouts should be sought.

In this work, we propose two efficient secure aggregation schemes for privacy-preserving federated learning: federated learning based on one-way secret sharing (FLOSS) and federated learning based on multi-shot secret sharing (FLMSS). The problem of constructing a global model where no participant reveals its updated gradients is referred to as secure aggregation for federated learning. We consider the general cross-device federated learning framework where mobile terminals have high communication constraints, weak computing power, and the possibility of dropouts at any time. FLOSS allows participants to join secret sharing of gradients within their own sharing group and reduce the communication cost by replacing entire high-dimensional random vectors with pseudorandom seeds. FLMSS takes advantage of the large number of cross-device federated learning participants in the real world to design secure aggregation protocols with tunable performance and the ability to solve participant dropout with minimal cost. It is worth mentioning that we abandon All-PC for the first time in FLMSS. The theoretical analysis proves that the local dataset of any participant can be well protected under the severe threat of the honest-but-curious server and participant collusion. Extensive experiments demonstrate that our proposed schemes have distinct advantages over the state-of-the-art works.

The remainder of the paper is organized as follows. Section 2 discusses the related work. In Section 3, we review some preliminaries used in our proposed schemes. In Section 4, problem statements are given. In Section 5, we elaborate on the design details of FLOSS and FLMSS. In Section 6, we present the security analysis followed by the experimental results in Section 7. Section 8 concludes this paper.

2 Related work

Our research is related to the line of work on federated learning with secure aggregation to protect the updated model parameters. To the best of our knowledge, none of the existing works can achieve our effectiveness between privacy-preserving intensity and additional overhead in a federated learning setting.

Some works[16, 17] have tried to prevent honest-but-curious servers from gaining user privacy by encrypting the uploaded model parameters with original homomorphic encryption. For the first time, Phong et al.[16] applied homomorphic encryption to distributed learning completely and provided systematic practice based on two different encryption methods: Paillier encryption and LWE (learning with errors)-based encryption. However, the only key pair is available to all participants, which makes the security of the homomorphic encryption system a huge risk. DeepPAR[17] is a privacy-preserving distributed learning scheme based on re-encryption in an attempt to address the aforesaid security vulnerabilities of the shared key. However, due to the unreasonable design of the system mechanism, trusted third parties (TTP) can easily be used to decrypt any user's encrypted data. Most importantly, the computational overhead of the above schemes for the original homomorphic encryption of high-dimensional parameters is unacceptable for ordinary devices.

Due to the low computational power of federated learning participants, many efforts have attempted to use more efficient encryption methods[24, 25] and hybrid methods[24, 26] with DP. The works of Xu et al.[24] and Wu et al.[25] both adopted multi-input function encryption (MIFE) to complete the encryption aggregation of local model parameters, which allows parties to encrypt their data and the sever to compute a specific function on the ciphertext. HybridAlpha[24] improves conventional MIFE schemes to reduce the number of communication rounds and running time without intensive pairing operations in federated learning and allows participants to dynamically join or drop. Wu et al.[25] employed the all-or-nothing transform (AONT) to encode parameter matrixes, thereby reducing the number of parameters required to be encrypted with MIFE. Even if efficient homomorphic encryption methods are adopted, performance overheads are not reduced to a desired level. DP-based protection schemes can achieve a mathematically guaranteed protection strength with a small additional computation overhead. Truex et al.[26] skillfully found that leveraging the secure multi-party computation (SMC) framework can reduce the noise required by DP and maintain the original protection strength for local model parameters. Therefore, the hybrid approach combines the Threshold-Paillier (TP) system and DP to achieve privacy protection of federated learning and reduce the noise added by DP. HybridAlpha[24] follows the above findings, but the difference is that it combines DP with MIFE. However, we must realize that once DP is introduced, it inevitably reduces the accuracy of the global model. Obviously, the lightweight privacy protection based on reducing the availability of the global model is not the optimal solution.

Based on the above considerations, additive secret sharing, which is a lightweight and efficient privacy protection method, is becoming popular. Bonawitz et al.[21] used a double-masking structure to protect the shared model parameters and secret sharing to enable the masks to be securely removed. However, the proposed double-masking mechanism to solve the participant dropout problem would reveal dropout participants' secret keys and introduce a significant amount of additional computational and communication costs. SSDDL[22] uses a simple secret sharing algorithm to achieve high-strength protection of uploaded gradients, but the unreasonable design of its secret content induces huge communication overhead, and this scheme cannot cope with the situation of

DOI: 10.52396/JUSTC-2022-0116
JUSTC, **2024**, 54(1): 0104

Table 1. Comparison of secure aggregation schemes for federated learning.

Schemes	Privacy-preserving technology			Resilience against dropout clients		
	Type*	Lightweight	Without TTP	One round†	Nesting†	Without rekeying‡
Phong et al.[16]	HE	×	×	√	√	√
Duan et al.[22]	SS	×	√	×	×	×
Truex et al.[26]	TP+DP	×	×	√	√	√
Xu et al.[24]	MIFE+DP	×	×	√	√	√
Wu et al.[25]	MIFE+AONT	×	×	√	√	√
Bonawitz et al.[21]	SS+AE	√	√	√	√	×
Zheng et al.[23]	SS+AE	√	√	√	×	√
Proposed FLOSS	SS+AE	√	√	×	×	×
Proposed FLMSS	SS+AE	√	√	√	√	√

* SS represents secret sharing; AE represents the key agreement protocol and the authenticated encryption scheme.

† One round means that the scheme can solve one round of participant dropout in one-round aggregation; Nesting represents the scheme that can deal with new participant dropout in the process of handling dropout.

‡ Without rekeying indicates that the process of handling dropout does not expose dropout clients' secret keys.

participant dropouts.

There are some follow-up works [23, 27, 28] attempting to improve Ref. [21] in many aspects, such as system overhead, robustness, and security. Turbo-aggregate[28] also employs the double-masking structure to protect model parameters based on additive secret sharing, but it sets up a topology of multi-group circular to aggregate the results, which reduces the aggregation overhead from $O(N^2)$ to $O(N\log N)$. Redundancy based on Lagrange coding is introduced to Turbo-aggregate[28], which enables the federated learning system to tolerate 50% dropouts. However, the structure of multi-group circles depends on additional topology information and the increase in system complexity, and there is still redundancy in computing and communication when aggregation is performed in groups. In fact, the aggregation overhead of participants can be reduced to $O(N)$[23]. We particularly note that Kadhe et al.[27] used the term multi-secret sharing, which is similar to the FLMSS. However, there are essential differences between the above two schemes. FASTSECAGG in Ref. [27] is based on the fast Fourier transform (FFT) to improve secret sharing, and its tolerance for client collusion and dropout is restricted to a preset constant fraction of clients. Zheng et al.[23] simplified the dropout handling mechanism of Ref. [21], introduced quantization-based model compression and trusted hardware into the system realization and obtained bandwidth efficiency optimization and security against an actively adversarial server. However, Zheng et al.[23] did not make innovative changes to the most basic additive secret sharing scheme originating from Ref. [21], and their tenuous dropout-tolerance algorithm could not converge to an end once any new dropout occurred in the process of dealing with client dropouts.

Compared with Refs. [21, 23], we have made significant changes in the mechanisms of secret-sharing objects and dropout processing, which further reduce the computation-communication overhead. Table 1 summarizes the comparison with most of the related works discussed above. The specific advantages of the computation-communication overhead of our work will be compared in the following sections.

3 Preliminaries

In this section, we introduce some preliminaries, including federated learning, secret sharing, key agreement, and pseudorandom generators. In Table 2, we provide a summary of the main notations used in this paper.

3.1 Federated learning

Federated learning[8] is a new distributed deep learning system with the goal of improving global model utility and protecting participant data privacy, which is constrained by the computational performance and communication bandwidth of edge devices. The basic framework of federated learning, which is composed of one sever and N participants, is shown in Fig. 1. The local training dataset $\mathcal{D}_i$ for federated learning is stored locally by each participant P_i, and a central server S collects and aggregates each local gradient $\boldsymbol{G}_{t,i}$ to generate an updated global model $\boldsymbol{W}_t$ at the tth communication round.

Table 2. Key notations used in this paper.

Notation	Description
$\boldsymbol{W}_t$	Aggregate model in round t
$\boldsymbol{G}_{t,i}$	Plaintext gradient generated from participant P_i in round t
$\boldsymbol{Y}_{t,i}$	Masked gradient uploaded by participant P_i in round t
$\mathcal{P}_N$	Set of selected N participants in each round
$\mathcal{P}_i$	Set of participants which have shared secrets with P_i in each round
$\mathcal{P}^{\text{drop}}$	Set of dropout participants in each round
$\mathcal{P}^{\text{help}}$	Set of participants which help deal with dropout in each round
(SK_u, PK_u)	Key pair of participant P_u in key agreement protocol
$AK_{u,v}$	Agreement key computed from SK_u and PK_v
$\boldsymbol{s}_{u,v}$	Mask computed from $AK_{u,v}$, i.e., secret sharing between P_u and P_v
d	Preset shot count of one participant in FLMSS

DOI: 10.52396/JUSTC-2022-0116
JUSTC, **2024**, 54(1): 0104

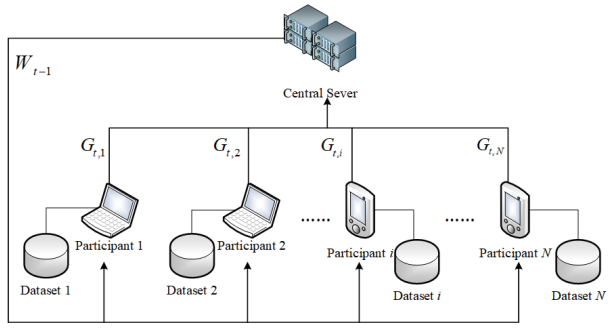


Fig. 1. The basic framework of federated learning.

At the tth communication round, the central sever selects a subset of N participants $\mathcal{P}_N$ and sends them the current global model $\boldsymbol{W}_{t-1}$. Each selected participant uses the current global model $\boldsymbol{W}_{t-1}$ and local dataset $\mathcal{D}_i$ to train to obtain the local gradient $\boldsymbol{G}_{t,i}$ and then uploads it back to the central server. The central sever averages the received gradients to compute the new global model:

$$\boldsymbol{W}_t = \boldsymbol{W}_{t-1} - \sum_{i=1}^{N} \alpha_{t,i} \boldsymbol{G}_{t,i}, \tag{1}$$

where $\alpha_{t,i} = |\mathcal{D}_i| \Big/ \sum_{i=1}^{N} |\mathcal{D}_i|$ is the weight corresponding to participant P_i at the tth communication round. A number of studies[29, 30] have improved the above benchmark federated learning framework and aggregation algorithms in terms of global model convergence speed and accuracy.

