Proceedings of the 8th International Conference on Computing and Data Science
DOI: 10.54254/2755-2721/2026.36451

A Privacy-Preserving Distributed Machine Learning Aggregation Scheme

Mingyuan Li

School of Computer Science, Northwestern Polytechnical University, Xi'an, China
lmy361088@mail.nwpu.edu.cn

Abstract. Federated Learning enables data to remain on local devices, yet malicious servers can still infer sensitive user information by analyzing client-uploaded model updates, posing significant privacy leakage risks. Existing secure aggregation schemes—such as Differential Privacy, Homomorphic Encryption, and traditional Secret Sharing—struggle to achieve an ideal balance among model accuracy, computational/communication overhead, and adaptability to complex aggregation scenarios. To address this, this paper proposes a multi-server secure aggregation scheme based on additive secret sharing. The scheme introduces multiple non-colluding servers; each client splits its local model update into random secret shares and distributes them to these servers. Each server independently performs weighted aggregation, and the global model is reconstructed by a leader server. Theoretical analysis demonstrates that, owing to the linear homomorphism of additive secret sharing, the aggregation result of our scheme is mathematically equivalent to the standard FedAvg algorithm. Experimental results on the MNIST dataset show that our scheme achieves a test accuracy of 86.13% after 30 global rounds, closely matching the baseline FedAvg (86.12%), with only a 6% additional time overhead per round. Under reasonable non-collusion assumptions, the scheme achieves information-theoretic privacy protection, effectively breaking the trilemma among privacy, accuracy, and efficiency.Keywords:Federated Learning, Additive Secret Sharing,Secure Aggregation; Privacy Preservationords.

Keywords: Federated Learning, Additive Secret Sharing, Secure Aggregation, Privacy Preservationords

1. Introduction

Federated Learning (FL), as a distributed machine learning paradigm, allows multiple clients to collaborate on a shared model under the coordination of a central server without uploading raw data to the server [1]. This feature makes federal learning a natural advantage in protecting data privacy, especially in data-sensitive areas such as mobile devices, medical institutions and financial organizations. However, recent studies have shown that federated learning does not naturally have the ability to protect privacy. Malicious servers or third-party attackers can perform reconstruction attacks or member inference attacks by analyzing model updates (such as gradients or weights) uploaded by the client, thereby restoring the training data of the client or judging whether a data is

DOI: 10.54254/2755-2721/2026.36451

involved in training [2]. Therefore, how to effectively protect client privacy under the federal learning framework has become a core issue of concern to academia and industry.

To address the privacy leakage problem in FL, researchers have proposed various technical approaches. Differential Privacy (DP) mitigates inference attacks by adding calibrated noise to model updates, but the introduction of noise inevitably degrades model accuracy, and training high-precision models often requires high privacy budgets [3, 4]. Homomorphic Encryption (HE) allows computations to be performed directly on ciphertexts, providing strong cryptographic guarantees; however, its high computational and communication overhead severely limits practical deployment in cross-device scenarios [5, 6]. Trusted Execution Environments (TEE) rely on hardware isolation mechanisms to ensure runtime security but face potential threats such as side-channel attacks [2]. In recent years, Secure Multi-Party Computation (MPC) frameworks based on secret sharing have garnered significant attention. Schemes such as SecureML [7] and ABY3 [8] enable privacy-preserving machine learning training within MPC frameworks but are primarily designed for data center scenarios and still incur considerable communication overhead. The recently proposed FedShare framework [9] leverages additive secret sharing to achieve efficient federated secure aggregation and is highly relevant to our scheme in both naming and technical approach; however, its implementation has certain limitations, such as inadequately addressing the weighted aggregation problem arising from varying client data sizes, and the protocol's stability under an increasing number of clients requires further validation. Overall, existing schemes still struggle to achieve an ideal balance among privacy, accuracy, and efficiency.

In response to the above challenges, this paper proposes a multi-server secure aggregation scheme based on additive secret sharing. The core idea of the scheme is to abandon the single aggregation server architecture in traditional federated learning and introduce $n \geq 3$ non-collusion servers ; each client splits the local model update into n addition secret shares and distributes them to each server. Any single server cannot restore the original model update. After each server independently completes the weighted aggregation of shares, the master server collects all shares and reconstructs the global model. Through the linear homomorphism of additive secret sharing, this paper strictly proves that the aggregation result of the scheme is mathematically equivalent to the weighted average process of the standard FedAvg [1]. The security of the scheme is based on the reasonable assumptions of ' at least one server does not collude ' and ' at least two clients do not collude ', and does not depend on any computational complexity assumptions [3].

The main contributions of this paper include : (1) A distributed secure aggregation framework based on additive secret sharing is proposed, which achieves strict client-level privacy protection under multiple non-collusion server architectures. (2) It is mathematically proved that the scheme is completely equivalent to the standard FedAvg, which fundamentally avoids the trade-off dilemma of ' privacy budget-precision ' in differential privacy. (3) The scheme only involves random number generation and ordinary arithmetic operations, and does not require public key cryptography operations such as homomorphic encryption. The computational overhead is much lower than that of the homomorphic encryption scheme [6]. Each client only needs $O(K)$ random number generation and addition operations (K is the number of model parameters), and the server only needs $O(mK)$ addition operations (m is the number of clients). (4) The experimental results show that the proposed scheme achieves 86.13 % test accuracy after 30 rounds of global iteration, which is completely equal to FedAvg (86.12 %), and the extra time overhead per round is only 6 %, which verifies the superiority of the scheme in the three dimensions of privacy, accuracy and efficiency.

DOI: 10.54254/2755-2721/2026.36451

2. Preliminaries

2.1. Federated learning

Federated learning was first proposed by McMahan et al. in 2016 as a distributed machine learning paradigm [1]. Its core idea is to allow multiple clients to collaboratively train a shared model under the coordination of a central server without uploading raw data to the server.

Assume there are K clients, with client k holding a local dataset D_k of size $n_k = |D_k|$, and the total global sample size $n = \sum_{k=1}^{K} n_k$. The optimization objective of FL is to minimize the global loss function as shown in Eq. (1):

$$\min_{w \in R^d} f(w) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(w),\ F_k(w) = \frac{1}{n_k} \sum_{i \in D_k} l_i(w) \tag{1}$$

where w denotes the global model parameters and $F_k(w)$ is the local loss of client k.

To address communication efficiency, McMahan et al. proposed the FederatedAveraging algorithm (FedAvg) [1]. In each communication round t, the server selects a subset of clients S_t, sends the current global model w_t to them; each client performs multiple local SGD updates based on w_t and its local data, obtaining $w_{t+1}(k)$, and returns it to the server. FedAvg significantly reduces the number of communication rounds required to train high-quality models by increasing the amount of local computation per client per round [1]. The server then performs a weighted average to update the global model as in Eq. (2):

$$w_{t+1} = \sum_{k \in S_t} \frac{n_k}{n} w_{t+1}^{(k)} \tag{2}$$

2.2. Secret sharing

Secret sharing was independently proposed by Shamir and Blakley in 1979 [3, 4]. Its core idea is to split a secret value into multiple shares and distribute them to different parties, such that no single party can learn the original secret; only when at least a threshold number of parties combine their shares can the secret be reconstructed. This is known as a (t, n) threshold scheme (where n is the total number of parties and$t$ is the minimum number of shares required for reconstruction) [3].

Additive secret sharing is the core technical tool of our scheme. Suppose a secret value $s \in Z_p$ is to be shared among n parties. Randomly choose $n-1$ shares $s_1, s_2, \ldots, s_{n-1}$ and compute $s_n = s - \sum_{i=1}^{n-1} s_i \pmod p$, then send s_i to party P_i. This satisfies in Eq. (3):

$$\sum_{i=1}^{n} s_i = s \pmod p \tag{3}$$

This scheme has perfect security: any subset of fewer than n shares is statistically independent of the secret s, and an adversary cannot obtain any information about ss [3, 5]. Its additive homomorphism enables linear combinations in the share domain to correspond to linear combinations of the original secrets: if $\{s_i\}$ and $\{t_i\}$ are shares of s and t, respectively, then $u_i = s_i + t_i$ is a valid share of $u = s + t$; similarly, $c \bullet s_i$ is a share of $c \cdot s$. This property makes additive secret sharing an ideal tool for implementing secure aggregation in MPC [5], and our scheme leverages this property in Section 3.1 to achieve secure weighted aggregation on the server side.