3.2 Secret sharing

This paper adopts Shamir's t-out-of-n secret sharing[31], which allows a user to split a secret s into n shares, such that any t shares can reconstruct s, but any $t-1$ or fewer shares cannot reveal anything about s. The scheme consists of the sharing algorithm and the reconstruction algorithm. The sharing algorithm $\mathrm{SS.share}(s,t,n) \to \{s_1, s_2, \cdots, s_n\}$ takes as input a secret s, the threshold t, and the number of shares $n \geqslant t$, and then generates a set of shares $\mathcal{S} = \{s_1, s_2, \cdots, s_n\}$. Given a subset $\mathcal{R}$ where $\mathcal{R} \subseteq \mathcal{S}$ and $|\mathcal{R}| \geqslant t$, the reconstruction algorithm $\mathrm{SS.recon}(\mathcal{R}, t) \to s$ can reconstruct the secret s.

3.3 Key agreement

Key agreement allows the communicating parties to exchange keys securely by negotiating a consistent key without revealing any information about this key. This paper adopts the Diffie–Hellman key agreement[32] to produce secret shares securely. Diffie–Hellman key agreement consists of parameter generation algorithms, key generation algorithms, and key agreement algorithms. Algorithm $\mathrm{KA.param}(k) \to pp = (\mathbb{G}', g, q)$ produces the needed public parameters, including prime order q and generator g of cyclic group $\mathbb{G}'$. Each party u can use the algorithm $\mathrm{KA.gen}(pp) \to (SK_u, PK_u) = (x, g^x)$ to generate its private-public key pair. x is sampled randomly from group $\mathbb{Z}_q$ (a cyclic group of prime order integers). Another party v can use the algorithm $\mathrm{KA.agree}(SK_v, PK_u) \to AK_{v,u} = g^{x_u x_v}$ to obtain a securely shared key between u and v. In the same way, only u can also obtain $AK_{u,v}$, where $AK_{u,v} = AK_{v,u}$. A Hash process can be added to the algorithm KA.agree to change the field of the shared key[21].

3.4 Pseudorandom generator

A secure pseudorandom generator[33, 34] PRG is a mapping-deterministic but unpredictable function that expands a binary bit string into a longer bit string. We can simplify the PRG function as $\mathrm{PRG}(\{0,1\}^{\lambda}) \to \{0,1\}^{p(\lambda)}$ where $p(\lambda) > \lambda$. The input of PRG is a seed, i.e., a bit string of length security parameter λ. Security for a pseudorandom generator guarantees that it is negligibly possible to computationally distinguish the output of PRG from a truly random one, as long as the seed is confidential. It is worth emphasizing that when feeding the same seeds to the same pseudorandom generator, the output is the same, which is one of the important methods used to save communication overhead.

4 Problem statements

In this section, we introduce the threat model, system models, and requirements of FLOSS and FLMSS.

4.1 Threat model

Our system is designed to withstand two potential adversaries: the central sever and the participants.

(Ⅰ) Honest-but-curious central sever. The honest-but-curious or semi-honest adversarial model is commonly considered in privacy-preserving data mining and cloud computing[35, 36]. The honest-but-curious central sever follows the defined protocol correctly and will not intentionally modify, add or delete data required to be processed. However, the sever will try to learn additional information or understand the content of data. Therefore, participants' private information may be learned and leaked from the model updates by the sever.

(Ⅱ) Curious and colluding participants. We assume that some participants may collude with the central server to acquire the privacy of other participants by inspecting the information that they have access to during the correct execution of the protocol. The unrealistic extreme case in which there is only one honest participant and all other participants collude with the sever is not considered. This is a common assumption in secret sharing-based privacy-preserving schemes[22, 23].

We assume that dependable and secure channels are guaranteed in all communications. Thus, messages are prevented from accidental attacks such as snooping, loss, and tampering during transmission.

4.2 System models of FLOSS and FLMSS

First, we define FLOSS and FLMSS in a functional way with a comprehensive textual explanation.

Definition 1: we formalize the definition of FLOSS = (Init, groupSet, LocalTrain, GradShare, ModelUpdate) which consists of five algorithms.

(ⅰ) $\mathrm{Init}(\mathcal{P}_N, m) \to (\boldsymbol{W}, \boldsymbol{\alpha}, \boldsymbol{M}, \boldsymbol{K})$: The central server selects $\mathcal{P}_N$ to participate in this round of training, the participation weights form a vector $\boldsymbol{\alpha}$, the latest global model is used as the initial model $\boldsymbol{W}$ for this round, and N participants are grouped according to the sharing-group size m to obtain the sharing-group set M. K is the set of all participants' key pairs.

DOI: 10.52396/JUSTC-2022-0116
JUSTC, **2024**, 54(1): 0104

(ii) GroupSet($\boldsymbol{M}$) $\rightarrow$ (μ,$\boldsymbol{idx}_\mu$): On input the sharing-group set $\boldsymbol{M}$, the server selects a group μ and generates participants' IDs $\boldsymbol{idx}_\mu$ in this group.

(iii) LocalTrain($\boldsymbol{W}$,μ,η) $\rightarrow$ $\boldsymbol{G}_\mu$: Participant P_i in group μ uses local dataset $\mathcal{D}_i$ and global model $\boldsymbol{W}$ to run the deep learning algorithm with parameter selection rate η and obtains local gradient $\boldsymbol{G}_i$ where $\boldsymbol{G}_\mu = \{\boldsymbol{G}_i\}_{i\in\mu}$.

(iv) GradShare($\boldsymbol{G}_\mu$,$\boldsymbol{K}$) $\rightarrow$ $\boldsymbol{Y}_\mu$: On input the set of local models $\boldsymbol{G}_\mu$ and the set of key pairs $\boldsymbol{K}$, participants obtain the set of secret gradients $\boldsymbol{Y}_\mu$ by secret sharing in group μ and then upload these secret gradients.

(v) ModelUpdate($\boldsymbol{Y}_{\mu_1}$,$\boldsymbol{Y}_{\mu_2}$,$\cdots$) $\rightarrow$ $\boldsymbol{W}$: After receiving the secret gradients uploaded by all sharing groups, the server performs a global model update to obtain a new global model $\boldsymbol{W}$.

Definition 2: We formalize the definition of FLMSS = (Init, LocalTrain, GradShare, DropHandle, ModelUpdate) which consists of five algorithms.

(i) Init($\mathcal{P}_N$) $\rightarrow$ ($\boldsymbol{W}$,$\boldsymbol{\alpha}$,$\boldsymbol{K}$): FLMSS no longer needs to group participants in the initialization phase, and the rest is the same as definition 1.

(ii) LocalTrain($\boldsymbol{W}$,$\mathcal{P}_N$,η,$\boldsymbol{\alpha}$) $\rightarrow$ ($\boldsymbol{G}_1$,$\boldsymbol{G}_2$,$\cdots\boldsymbol{G}_N$): Participant P_i obtains the local gradient $\boldsymbol{G}_i$ in the same way as in definition 1.

(iii) GradShare($\boldsymbol{G}_1$,$\cdots$,$\boldsymbol{G}_N$,$\boldsymbol{K}$,d) $\rightarrow$ ($\boldsymbol{Y}_1$,$\boldsymbol{Y}_2$,$\cdots\boldsymbol{Y}_N$): On input the shot count d of secret sharing and the set of key pairs $\boldsymbol{K}$, each participant splits its plaintext gradient into $d+1$ parts by generating d mask vectors computed from agreement keys and secretly sharing them with d participants. The plaintext gradient $\boldsymbol{G}_i$ of participant P_i becomes the secret gradient $\boldsymbol{Y}_i$ after secret sharing is completed.

(iv) DropHandle$\left(\mathcal{P}^{\text{drop}},\ \{\boldsymbol{Y}_i\}_{P_i\in\mathcal{P}^{\text{help}}},\ \boldsymbol{K}\right) \rightarrow \left(\{\boldsymbol{Y}_i\}_{P_i\in\mathcal{P}^{\text{help}}}\right)$: On input of the set of dropped participants $\mathcal{P}^{\text{drop}}$ and the set of key pairs $\boldsymbol{K}$, the participant $\mathcal{P}^{\text{help}}$, which shares secrets with the dropped participants, cooperates to resolve dropout and upload new secret gradients.

(v) ModelUpdate($\{\boldsymbol{Y}_i\}_{P_i\in\mathcal{P}_N-\mathcal{P}^{\text{drop}}}$) $\rightarrow$ $\boldsymbol{W}$: After receiving all the secret gradients without unresolved dropped participants, the server performs a global model update to obtain a new global model $\boldsymbol{W}$.

Figs. 2 and 3 show the system models of FLOSS and FLMSS. The yellow dashed boxes mark the sequence of the main steps of secret sharing. Key notations in both figures can be explained in Table 2. In Fig. 2, STEP 1, STEP 2, and STEP 3 correspond to LocalTrain, GradShare, and ModelUpdate in the definition of FLOSS, respectively. In Fig. 3, STEP 1 and STEP 2 correspond to LocalTrain and GradShare in the definition of FLMSS. It is worth noting that we need to understand STEP 3 and STEP 4 together because there are situations where new participants drop out when dealing with dropouts. This means that STEP 3 and STEP 4 may be alternating and nested. The combination of STEP 3 and STEP 4 in Fig. 3 corresponds to the process of DropHandle with ModelUpdate.

4.3 Requirements

To guarantee the performance of the proposed methods, we first define the following four requirements.

Security. The security guaranteed by the proposed schemes is based on the threat model described in Section 4.1. The local dataset and plaintext gradients computed from it should be kept secret against both the central sever and the other participants. No adversary can derive private information from any messages communicated in the federated learning system that would reveal the participants' local data and models.

Efficiency. The extra computation and communication cost due to privacy-preserving schemes should be as small as possible, especially the cost borne by the participants.

Accuracy. Privacy-preserving schemes should minimize negative effects on global model accuracy.

Robustness. The system should be able to have an efficient solution to the problem of participant dropout at any point in the federated learning process.

5 Proposed schemes

In this section, we give a detailed construction of FLOSS and

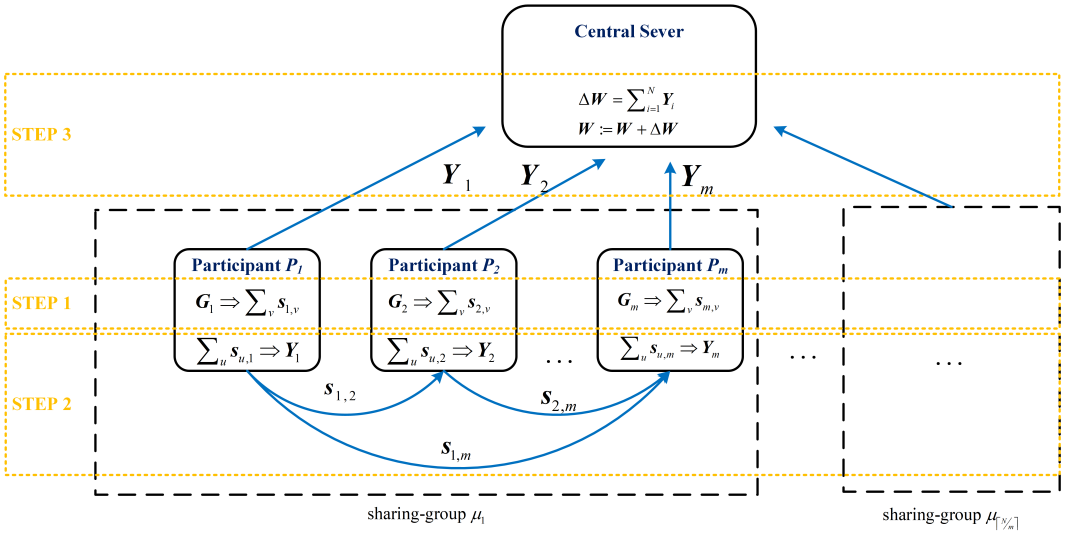


Fig. 2. System model of FLOSS where the formulas are simplified versions.