DOI: 10.54254/2755-2721/2026.36451

3. System model

3.1. System architecture

The system architecture of our scheme, as illustrated in Figure 1, comprises two types of entities: a Client set $\{c_1, c_2, \ldots, c_m\}$ and a Server set S=$\{ s_1 , s_2 ,\ldots, s_n \}$. Clients are participants holding local private datasets (e.g., mobile devices or institutions), while servers are multi-party computation nodes responsible for collaboratively completing the secure aggregation computation. In this scheme, the number of servers $n \geq 3$. In each global round, one server is randomly selected as the leader server, and the remaining $n-1$ servers act as auxiliary servers. The complete workflow is as follows:

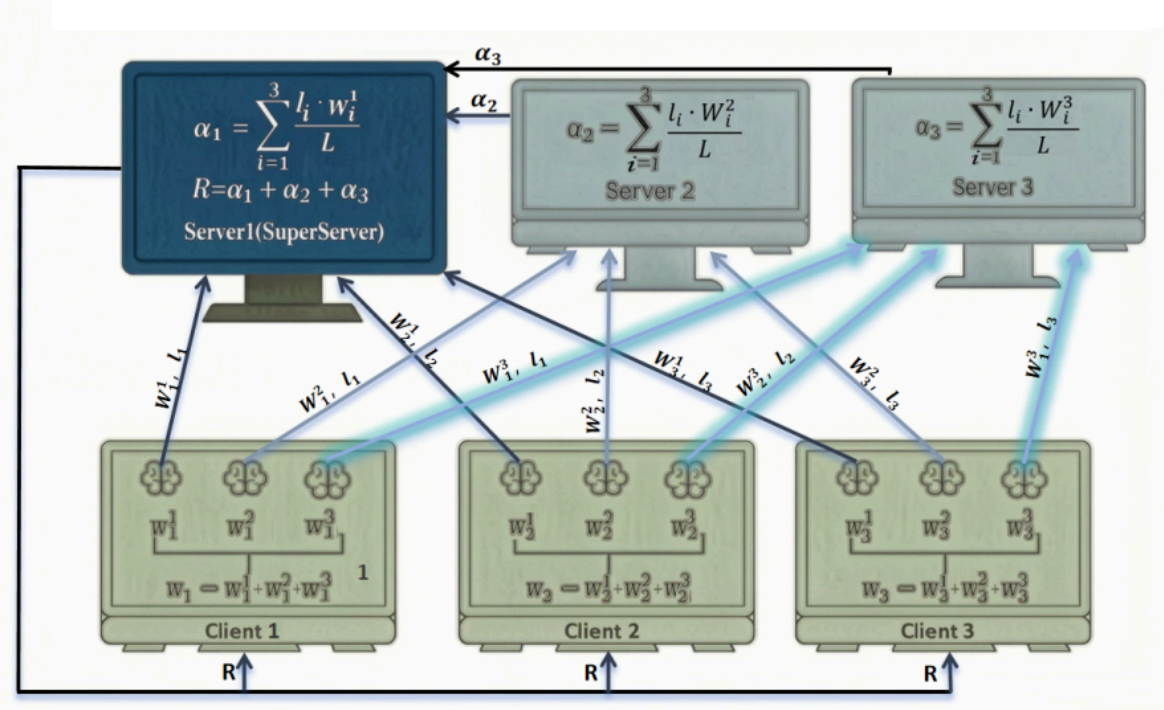

Figure 1. System architecture siagram of our scheme

Phase 1: Local Training and Secret Splitting. In each global round, each client c_i first performs several rounds of local SGD training based on the current global model weights W and its local dataset d_i (hyperparameters include local epochs E, batch size B, and learning rate η), obtaining a local model update W_i. Subsequently, client c_i splits W into n additive secret shares: randomly generate $n-1$ random shares $W_i^1, W_i^2, \ldots, W_i^{n-1}$, and compute $W_i^n = W_i - \sum_{j=1}^{n-1} W_i^j$ such that $\sum_{j=1}^{n-1} W_i^j = W_i$.