DOI: 10.52396/JUSTC-2022-0116
JUSTC, **2024**, 54(1): 0104

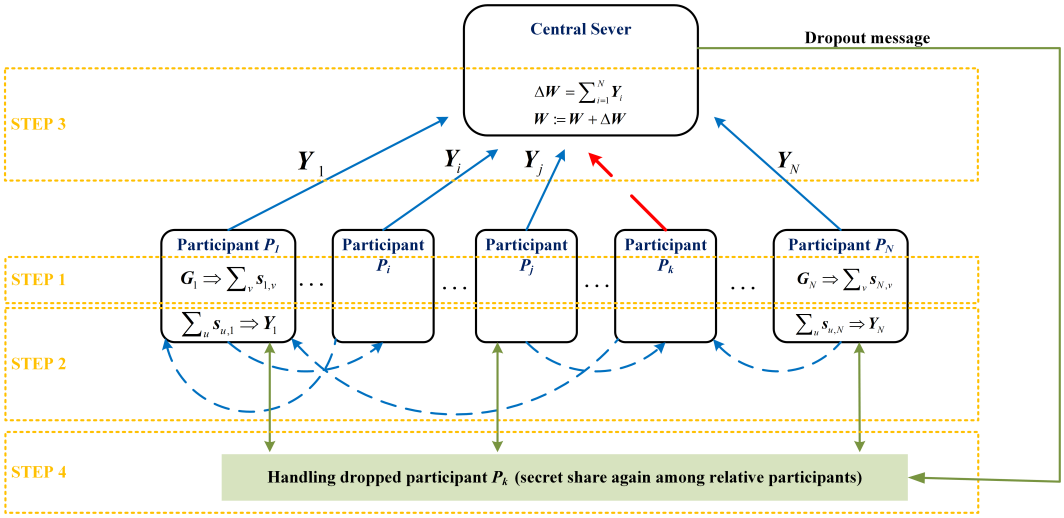


Fig. 3. System model of FLMSS with the case of $d = 1$. The omitted secret share values can be obtained in Fig. 2. The aggregation process after handling the first round of dropped participants is not shown in this figure. The red dashed arrow represents the dropout state of the participant.

FLMSS.

5.1 Federated learning based on one-way secret sharing

Compared with redundant two-way secret sharing[22], the FLOSS scheme mainly performs one-way secret sharing of each participant's plaintext gradients after local training is completed. According to definition 1, the five constituent algorithms of FLOSS are elaborated next.

5.1.1 Init$(\mathcal{P}_N, m) \to (\boldsymbol{W}, \alpha, M, K)$

At the beginning of each round of training, the central server first randomly selects N participants from all participants to participate in this round of training, i.e., $\mathcal{P}_N = \{P_1, P_2, \cdots, P_N\}$. To flexibly control the scope of secret sharing and the tradeoff between privacy protection strength and communication cost, we divide N participants into $\lceil \frac{N}{m} \rceil$ sharing groups of size m. μ_i represents the sharing group with the index i. Therefore, the group set is denoted $\boldsymbol{M} = \{\mu_1, \cdots, \mu_{\lceil N/m \rceil}\}$. The server calculates the participation weights α, which are explained in Section 3.1. Given the security parameter k, each participant P_u needs to generate its own key pair (SK_u, PK_u) through KA.param(k) and KA.gen(pp). Then, participants send their public keys to the central sever. Finally, the central server broadcasts the global model $\boldsymbol{W}$ and the public key set $\{PK_i\}_{P_i \in \mu}$ to all participants and group μ.

5.1.2 GroupSet$(\boldsymbol{M}) \to (\mu, \boldsymbol{idx}_\mu)$

After dividing the N participants into groups, the central server distinguishes these $\lceil \frac{N}{m} \rceil$ groups by numbering them from 1 to $\lceil \frac{N}{m} \rceil$ and then randomly numbers each member of all shared groups from 1 to m, i.e., $\boldsymbol{idx}_\mu = \{idx_1, \cdots, idx_m\} = \{1, \cdots, m\}$. Each participant can be uniquely identified by (μ, idx). The output of GroupSet($\boldsymbol{M}$) is the result of setting the group information for one shared group. The above elaboration ignores one group that may have fewer than m participants, which is nonessential.

5.1.3 LocalTrain$(\boldsymbol{W}, \mu, \eta, \alpha) \to \boldsymbol{G}_\mu$

Without loss of generality, consider a participant P_i in a shared group μ as an example. P_i receives the global model $\boldsymbol{W}$ sent by the central server, starts training on the local dataset $\mathcal{D}_i$ and obtains the local gradient $\boldsymbol{G}_i$ after replacing the local model with $\boldsymbol{W}$.

To further reduce communication overhead, many schemes[37–39] sample gradients or perform higher-level compression, thus degrading model utility for a significant reduction in communication data size of gradient exchange. Taking the distributed selective stochastic gradient descent (DSSGD)[38] as an example, participant P_i chooses a fraction η of $\boldsymbol{G}_i$ to be shared rather than the entire $\boldsymbol{G}_i$. The contribution of each parameter of the ultrahigh-dimensional gradient to SGD varies greatly, and gradient descent is a dynamic and continuous process. Therefore, the selective abandonment of some parameters will not have a great impact on the usability of the final model. The two common parameter selection methods are random sampling and TOPK selection, where TOPK is the first K values in descending order. In recent years, more advanced gradient compression methods, such as Refs. [37, 39], can compress gradients more than 300 times with only negligible loss of accuracy by error compensation techniques. The focus of this paper is not on the gradient compression method but on making privacy-preserving communication more effective by improving the secret sharing protocol. Finally, participant P_i multiplies the compressed gradient by the participation weight α_i in advance.

5.1.4 GradShare$(\boldsymbol{G}_\mu, \boldsymbol{K}) \to \boldsymbol{Y}_\mu$

Without loss of generality, take a shared group μ and one of its members $P_{(\mu,k)}$ whose number is $idx = k$ as an example. Next, m-out-of-m additive secret sharing of gradients needs to be performed within the shared group μ, i.e., each group member splits its local gradient into several shares as required and keeps one of them local. Other shares are sent to designated group members for preservation. It is worth noting that the secret sharing method used in this paper only needs the sharing algorithm instead of the reconstruction algorithm.

Specifically, the group member $\boldsymbol{P}_{(\mu,k)}$ divides $\boldsymbol{G}_{(\mu,k)}$ into k parts, one of which is saved by himself, and the other $k-1$

DOI: 10.52396/JUSTC-2022-0116
JUSTC, **2024**, 54(1): 0104

parts are sent to the group members $\{P_{(\mu,1)},\cdots,P_{(\mu,k-1)}\}$ with a smaller idx than himself to be saved. Therefore, group member $P_{(\mu,m)}$ needs to send shares to all other group members without receiving shares. Meanwhile, the group member $P_{(\mu,1)}$ only needs to receive the shares.

How $P_{(\mu,k)}$ can safely split local gradients and send shares to designated members is described as follows. The main methods are key agreement and pseudorandom number generation. First, in the initialization phase, $P_{(\mu,k)}$ receives the set of required public keys $\{PK_i\}_{P_i\in\mu}$ from the server. Without considering the group number μ, P_k needs to split the gradient $\boldsymbol{G}_k$ into shares $\{\boldsymbol{s}_{k,k},\cdots,\boldsymbol{s}_{k,1}\}$ where $k\geqslant 2$. The secret share is calculated as:

$$\boldsymbol{s}_{k,j}=\mathrm{PRG}(\mathrm{KA.agree}(SK_k,PK_j)),\ j<k, \tag{2}$$

$$\boldsymbol{s}_{k,k}=\boldsymbol{G}_k-\sum_{j=1}^{k-1}\boldsymbol{s}_{k,j}. \tag{3}$$

P_k uses the key agreement algorithm to generate private shared keys for each shared object and uses these keys as seeds to the pseudorandom generator to obtain secret vectors with the same dimension as the gradient. The participants use the same pseudorandom generator. The specific operation of the PRG function is not the focus of this paper and can be found in existing work[21, 22]. After splitting the gradient, P_k needs to send the secret shares one by one to the corresponding participants. It is worth noting that P_k does not need to actually perform the sending operation, and other participants can obtain the secret shares from P_k by themselves according to the key agreement algorithm. For example, P_j can obtain $\boldsymbol{s}_{k,j}$ where $\boldsymbol{s}_{k,j}=\boldsymbol{s}_{j,k}$ by

$$\boldsymbol{s}_{k,j}=\mathrm{PRG}(\mathrm{KA.agree}(SK_j,PK_k)). \tag{4}$$

After all members of the group complete the abovementioned secret sharing, each participant computes its aggregation result as:

$$\boldsymbol{Y}_k=\sum_{j=k}^{m}\boldsymbol{s}_{j,k}. \tag{5}$$

Finally, P_k sends the result $\boldsymbol{Y}_k$ to the central sever for model updating.

5.1.5 ModelUpdate$(\boldsymbol{Y}_{\mu_1},\boldsymbol{Y}_{\mu_2},\cdots)\to\boldsymbol{W}$

Without loss of generality, the case of group μ is discussed. Upon receiving all the results $\{\boldsymbol{Y}_i\}_{P_i\in\mu}$, the central sever updates the global parameters by

$$\boldsymbol{W}:=\boldsymbol{W}+\Delta\boldsymbol{W}, \tag{6}$$

where

$$\Delta\boldsymbol{W}=\sum_{i=1}^{m}\boldsymbol{Y}_i. \tag{7}$$

Combining Eqs. (3) and (5), we can obtain

$$\Delta\boldsymbol{W}=\sum_{i=1}^{m}\boldsymbol{Y}_i=\sum_{i=1}^{m}\sum_{j=1}^{i}\boldsymbol{s}_{i,j}=\sum_{i=1}^{m}\boldsymbol{G}_i. \tag{8}$$