Phase 2: Server-Side Share Aggregation. Each server S_j, upon receiving the corresponding shares $\left\{W_1^j, W_2^j, \ldots, W_m^j\right\}$ from all m clients, computes the weighted aggregation based on each client's local dataset size l_i. The weighted aggregated share obtained by server S_j is in Eq. (4):

$$\alpha_j = \sum_{i=1}^{m} \left(\frac{l_j}{L} \bullet W_i^j\right) \tag{4}$$

DOI: 10.54254/2755-2721/2026.36451

where $L = \sum_{i=1}^{m} l_i$. Each auxiliary server then sends α_j to the leader server for this round.

Phase 3: Leader Server Reconstructs the Global Model. The leader server collects its own computed α_{leader} and all $\{\alpha_j\}_{j \neq leader}$ from the auxiliary servers, and computes their sum as in Eq.(5):

$$R = \sum_{j=1}^{n} \alpha_j = \sum_{j=1}^{n} \sum_{i=1}^{m} \left(\frac{l_i}{L} \bullet W_i^j \right) = \sum_{i=1}^{m} \frac{l_i}{L} \bullet \sum_{j=1}^{n} W_i^j = \sum_{i=1}^{m} \frac{l_i}{L} \bullet W_i \quad (5)$$

This result R is exactly the weighted average result of the standard FedAvg algorithm [1]. The leader server broadcasts Rto all clients, which then update their local models accordingly and proceed to the next round. The above derivation rigorously demonstrates that the aggregation result of our scheme is precisely equivalent to FedAvg—this serves as the core theoretical guarantee for lossless accuracy in our scheme in Figure 1:

3.2. Threat model

This scheme adopts the standard Honest-but-Curious (semi-honest) threat model used in Secure Multi-Party Computation (MPC) [7], also referred to as the passive adversary model. Under this model, all participating clients and servers faithfully follow the protocol specifications and execute the prescribed computations, without sending erroneous or maliciously crafted data; however, they may be curious and attempt to infer other parties' private inputs from the intermediate information collected during protocol execution. Specifically, this scheme assumes:Server Non-Collusion. No two servers collude; i.e., the honest parties constitute a majority, and an adversary can control at most one server [9]. Client Non-Collusion. No two clients collude; i.e., an adversary can control at most one client [9]. External Attackers. This scheme does not currently consider active attacks such as Denial-of-Service, model poisoning, or message tampering; defenses against such threats are left for future work [2].

Under the above threat model, this scheme ensures that even if an adversary fully controls one server (or one client), it cannot infer any honest client's original model update or private training data from the secret shares observed by that server/client.

3.3. Threat model

This scheme adopts the standard Honest-but-Curious (semi-honest) threat model used in Secure Multi-Party Computation (MPC) [7], also referred to as the passive adversary model. Under this model, all participating clients and servers faithfully follow the protocol specifications and execute the prescribed computations, without sending erroneous or maliciously crafted data; however, they may be curious and attempt to infer other parties' private inputs from the intermediate information collected during protocol execution. Specifically, this scheme assumes:

Server Non-Collusion. No two servers collude; i.e., the honest parties constitute a majority, and an adversary can control at most one server [9].

Client Non-Collusion. No two clients collude; i.e., an adversary can control at most one client [9].

External Attackers. This scheme does not currently consider active attacks such as Denial-of-Service, model poisoning, or message tampering; defenses against such threats are left for future work [2].

Under the above threat model, this scheme ensures that even if an adversary fully controls one server (or one client), it cannot infer any honest client's original model update or private training

DOI: 10.54254/2755-2721/2026.36451

data from the secret shares observed by that server/client.

3.4. Design goals

Privacy Preservation. Under the threat model described in Section 3.2, ensure that the local model updates W_i uploaded by clients are not disclosed to any single server or client. Specifically, any single server holds only one random secret share W_i^j from each client and cannot reconstruct W_i without obtaining the other $n-1$ shares; any single client obtains only the final global aggregation result R and cannot learn other clients' individual updates. The strength of this privacy protection relies on the information-theoretic security of additive secret sharing and does not depend on any computational hardness assumptions [3].