Algorithm 1: FLMSS

Initialization:
for each participant P_i **do**
  $SK_i=x_i\in\mathbb{Z}_q$
  $PK_i=g^{x_i}\in\mathbb{G}'$
  Upload PK_i to the central sever
end for
Initialize $\boldsymbol{W}_0$
Sever executes:
for each round $t=1,2,\cdots,$ **do**
  $\mathcal{P}_N\leftarrow$ (random set of m participants)
  for each participant P_k in $\mathcal{P}_N$ **in parallel do**
    $\mathcal{P}_{t,k}^{\mathrm{send}}\leftarrow$(randomly sample d participants from $\mathcal{P}_N$)
    Upload $\mathcal{P}_{t,k}^{\mathrm{send}}$ to the central sever.
  end for
  $\mathcal{P}_{t,1}^{\mathrm{acce}},\mathcal{P}_{t,2}^{\mathrm{acce}},\cdots,\mathcal{P}_{t,N}^{\mathrm{acce}}\leftarrow\mathcal{P}_{t,1}^{\mathrm{send}},\mathcal{P}_{t,2}^{\mathrm{send}},\cdots,\mathcal{P}_{t,N}^{\mathrm{send}}$
  for each participant P_k in $\mathcal{P}_N$ **in parallel do**
    $\boldsymbol{w}_{t,k}\leftarrow\mathrm{ClientUpdate}(\boldsymbol{W}_{t-1},\mathcal{P}_{t,k}^{\mathrm{acce}})$
  end for
  $\boldsymbol{W}_t\leftarrow\frac{1}{|\mathcal{D}|}\cdot\sum_{k=1}^{N}\boldsymbol{w}_{t,k}$ //$|\mathcal{D}|=\sum_{k=1}^{N}|\mathcal{D}_k|$
end for
ClientUpdate($W_{t-1},\mathcal{P}_{t,k}^{\mathrm{acce}}$**):** // Run on participant P_k
$\mathcal{P}_{t,k}\leftarrow\mathcal{P}_{t,k}^{\mathrm{send}}\cup\mathcal{P}_{t,k}^{\mathrm{acce}}$
$\mathcal{B}\leftarrow$(split $\mathcal{D}_k$ into batches of size B)
for each local epoch e from 1 to E **do**
  for batch $b\in\mathcal{B}$ **do**
    $\boldsymbol{w}\leftarrow\boldsymbol{w}-\eta\cdot\nabla\ell(\boldsymbol{W}_{t-1};b)$
  end for
end for
for each key PK_j where $P_j\in\mathcal{P}_{t,k}$ **do**
  $\boldsymbol{s}_{k,j}\leftarrow PRG\left(\mathrm{KA.agree}\left(SK_k,PK_j\right)\right)$
end for
$\boldsymbol{w}_{t,k}\leftarrow|\mathcal{D}_k|\cdot\boldsymbol{w}-\sum_{P_u\in\mathcal{P}_{t,k}^{\mathrm{send}}}\boldsymbol{s}_{k,u}+\sum_{P_v\in\mathcal{P}_{t,k}^{\mathrm{acce}}}\boldsymbol{s}_{k,v}$
Return $\boldsymbol{w}_{t,k}$ to the central sever

Therefore, the result obtained by central server aggregation is the result obtained by initial gradient aggregation without secret sharing. In parallel, the server is able to obtain a new global model after performing the same security aggregation process with the other sharing groups.

5.2 Federated learning based on multi-shot secret sharing

Many processes and implementation methods of FLMSS are common to FLOSS, so this section mainly describes the differences. To avoid confusion between the FLMSS and FLOSS, we first point out the essential differences between them. FLOSS performs m-out-of-m fully connected secret sharing in each sharing group of m size, i.e., the group members will share secrets with each other. However, FLMSS

DOI: 10.52396/JUSTC-2022-0116
JUSTC, 2024, 54(1): 0104

does not need to group participants, which is easy to intentionally set by the colluding parties, and secret sharing is carried out randomly by all participants. In essence, d-shot FLMSS means that each participant performs d'-out-of-d' secret sharing where $d' \geqslant d+1$ among all participants. According to definition 2, the five constituent algorithms of the FLMSS are elaborated next. The overall FLMSS scheme is presented in Algorithm 1. Meanwhile, the protocol for dealing with dropout is shown separately in Algorithm 2.

5.2.1 Init$(\mathcal{P}_N) \to (\boldsymbol{W}, \boldsymbol{\alpha}, \boldsymbol{K})$

During the initialization phase, the central server no longer needs to group participants. After generating their own key pairs, participants send their public keys to the central sever. The central server needs to broadcast the identity information of all participants, which occupies very small communication costs to all participants.

5.2.2 LocalTrain$(\boldsymbol{W}, \mathcal{P}_N, \eta, \boldsymbol{\alpha}) \to (\boldsymbol{G}_1, \boldsymbol{G}_2, \cdots, \boldsymbol{G}_N)$

Participant P_i obtains the local gradient $\boldsymbol{G}_i$ through training on model $\boldsymbol{W}$, dataset $\mathcal{D}_i$, and compression rate η, and the specific method is the same as that of FLOSS.

5.2.3 GradShare$(\boldsymbol{G}_1, \cdots, \boldsymbol{G}_N, K, d) \to (\boldsymbol{Y}_1, \boldsymbol{Y}_2, \cdots, \boldsymbol{Y}_N)$

Without loss of generality, consider a participant P_i as an example. According to the participant information received from the sever in this round, P_i randomly selects d participants as its secret sharing destination. We can express it as $\mathcal{P}_i^{\text{send}} \overset{R}{\leftarrow} \mathcal{P}_{N-1}$, where $\left|\mathcal{P}_i^{\text{send}}\right| = d$. Specifically, P_i first sends the identities of participants in $\mathcal{P}_i^{\text{send}}$ as a message to the central sever. Assuming $P_j \in \mathcal{P}_i^{\text{send}}$, the central sever verifies that P_j is online and then sends the public key PK_j and PK_i to P_i and P_j. It should be emphasized that the central server is required to record each pair of participants, which confirms the intent for secret sharing. When the pair of participants completes the public key exchange, P_i executes:

$$\boldsymbol{G}_i := \boldsymbol{G}_i - \text{PRG}(\text{KA.agree}(SK_i, PK_j)). \tag{9}$$

Correspondingly, P_j executes:

$$\boldsymbol{G}_j := \boldsymbol{G}_j + \text{PRG}(\text{KA.agree}(SK_j, PK_i)). \tag{10}$$

When P_i finds d destinations and completes secret sharing d times, its active sharing ends. Under this mechanism, each participant needs to actively share secrets with d participants and accept the sharing from random participants $\mathcal{P}_i^{\text{acce}}$. After completing the above sharing, P_i updates the local aggregation result $\boldsymbol{Y}_i$:

$$\boldsymbol{Y}_i = G_i - \sum_{P_u \in \mathcal{P}_i^{\text{send}}} \boldsymbol{s}_{i,u} + \sum_{P_v \in \mathcal{P}_i^{\text{acce}}} \boldsymbol{s}_{v,i}. \tag{11}$$

Finally, the central server has recorded the collaborative sharing participant set $\mathcal{P}_i$ of each participant P_i, where $\mathcal{P}_i = \mathcal{P}_i^{\text{send}} \cup \mathcal{P}_i^{\text{acce}}$.

5.2.4 DropHandle$\left(P_{\text{drop}}, \{Y_i\}_{P_i \in \mathcal{P}^{\text{help}}}, K\right) \to \left(\{Y_i\}_{P_i \in \mathcal{P}^{\text{help}}}\right)$

Participant dropout before completion of secret sharing has a negligible impact on the global model. However, if the participant is disconnected after completing the secret sharing, the server aggregation result will lose some random shares, which greatly reduces the availability of the aggregated gradient.

Algorithm 2 DropHandle for FLMSS

w_k: Model parameters uploaded by P_k before handling dropout

Initialization:

  for round $i = 2, 3, \cdots$, **do**

    $\mathcal{P}_{i\text{th}}^{\text{drop}} \leftarrow \emptyset$, $\mathcal{P}_{i\text{th}}^{\text{help}} \leftarrow \emptyset$

  end for

Sever computes $\mathcal{P}_{1\text{st}}^{\text{help}} \leftarrow \mathcal{P}_1^{\text{send}}, \mathcal{P}_2^{\text{send}}, \cdots, \mathcal{P}_N^{\text{send}}, \mathcal{P}_{1\text{st}}^{\text{drop}}$

$i \leftarrow 1$

while $\left|\mathcal{P}_{i\text{th}}^{\text{drop}}\right| \neq 0$ **do**

  Sever updates $\mathcal{P}_{i\text{th}}^{\text{help}} \leftarrow \mathcal{P}_1^{\text{send}}, \mathcal{P}_2^{\text{send}}, \cdots, \mathcal{P}_N^{\text{send}}, \mathcal{P}_{\text{history}}^{\text{drop}}, \mathcal{P}_{i\text{th}}^{\text{drop}}$

  Sever sends $\mathcal{P}_{i\text{th}}^{\text{drop}}$ and $\mathcal{P}_{i\text{th}}^{\text{help}}$ to the participants in $\mathcal{P}_{i\text{th}}^{\text{help}}$

  for each participant P_k in $\mathcal{P}_{i\text{th}}^{\text{help}}$ **in parallel do**

    if ($\mathcal{P}_k \subseteq \mathcal{P}_{\text{history}}^{\text{drop}}$ and $\left|\mathcal{P}_{i\text{th}}^{\text{help}}\right| = 1$) or ($P_k$ drops out) **do**

      $\mathcal{P}_{(i+1)\text{th}}^{\text{drop}} \leftarrow \mathcal{P}_{(i+1)\text{th}}^{\text{drop}} \cup \{P_k\}$

    else

      Initialize a vector $\boldsymbol{g}_k$ of size $|\boldsymbol{w}_k|$

      $\boldsymbol{g}_k \leftarrow \boldsymbol{w}_k + \sum_{P_u \in \mathcal{P}_k^{\text{send}} \cap \mathcal{P}_{\text{history}}^{\text{drop}}} \boldsymbol{s}_{k,u} - \sum_{P_v \in \mathcal{P}_k^{\text{acce}} \cap \mathcal{P}_{\text{history}}^{\text{drop}}} \boldsymbol{s}_{k,v}$

      Select $d_{i\text{th}}^{(k)} \in \{0, 1\}$ participant P_d randomly to share secret

      Update $\boldsymbol{g}_k$ according to secret sharing in Algorithm 1

      Send g_k to the central sever

      $\mathcal{P}_k^{\text{send}} \leftarrow \mathcal{P}_k^{\text{send}} \cup \{P_d\}$, update $\mathcal{P}_k^{\text{acce}}$ accordingly

    end if

  end for

  $i \leftarrow i + 1$

end while

Sever computes $\boldsymbol{W} \leftarrow \sum_{P_k \in \mathcal{P}_N - \mathcal{P}_{\text{history}}^{\text{drop}} - \mathcal{P}_{\text{history}}^{\text{help}}} \boldsymbol{w}_k + \sum_{P_k \in \mathcal{P}_{\text{history}}^{\text{help}}} \boldsymbol{g}_k$

The set of newly dropped participants in the ith processing round is denoted $\mathcal{P}_{i\text{th}}^{\text{drop}}$. $\mathcal{P}_{i\text{th}}^{\text{drop}}$ can be obtained by the central server in the process of collecting local aggregation results. Correspondingly, the central server can obtain the set $\mathcal{P}_{i\text{th}}^{\text{help}}$, which shares secrets with the dropped participants $\mathcal{P}_{i\text{th}}^{\text{drop}}$ and is alive from the record.

In the first round of processing dropped participants, the server obtains the identities of the online participants $\mathcal{P}_{1\text{st}}^{\text{help}}$, which is required to offer help in this round from $\{\mathcal{P}_i\}_{P_i \in \mathcal{P}_{1\text{st}}^{\text{drop}}}$. After determining the set of dropped participants, the central server is required by the protocol to no longer accept local results, which may be late from participants in $\mathcal{P}_{i\text{th}}^{\text{drop}}$. The server then sends $\mathcal{P}_{1\text{st}}^{\text{drop}}$ and $\mathcal{P}_{1\text{st}}^{\text{help}}$ to each participant in $\mathcal{P}_{1\text{st}}^{\text{help}}$. Without loss of generality, suppose $P_a \in \mathcal{P}_{1\text{st}}^{\text{help}}$. After P_a receives the message, P_a first strips out all shares that are

DOI: 10.52396/JUSTC-2022-0116
JUSTC, 2024, 54(1): 0104

shared with the dropped participants $\mathcal{P}_{1\text{st}}^{\text{drop}}$ in $\boldsymbol{Y}_a$:

$$\boldsymbol{Y}_a := \boldsymbol{Y}_a + \sum_{P_i \in \mathcal{P}_{1\text{st}}^{\text{drop}} \cap \mathcal{P}_a^{\text{ssend}}} \boldsymbol{s}_{a,i} - \sum_{P_j \in \mathcal{P}_{1\text{st}}^{\text{drop}} \cap \mathcal{P}_a^{\text{acce}}} \boldsymbol{s}_{j,a}. \tag{12}$$

Then, in the first dropout-handling round, P_a performs $d_{1\text{st}}^{(a)}$-shot secret sharing for $\boldsymbol{Y}_a$ among participants $\mathcal{P}_{1\text{st}}^{\text{help}}$. For P_a and $\mathcal{P}_a$ in dropout-handling round i, we have

$$d_{i\text{th}}^{(a)} = \begin{cases} 0, & \mathcal{P}_a \not\subset \mathcal{P}_{\text{history}}^{\text{drop}}; \\ \min\{1, \left|\mathcal{P}_{i\text{th}}^{\text{help}}\right| - 1\}, & \mathcal{P}_a \subseteq \mathcal{P}_{\text{history}}^{\text{drop}}; \end{cases} \tag{13}$$

where we define $\mathcal{P}_{\text{history}}^{\text{drop}}$ as $\mathcal{P}_{1\text{st}}^{\text{drop}} \cup \mathcal{P}_{2\text{nd}}^{\text{drop}} \cup \cdots \cup \mathcal{P}_{i\text{th}}^{\text{drop}}$. Moreover, $\mathcal{P}_{\text{history}}^{\text{help}}$ can be similarly defined in Algorithm 2. When all participants in $\mathcal{P}_{1\text{st}}^{\text{help}}$ have completed the stripping and secret sharing operations, the new $\{\boldsymbol{Y}_a\}_{P_a \in \mathcal{P}_{1\text{st}}^{\text{help}}}$ are generated to be uploaded.

After the central server receives newly uploaded aggregation results for all participants in $\mathcal{P}_{1\text{st}}^{\text{help}}$, it replaces the corresponding original values with $\{\boldsymbol{Y}_a\}_{P_a \in \mathcal{P}_{1\text{st}}^{\text{help}}}$, and the task of processing dropped participants ends. If there are new dropped participants while FLMSS processes the first round of dropout, the system goes to the second round of processing. In the same way, the server uses $\mathcal{P}_{2\text{nd}}^{\text{drop}}$ and $\mathcal{P}_{2\text{nd}}^{\text{help}}$ to organize the same process as the previous round among $\mathcal{P}_{2\text{nd}}^{\text{help}}$. Iterate the above process until $\left|\mathcal{P}_{i\text{th}}^{\text{drop}}\right| = 0$. Now, it is necessary to explain that $\left|\mathcal{P}_{i\text{th}}^{\text{drop}}\right|$ easily converges to 0 as i increases in the FLMSS. An objective fact is that the participants that successfully upload the gradient parameters in a round are likely to be able to stay online and provide reliable assistance[23]. This is because participants who successfully upload the gradient are considered to have sufficient conditions to participate in the federated learning process during this time period, such as being charged, idle and on a favorable network[40, 41], which are not easily changed in a short period of time. As a result, the rate of participant dropout will decline rapidly and substantially in the process of dealing with dropout. On the other hand, it is also crucial that FLMSS is not a fully connected secret sharing scheme. When one participant drops out, only a small number of relevant other participants are required for assistance. In past works, when one participant dropped out, a large number[21, 28] or even all other participants[22, 23] were required to participate in recovering masks.

Consider below one extreme situation that can cause privacy leakage when dealing with dropped participants. When $\left|\mathcal{P}_{i\text{th}}^{\text{help}}\right| = 1$ and P_a have only shared secrets with the current round of dropped users, i.e., $\mathcal{P}_a \subseteq \mathcal{P}_{i\text{th}}^{\text{drop}}$, P_a will be left with only plaintext $\boldsymbol{Y}_a$ where $\boldsymbol{Y}_a = \boldsymbol{G}_a$ after the stripping operation is completed. When this extremely unlikely situation occurs, P_a takes the initiative to announce the drop to the server to protect its privacy. FLMSS then proceeds to the next round of drop processing until this very event with small probability does not happen again.

5.2.5 ModelUpdate$(\{\boldsymbol{Y}_i\}_{P_i \in \mathcal{P}_N - \mathcal{P}^{\text{drop}}}) \to \boldsymbol{W}$

When no more participants drop out, the server uses the latest local results uploaded by online participants for secure aggregation in the same way as FLOSS.

6 Security analysis

In this section, we present the security analysis of FLOSS and FLMSS. The following proofs are based on an existing lemma[21] that if participants' values have uniformly random masks that are added to them, then the resulting values look uniformly random. Theorem 3 analyzes the security from a probabilistic point of view. Since the probability and circumstances of participant dropouts are difficult to estimate and FLMSS provides targeted protection against possible privacy leakage, the impact of the dropout handling process on the probability is ignored.

Theorem 1. In FLOSS, the local dataset privacy of P_k cannot be guaranteed if and only if the honest-but-curious central sever and all $(m-1)$ other participants in the shared group collude. Otherwise, the privacy of the local dataset $\mathcal{D}_k$ can be guaranteed.

Proof: The original local gradient $\boldsymbol{G}_k$ of P_k is split into $\{\boldsymbol{s}_{k,1}, \boldsymbol{s}_{k,2}, \cdots, \boldsymbol{s}_{k,k}\}$. To recover $\boldsymbol{G}_k$, the adversary must obtain these k shares or the sum of these k shares. Since in the whole communication process of the FLOSS, there is no separate addition between these k shares, each value of these k shares must be obtained to restore $\boldsymbol{G}_k$. Except for P_k itself, only $\{P_1, P_2, \cdots, P_{k-1}\}$ know $\{\boldsymbol{s}_{k,1}, \boldsymbol{s}_{k,2}, \cdots, \boldsymbol{s}_{k,k-1}\}$. Therefore, the adversary needs the collusion of these $(k-1)$ participants. Furthermore, the only way for an adversary to obtain the local share value $\boldsymbol{s}_{k,k}$ is $\boldsymbol{s}_{k,k} = \boldsymbol{Y}_k - \boldsymbol{s}_{k+1,k} - \boldsymbol{s}_{k+1,k} - \cdots - \boldsymbol{s}_{m,k}$. Except for P_k itself, only $\{P_{k+1}, P_{k+2}, \cdots, P_m\}$ know $\{\boldsymbol{s}_{k+1,k}, \boldsymbol{s}_{k+2,k}, \cdots, \boldsymbol{s}_{m,k}\}$, and only the central sever knows $\boldsymbol{Y}_k$. Therefore, the adversary must also require the server to collude with these $(m-k)$ group members. In summary, to obtain $\boldsymbol{G}_k$, which reveals the privacy of P_k, the central server must collude with all other $(m-1)$ members $\{P_1, \cdots, P_{k-1}, P_{k+1}, P_{k+2}, \cdots, P_m\}$.

Theorem 2. In FLMSS, in the face of an adversary that is the honest-but-curious server or a group composed of any number of other participants in collusion, P_k can guarantee the privacy of the local dataset $\mathcal{D}_k$.

Proof. First, we demonstrate the secret sharing process. Without the participation of the central server, the adversary cannot obtain $\boldsymbol{Y}_k$ and $\boldsymbol{s}_{k,k}$. Without the help of other participants, the adversary cannot obtain $\boldsymbol{s}_{k,j}$ and $\boldsymbol{s}_{i,k}$, where $i \neq k$, $j \neq k$. Next, the security of the process of handling dropped users is discussed. The process of dealing with dropped participants involves $\mathcal{P}^{\text{drop}}$ and $\mathcal{P}^{\text{help}}$, so their security needs to be demonstrated. If P_k is any dropped user in $\mathcal{P}^{\text{drop}}$, neither the central server nor the other participants can obtain $\boldsymbol{Y}_k$ and $\boldsymbol{s}_{k,k}$. If P_k is any helper in $\mathcal{P}^{\text{help}}$, P_k is required by the FLMSS protocol to participate in another round of secret sharing in the process that handles the dropped participants before uploading the new local aggregation result, where the situation is consistent with the above proved situation. In summary, with the proof of Theorem 1, the adversary cannot recover $\boldsymbol{G}_k$ in the above two cases, i.e., the privacy of local dataset $\mathcal{D}_k$ is guaranteed.

Theorem 3. In the FLMSS, which is a d-shot, suppose there are a total of N participants, x participants are curious about colluding with the central server, and P_k is any normal participant. Then, the probability that an adversary composed of curious participants and the honest-but-curious server can

DOI: 10.52396/JUSTC-2022-0116
JUSTC, **2024**, 54(1): 0104

obtain the training data privacy $\boldsymbol{G}_k$ of P_k is $\left(\frac{N-1-d}{N-1}\right)^{N-1-x}\cdot\prod_{i=1}^{d}\frac{x+1-i}{N-i}$.

Proof. Recovery of the local gradient $\boldsymbol{G}_k$ by the adversary first requires that all other $(N-1-x)$ normal participants do not actively send secret shares to P_k. The probability that one normal participant does not send a secret share to P_k is $\frac{N-1-d}{N-1}$, so the conditional probability obtained from this is $\left(\frac{N-1-d}{N-1}\right)^{N-1-x}$. On this basis, a successful attack on P_k also requires that P_k send all of its d shares generated actively to the curious participants, and this probability is $\prod_{i=1}^{d}\frac{x+1-i}{N-i}$. Thus, the probability that the local gradient $\boldsymbol{G}_k$ of a normal participant P_k can be stolen by the adversary is $\left(\frac{N-1-d}{N-1}\right)^{N-1-x}\cdot\prod_{i=1}^{d}\frac{x+1-i}{N-i}$.

To better analyze the security of the FLMSS, we plot surface graphs of the probability of a normal participant being successfully attacked as a function of environmental parameters. When $5000\leqslant N\leqslant 10000$ and $0\leqslant x\leqslant 5000$, the graphs of $P=f(N,x)$, where $d=5,10$ are shown in Fig. 4. According to Theorem 3, we also plot the probability of a normal participant being successfully attacked as a function of the number of curious participants when the total number of participants N is fixed to better demonstrate the impact of d on the security of the FLMSS (see Fig. 5). As shown in Fig. 4, in the face of drastic changes in the federated learning environment, a small value of d enables FLMSS to guarantee almost 100% security in the majority of cases. Fig. 5 shows that a very small increase in d can greatly increase the privacy-preserving ability of FLMSS. An appropriate increase in the value of d is an extremely effective option for communication costs.

To better demonstrate the superiority of FLOSS and FLMSS in terms of security, we compare the security of our proposed schemes with the current state-of-the-art secret sharing-based security aggregation schemes, as shown in Table 3. $t=\left\lfloor\frac{2N}{3}\right\rfloor+1$ in Table 3 is the secret sharing threshold in Ref. [21]. In SSDDL, since the grouping is controlled by the central server, as long as the number of curious participants colluding with the server is greater than $m-2$, the adversary can attack any participant. Increasing m will exponentially increase the communication cost, which will be proven later. However, in our FLOSS, m increases drastically without worrying about a surge in communication costs. The work of Zheng et al.[23] has similar principles and security. In Bonawitz's protocol[21], even if t takes the suggested value $\left\lfloor\frac{2N}{3}\right\rfloor+1$ and the number of curious participants colluding with the server is greater than $\left\lceil\frac{N}{3}\right\rceil$, the privacy of all normal participants will be exposed. At the same time, the capacity to handle dropped participants is also limited. In contrast, both the security and robustness of our FLMSS show great advantages. When $d=10$ and $x\leqslant 3N/5$, the probability of privacy leakage of one participant, which is less than 0.0001104, can be negligible, where $N=10000$. Changes in the large value of N have little effect on this probability. For practical scenarios in federated learning where curious participants account for a small proportion, the protection provided by our FLMSS is sufficient. Moreover, FLMSS can handle any number of

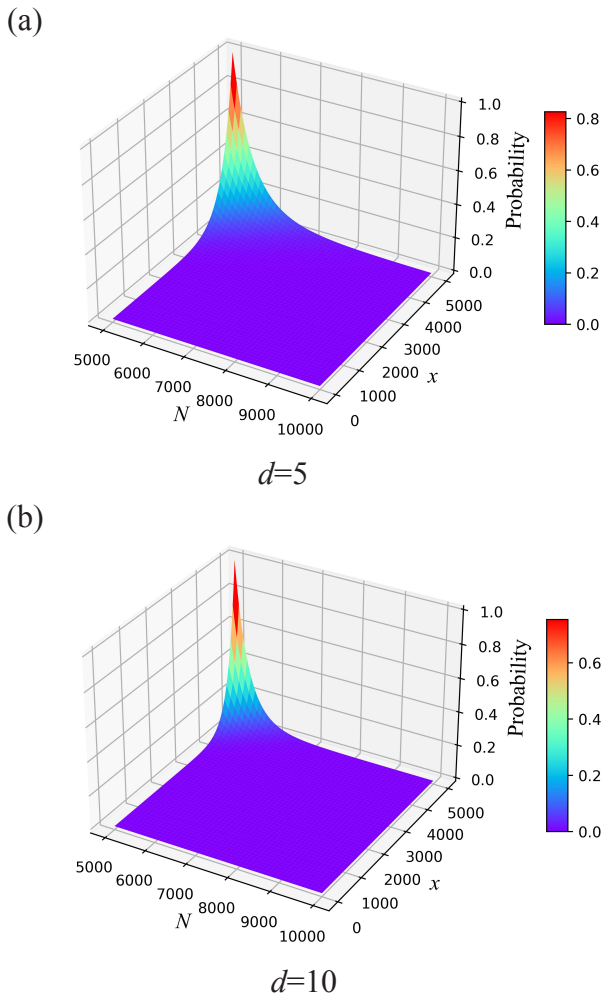


Fig. 4. Surface graphs of the security of FLMSS as a function of environmental parameters when the number of shots d is fixed.

Table 3. Security comparison among secure aggregation schemes based on secret sharing.

Scheme	Security against adversary		
	Sever	Participants	Sever & x participants
Bonawitz et al.[21]	conditional	√	$x\leqslant\lceil 2t-N\rceil-1$
SSDDL[22]	√	√	$x\leqslant m-2$
Zheng et al.[23]	√	√	$x\leqslant m-2$
Proposed FLOSS	√	√	$x\leqslant m-2$
Proposed FLMSS ($d=10$)	√	√	almost $x\leqslant\frac{3}{5}N$

DOI: 10.52396/JUSTC-2022-0116
JUSTC, **2024**, 54(1): 0104

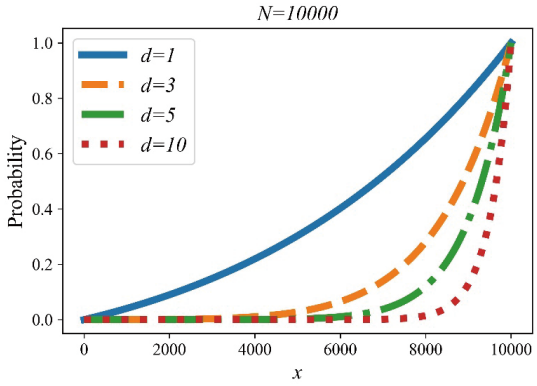


Fig. 5. Curves of the security of FLMSS as a function of the number of curious participants x when the total number of participants N is fixed.

participant dropouts.

7 Experimental results

In this section, we conduct experiments to evaluate the performance of our proposed privacy-preserving FLOSS and FLMSS schemes. We set the operation environment, which consists of an Intel(R) Xeon(R) Gold 5117 CPU, 128.00 GB RAM, an NVIDIA RTX 3090 GPU, and a 64-bit operating system with the PyTorch platform. We evaluate our proposed schemes on a well-known standard dataset: MNIST[42]. The MNIST dataset consists of 60000 training samples and 10000 test samples. The sample images are single-channel 28×28 handwritten digit images. The models we use for accuracy evaluation are multilayer perceptron (MLP) and LeNet-5, which have 104938 and 136886 parameters, respectively. We also use ResNet-18[7], ResNet-34, and ResNet-50 for performance evaluation, which have 11689512, 21797672, and 25557032 parameters, respectively. We use C++ to realize encryption and secret sharing based on CryptoPP.

7.1 Accuracy evaluation

According to the principle of secret sharing and the implementation mechanism of our proposed scheme, FLOSS and FLMSS do not cause noteworthy damage to the transmitted model parameters, i.e., they do not decrease the model accuracy on the basis of the original gradient compression algorithm. Therefore, the purpose of this section focuses on validation.

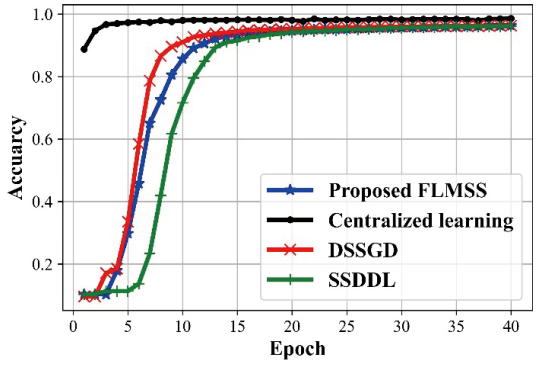


Fig. 6. Comparison of model accuracy for different learning methods.

First, we conduct a comparative experiment of model utility using the LeNet model on the MNIST dataset. We use centralized learning in this scenario as a traditional benchmark. We use randomly selective SGD as a simple gradient compression method and add DSSGD[38], which only uses this method as a privacy-preserving method for comparison. Moreover, we also reimplement the SSDDL scheme[22], which is also based on secret sharing. The number of participants N is set to 100, where each participant has 1% of the entire dataset. The sampling rate η of gradient selection is uniformly set to 0.1. Since the comparison schemes have no mechanism to handle user dropouts, we do not set up user dropout cases. We can use the performance of FLMSS to represent that of FLOSS without needing to repeat the experiment. The comparison of the accuracy performance is illustrated in Fig. 6. Since DSSGD[38], SSDDL[22], and our proposed FLMSS use the same distributed SGD method and the secret sharing mechanism does not affect the aggregated model parameter values, their accuracy performances are almost the same. Compared with the traditional method without gradient selection and using centralized training, our proposed scheme converges slower, but the final model accuracy is almost the same.

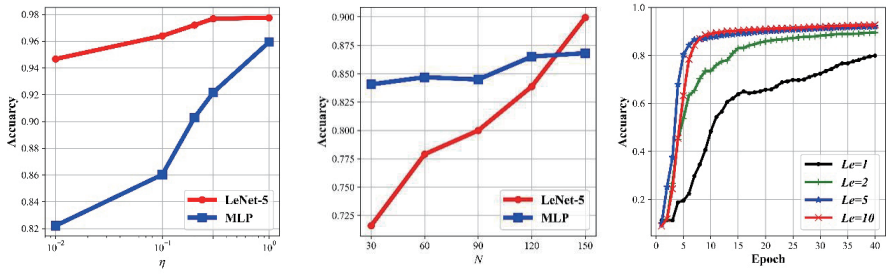
Subsequently, we briefly verify the impact of key parameter settings in federated learning on model accuracy. According to the principle of control variables, we set the benchmark values of the gradient selection rate η, the number of participants N, and the number of local training epochs Le to 0.1, 100, 5 , respectively. We vary η, N, and Le in the range of $\{0.01,0.1,0.2,0.3,1\}$, $\{30,60,90,120,150\}$, $\{1,2,5,10\}$, respectively. To better simulate the federated learning scenario, in the experiment for N, each participant has a smaller proportion of the data, which is set to 0.2% of the entire dataset. As shown in Fig. 7, when the gradient uploading rate η increases, the model accuracy increases. However, when we adopt more advanced gradient compression methods, the model usability is affected by the new compression method itself. A larger number of participants N means that more data are involved in training, which will improve the model accuracy. When the local epoch Le increases, the global model converges faster and has higher accuracy, which is consistent with the relevant conclusions of FedAvg[8]. We can conclude from the model performance experiments that our proposed privacy-preserving schemes will not affect the model performance of the original federated learning algorithm.

7.2 Performance evaluation

7.2.1 Communication performance

In the federated learning scenario, participants usually do not have a large communication bandwidth and computing power. The communication cost of the participants has become an important factor restricting the development of federated learning. For fairness, we only count the participants' communication overhead incurred during one round of global training. We use N to denote the number of participants in the round, n to denote the number of parameters, b to denote the bit precision, and η to denote the gradient compression

DOI: 10.52396/JUSTC-2022-0116
JUSTC, **2024**, 54(1): 0104



(a) The effect of η on the accuracy

(b) The effect of N on the accuracy

(c) The effect of Le on the accuracy (MLP for MNIST)

Fig. 7. Model performance under the influence of different parameter settings.

rate. In FLOSS and SSDDL, we use m to denote the group size and $m \leqslant N$. In FLOSS, FLMSS, and Bonawitz's protocol, we all use b_k to denote the number of bits in a key exchange public key. We use b_s to denote the number of bits in a secret share of Bonawitz's protocol.

In FLOSS, N participants first download a model with a size of nb from the central server, and the communication cost is Nnb. Each participant then uploads its own public key and downloads the public keys of all other participants in its sharing group, with a communication cost of Nmb_k. After the participants complete the secret sharing through key agreement, they upload the compressed local aggregation results to the central server, and the communication cost is $N\eta nb$. Therefore, the total cost of our proposed FLOSS in one round of training is $N(mb_k + (\eta+1)nb)$. The work of Zheng et al.[23] considers group management as an optional practical solution, and their scheme without group management is $N(Nb_k + (\eta+1)nb)$ according to FLOSS. Of course, they are different in the method of implementing compression of model parameters. In FLMSS, each participant that has uploaded its own public key to the sever shares a secret with one other participant by key agreement, with the total communication cost of both parties being $2b_k$. The above communication process is performed d times by each participant, so the communication cost of the system for secret sharing is approximately $2Ndb_k$. Hence, the total communication cost of the FLMSS in one round of training is $N(2db_k + (\eta+1)nb)$. We summarize the one-round participant communication costs of several other secret sharing-based privacy-preserving distributed learning schemes in Table 4. The privacy-preserving capability of DSSGD is poor, and its communication cost is a starting point for reference. Compared to SSDDL, our FLOSS has the same level of strict protection and lower communication costs. Compared to Bonawitz's protocol and other schemes, our FLMSS has strong privacy protection strength and the ability to handle dropped participants, but the proportion of communication cost used for privacy protection is small to negligible compared to the original cost.

We measure the impact of the number of participants N and the number of model parameters n on the communication cost. First, we train a LeNet network with 136886 parameters on the MNIST dataset. We set some moderate parameter values where group size $m = 50$, number of shots $d = 10$, and the rate $\eta = 0.1$. In fact, the protection strength of FLOSS with $m = 50$ is far from sufficient compared with FLMSS where $d = 10$, and η can be improved by using more advanced gradient compression methods. To make a more intuitive comparison of the communication costs used for privacy protection, we omit the basic communication overhead common to all these schemes for transferring model parameters. As shown in Fig. 8a, while the FLMSS guarantees a very high protection effect, it hardly increases the extra communication cost. Moreover, even if we increase the value of m to

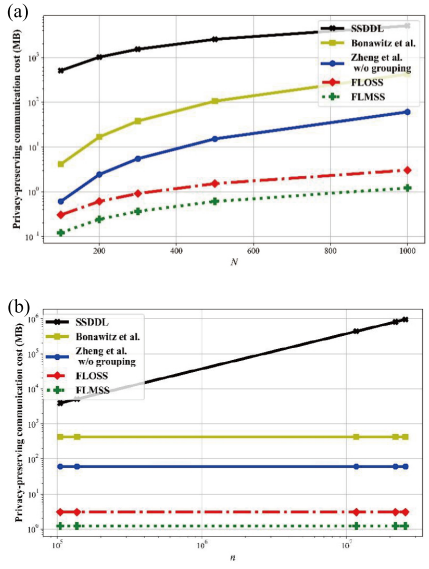


Fig. 8. (a) The communication cost used for privacy protection under different numbers of participants. (b) The communication cost used for privacy protection under different numbers of model parameters.

Table 4. Comparison of the participants' communication costs.

Scheme	Communication cost	Cost complexity
DSSGD[38]	$N(1+\eta)nb$	$O(Nn)$
Bonawitz et al.[21]	$N(2Nb_k + 5(N-4)b_s + 2nb)$	$O(N^2 + Nn)$
SSDDL[22]	$N(1+(2m-1)\eta)nb$	$O(Nmn)$
Zheng et al.[23]	$N(Nb_k + (\eta+1)nb)$	$O(N^2 + Nn)$
Proposed FLOSS	$N(mb_k + (\eta+1)nb)$	$O(Nm + Nn)$
Proposed FLMSS	$N(2db_k + (\eta+1)nb)$	$O(Nn)$

DOI: 10.52396/JUSTC-2022-0116
JUSTC, 2024, 54(1): 0104

increase the protection strength, the communication cost increase of our FLOSS is negligibly small compared to SSDDL, which can be inferred from Table 4. Furthermore, due to the logarithmic processing of the vertical axis, in fact, the superiority of our proposed schemes in communication cost is much greater than the intuitive perception of Fig. 8a. Second, when we set $N = 1000$ and change the model parameters n for simulation, we obtain Fig. 8b. The same conclusion can be drawn: our methods achieve a huge reduction in communication cost while achieving better protection compared to state-of-the-art woks. Apart from SSDDL, the communication overhead of other schemes used for privacy protection is independent of the size of the model parameters.

7.2.2 Computation performance

To intuitively reflect the computational cost of the system for secure aggregation, computation performance experiments no longer count model training, communication and other time consumption but focus on the process of encryption, masking, and aggregation. We simulate 100 clients and one server and dynamically adjust the structure of the LeNet model to obtain varying model parameters. We make a comparison with the most related and state-of-the-art works[21, 23] without heavy cryptographic operations, which are able to deal with dropout.

First, we examine the participant-side and sever-side computation performance without considering dropout. We set up the model LeNet with 136886 parameters and change the fraction of selected clients to participate in federated learning, i.e., the actual number of participants, to examine the average computation performance of each participant to encrypt the model and the sever to aggregate the model securely in different works. Considering the total number of participants, we set the size of the group in FLOSS to 10 and $d = 3$ in FLMSS. To control variables, we omitted operations that may affect runtime in the work of Zheng et al.[23], such as quantization techniques and trusted hardware. We show the participant's running time of encrypting model updates for varying fractions of participants selected in one round using different schemes in Fig. 9a. Since FLOSS uses a grouping strategy and FLMSS uses a fixed number of secret-sharing shots, the participant-side computation cost of our proposed scheme does not increase with the number of participants. The participant-side computation costs of works of Bonawitz et al.[21] and Zheng et al.[23] increase linearly with the number of participants. In Fig. 9a, we show computation costs on the sever side. In Ref. [21], the sever is required to perform reconstruction of secret keys and recompute masks to be subtracted from the uploaded model whether dropouts occur or not, which brings huge computation overhead. The central sever only needs to perform simple additive aggregation in our schemes. Then, we fix the selection fraction to 0.3 and examine the computation performance of clients and the sever when the size of the model varies. In Fig. 10, the computation costs of participants and the server for encryption and secure aggregation increase linearly with increasing model size.

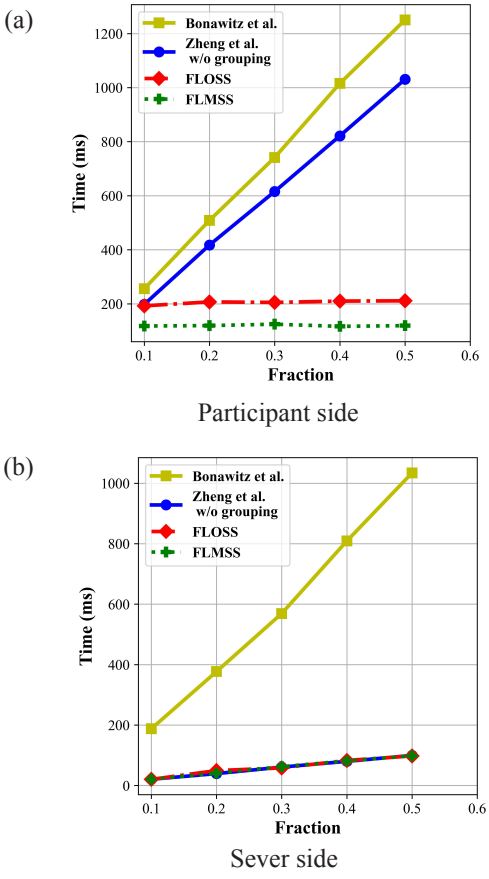


Fig. 9. Computation-cost comparison with varying fractions of selected participants per round.

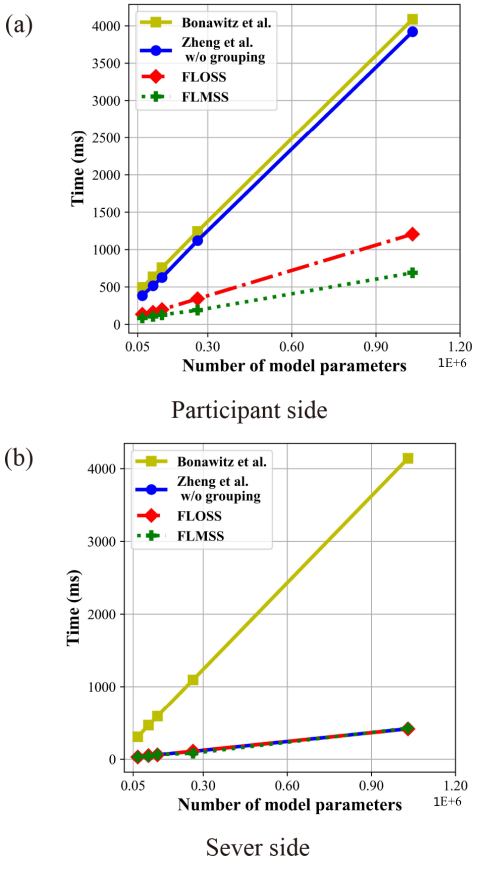


Fig. 10. Computation-cost comparison with varying numbers of model parameters per round.

DOI: 10.52396/JUSTC-2022-0116
JUSTC, **2024**, 54(1): 0104

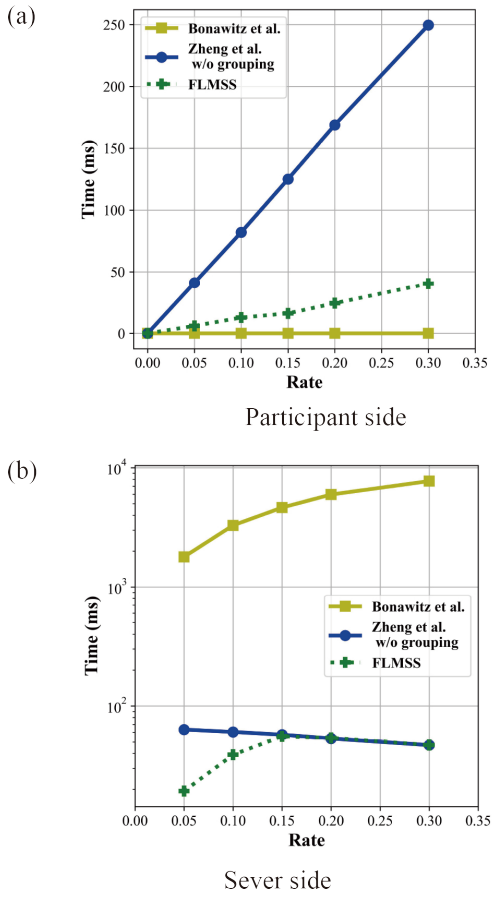


Fig. 11. Computation cost for handling dropout with varying dropout rates.

On the server side, where computation power is usually sufficient, the grouping management of FLOSS and the sparse connection method of FLMSS do not provide computing performance advantages when no dropout occurs. However, most importantly, on the client side where the performance is most constrained, the performance overhead of our proposed scheme is significantly lower than those of other schemes.

Although the work of Zheng et al.[23] cannot solve the nesting of participant dropout, we are still interested in comparing the computation performance when participant dropout occurs. For the above reason, we only consider one round of dropout. The selection fraction and the number of model parameters are fixed to 0.4 and 136886. When we change the dropout rate of participants, we can obtain Fig. 11. It is worth noting that the running time in Fig. 11 is dedicated to the time spent dealing with dropout. Since there is no separate process to address participants' dropout in the work of Bonawitz et al.[21], we regard the difference between the running time where participants drop out and the benchmark running time as the time spent addressing dropout. Participants in the work of Bonawitz et al. do not need to make additional calculations for the dropout of other participants. Although this sounds like a good advantage, Fig. 9 and Fig. 10 show that participants in their scheme have the disadvantage in computation overhead in the regular process. The computational cost of FLMSS for dealing with dropout is low on the participant side and the server side. Another important advantage that the average running time in Fig. 11a cannot reflect is that when the dropout rate is low, FLMSS only requires a few participants to offer assistance, while Zheng et al.'s scheme[23] requires all online participants to start the assistance program, which easily causes new large-scale dropout. This advantage can be verified from the difference in the front part of the curve in Fig. 11b.

Table 5. Performance complexity comparison.

Scheme	Client computation	Sever computation
Bonawitz et al.[21]	$O(N^2+nN)$	$O(n(N-N_d)+nN_d(N-N_d))$
Zheng et al.[23]	$O(n(N+N_d))$	$O(n(N-N_d))$
Proposed FLOSS	$O(mn)$	$O(nN)$
Proposed FLMSS	$O(n(d+N_d))$	$O(n(N-N_d))$

Finally, we compare the theoretical computational complexity. The number of dropout participants is denoted as N_d. Other notations follow the previous descriptions. Overall, FLOSS leads to $O(mn)$ computations on each client side and $O(Nn)$ computations on the server side. The FLMSS leads to $O(n(d+N_d))$ computations on each client side and $O(n(N-N_d))$ computations on the server side. According to related works[21, 23], we summarize the comparison of the asymptotic computational complexity in Table 5.

8 Conclusions

We propose two computation-communication efficient secure aggregation schemes for privacy-preserving federated learning based on secret sharing: FLOSS and FLMSS. FLOSS can achieve strict privacy preservation with a small communication cost. FLMSS has the ability to provide high-intensity privacy preservation to each participant in federated learning and efficiently handle participant dropout. Theoretical proof of security under the threat model of curious participants and honest-but-curious central servers is given. Finally, we demonstrate the superiority of our proposed FLOSS and FLMSS through extensive experiments.

Acknowledgements

This work was supported by the National Key Research and Development Program of China (2018YFB0804102), the National Natural Science Foundation of China (61802357), the Fundamental Research Funds for the Central Universities (WK3480000009), and the Scientific Research Startup Funds of the Hefei University of Technology (13020-03712022064).

Conflict of interest

The authors declare that they have no conflict of interest.

Biographies

Xuan Jin received his B.E. degree from the the Hohai University in 2022. He is currently a master's student at the University of Science and Technology of China. His research interests include deep learning and applied cryptography.

Yuanzhi Yao is currently an Associate Professor at the Hefei University of Technology. He received his Ph.D. degree from the University

DOI: 10.52396/JUSTC-2022-0116
JUSTC, **2024**, 54(1): 0104

of Science and Technology of China in 2017. His research interests include deep learning and multimedia security.

References

[1] LeCun Y, Bengio Y, Hinton G. Deep learning. *Nature,* **2015**, *521*: 436–444.

[2] Redmon J, Divvala S, Girshick R, et al. You only look once: Unified, real-time object detection. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Las Vegas, USA: IEEE, **2016**: 779–788.

[3] Minaee S, Kalchbrenner N, Cambria E, et al. Deep learning: Based text classification: A comprehensive review. *ACM Computing Surveys,* **2021**, *54* (3): 1–40.

[4] Lee M, Sanz L R D, Barra A, et al. Quantifying arousal and awareness in altered states of consciousness using interpretable deep learning. *Nature Communications,* **2022**, *13*: 1064.

[5] Wright L G, Onodera T, Stein M M, et al. Deep physical neural networks trained with backpropagation. *Nature,* **2022**, *601*: 549–555.

[6] Szegedy C, Liu W, Jia Y, et al. Going deeper with convolutions. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Boston, USA: IEEE, **2015**: 1–9.

[7] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Las Vegas, USA: IEEE, **2016**: 770–778.

[8] McMahan H B, Moore E, Ramage D, et al. Communication-efficient learning of deep networks from decentralized data. arXiv: 1602.05629, **2016**.

[9] Nasr M, Shokri R, Houmansadr A. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In: 2019 IEEE Symposium on Security and Privacy (SP). San Francisco, USA: IEEE, **2019**: 739–753.

[10] Wang Z, Song M, Zhang Z, et al. Beyond inferring class representatives: User-level privacy leakage from federated learning. In: IEEE INFOCOM 2019 —IEEE Conference on Computer Communications. Paris, France: IEEE, **2019**: 2512–2520.

[11] Zhu L, Liu Z, Han S. Deep leakage from gradients. In: Proceedings of the 33rd International Conference on Neural Information Processing Systems. New York: ACM, **2019**, 1323: 14774–14784.

[12] Hitaj B, Ateniese G, Perez-Cruz F. Deep models under the GAN: Information leakage from collaborative deep learning. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. New York: ACM, **2017**: 603–618.

[13] Xu G, Li H, Liu S, et al. VerifyNet: Secure and verifiable federated learning. *IEEE Transactions on Information Forensics and Security,* **2020**, *15*: 911–926.

[14] Mothukuri V, Parizi R M, Pouriyeh S, et al. A survey on security and privacy of federated learning. *Future Generation Computer Systems,* **2021**, *115*: 619–640.

[15] Abadi M, Chu A, Goodfellow I, et al. Deep learning with differential privacy. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. New York: ACM, **2016**: 308–318.

[16] Phong L T, Aono Y, Hayashi T, et al. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Transactions on Information Forensics and Security,* **2018**, *13* (5): 1333–1345.

[17] Zhang X, Chen X, Liu J K, et al. DeepPAR and DeepDPA: Privacy preserving and asynchronous deep learning for industrial IoT. *IEEE Transactions on Industrial Informatics,* **2020**, *16* (3): 2081–2090.

[18] Huang K, Liu X, Fu S, et al. A lightweight privacy-preserving CNN feature extraction framework for mobile sensing. *IEEE Transactions on Dependable and Secure Computing,* **2021**, *18* (3): 1441–1455.

[19] Fereidooni H, Marchal S, Miettinen M, et al. SAFELearn: Secure aggregation for private Federated learning. In: 2021 IEEE Security and Privacy Workshops (SPW). San Francisco, USA: IEEE, **2021**: 56–62.

[20] Yang Y, Mu K, Deng R H. Lightweight privacy-preserving GAN framework for model training and image synthesis. *IEEE Transactions on Information Forensics and Security,* **2022**, *17*: 1083–1098.

[21] Bonawitz K, Ivanov V, Kreuter B, et al. Practical secure aggregation for privacy-preserving machine learning. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. New York: ACM, **2017**: 1175–1191.

[22] Duan J, Zhou J, Li Y. Privacy-Preserving distributed deep learning based on secret sharing. *Information Sciences,* **2020**, *527*: 108–127.

[23] Zheng Y, Lai S, Liu Y, et al. Aggregation service for federated learning: An efficient, secure, and more resilient realization. *IEEE Transactions on Dependable and Secure Computing,* **2022**, *20* (2): 988–1001.

[24] Xu R, Baracaldo N, Zhou Y, et al. HybridAlpha: An efficient approach for privacy-preserving federated learning. In: Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security. New York: ACM, **2019**: 13–23.

[25] Wu D, Pan M, Xu Z, et al. Towards efficient secure aggregation for model update in federated learning. In: GLOBECOM 2020—2020 IEEE Global Communications Conference. Taipei, China: IEEE, **2020**: 1–6.

[26] Truex S, Baracaldo N, Anwar A, et al. A hybrid approach to privacy-preserving federated learning. *Informatik Spektrum,* **2019**, *42*: 356–357.

[27] Kadhe S, Rajaraman N, Koyluoglu O O, et al. FastSecAgg: Scalable secure aggregation for privacy-preserving federated learning. arXiv: 2009.11248, **2020**.

[28] So J, Güler B, Avestimehr A S. Turbo-aggregate: Breaking the quadratic aggregation barrier in secure federated learning. *IEEE Journal on Selected Areas in Information Theory,* **2021**, *2* (1): 479–489.

[29] Karimireddy S P, Kale S, Mohri M, et al. SCAFFOLD: stochastic controlled averaging for federated learning. In: Proceedings of the 37th International Conference on Machine Learning. New York: ACM, **2020**: 5132–5143.

[30] Ozfatura E, Ozfatura K, Gündüz D. FedADC: Accelerated federated learning with drift control. In: 2021 IEEE International Symposium on Information Theory (ISIT). Melbourne, Australia: IEEE, **2021**: 467–472.

[31] Shamir A. How to share a secret. *Communications of the ACM,* **1979**, *22* (11): 612–613.

[32] Diffie W, Hellman M. New directions in cryptography. *IEEE Transactions on Information Theory,* **1976**, *22* (6): 644–654.

[33] Blum M, Micali S. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM Journal on Computing,* **1984**, *13* (4): 850–864.

[34] Bellare M, Yee B. Forward-security in private-key cryptography. Topics in cryptology—CT-RSA 2003. Berlin, Heidelberg: Springer, **2003**: 1–18.

[35] Shen J, Yang H, Vijayakumar P, et al. A privacy-preserving and untraceable group data sharing scheme in cloud computing. *IEEE Transactions on Dependable and Secure Computing,* **2022**, *19* (4): 2198–2210.

[36] Fan K, Chen Q, Su R, et al. MSIAP: A dynamic searchable encryption for privacy-protection on smart grid with cloud-edge-end. *IEEE Transactions on Cloud Computing,* **2021**, *11*: 1170–1181.

[37] Lin Y, Han S, Mao H, et al. Deep gradient compression: Reducing the communication bandwidth for distributed training. arXiv: 1712.01887, **2017**.

[38] Shokri R, Shmatikov V. Privacy-preserving deep learning. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. New York: ACM, **2015**: 1310–1321.

[39] Vogels T, Karimireddy S P, Jaggi M. PowerSGD: practical low-rank gradient compression for distributed optimization. In: Proceedings of the 33rd International Conference on Neural Information Processing Systems. New York: ACM, **2019**: 14269–14278.

[40] Abdulrahman S, Tout H, Ould-Slimane H, et al. A survey on federated learning: The journey from centralized to distributed on-site learning and beyond. *IEEE Internet of Things Journal,* **2021**, *8* (7): 5476–5497.

[41] Rahman S A, Tout H, Talhi C, et al. Internet of Things intrusion detection: Centralized, on-device, or federated learning. *IEEE Network,* **2020**, *34* (6): 310–317.

[42] LeCun Y, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE,* **1998**, *86* (11): 2278–2324.

DOI: 10.52396/JUSTC-2022-0116
JUSTC, **2024**, 54(1): 0104