Lossless Model Accuracy. This scheme requires that the aggregation result be mathematically equivalent to the weighted average result of the standard FedAvg algorithm [1]. As derived in Section 3.1, due to the linear homomorphism of additive secret sharing, the global model RR aggregated at the server side exactly matches FedAvg's update, introducing no approximation errors, truncation errors, or noise. Therefore, the scheme achieves the same model convergence performance and final accuracy as the original FL algorithm while preserving privacy.

Efficiency. This scheme aims to keep communication and computational overhead within acceptable bounds while achieving strong privacy protection. Specific goals are: in each global round, each client needs only O(n) share transmissions (one share to each of the n servers), each server performs only O(m) additions for share aggregation, and no public-key cryptographic operations are required. This design makes the scheme feasible for practical deployment in large-scale cross-device FL scenarios.

4. Proposed scheme

4.1. Design philosophy

The design philosophy of our scheme stems from a core dilemma in Federated Learning: how to protect client data privacy while maintaining model accuracy and system efficiency.

Distributed Trust Architecture. We abandon the traditional single aggregation server architecture and introduce multiple non-colluding servers. Leveraging the information-theoretic security of additive secret sharing [3], each client splits its model update into random shares and distributes them to different servers. No single server can reconstruct the original update, while the final aggregation result can be precisely reconstructed through linear combinations of all shares. Under the assumptions that "at least one server does not collude" and "at least two clients do not collude," rigorous client-level privacy protection is achieved.

Lossless Accuracy Guarantee. The linear homomorphism of additive secret sharing and the weighted summation operation of the FederatedAveraging algorithm are algebraically compatible. As derived in Section 3.1, the server-side share aggregation result is mathematically equivalent to standard FedAvg [1]. This equivalence fundamentally circumvents the "privacy budget–accuracy" trade-off inherent in DP.

DOI: 10.54254/2755-2721/2026.36451

4.2. Protocol details

Algorithm 1: Client Local Training

Input: n servers $s_1, . . , s_n$, mini-batch size B, number of local epochs E, learning rate η, dataset d, number of records l in dataset

1: /*Local training process */
2: Split d into a set $\mathscr{B}$ of batches with size B ;
3: **For** $i = 1$ to E **then**;
4: **For** batch $b \in \mathscr{B}$ **then**;
5: $W \leftarrow W - \eta \nabla \ell(W;b)$; //Gradient descent weight update
6: /*Secret splitting process*/
7: **For** each server $j=1$ to $n-1$ **then**;
8: W^j= create a random weight in the range U;
9: $W^n=(W-\sum_{j=1}^{n-1} W^j)$;
10: **For** each server $j=1$ to n **then**;
11: Send W^j and l to server s_j ;

Algorithm 2: Server-Side Aggregation

Input: Local dataset sizes ($l_1,l_2,..,l_m$) from m clients ($c_1,c_2,..,c_m$), Secret shares of model weights ($W^1_k,W^2_k,..,W^m_k$) from m clients

Output: Output: Aggregated share Cnt, Tset,BF

1: $L=\sum_{j=1}^{m} l_i$;
2: $\alpha_k=\sum_{j=1}^{m}(L_i/LW^i_k)$;
3: Randomly select a server $s_{i,i\in[1,n]}$ as the leader server;
4: Servers $\{s_{j,j\in[1,n]/i}\}$ Send α_k to the leader server s_i;

Algorithm 3: Leader Server Aggregation

Input: Number of servers n, aggregated shares ($\alpha_1, \alpha_2, . . , \alpha_{n-1}$)

1: $R=(\sum_{i=1}^{n}\alpha_i)$;
2: Broadcast R to all clients;

5. Experiments

5.1. Experimental setup

Experimental Environment: All experiments were conducted on a Linux server equipped with an AMD Ryzen7 processor and an NVIDIA GeForce RTX 2060 GPU, with 16GB RAM. The code was implemented in Python 3.9 using deep learning frameworks. Communication between clients and servers was simulated via local sockets to validate the correctness and performance overhead of the protocol. Each client and server ran as independent processes.

Dataset: The MNIST handwritten digit dataset [10] was used, comprising 60,000 training images and 10,000 test images, each a 28×28 grayscale image. The training data was partitioned among 100 clients in a pathological Non-IID manner: the data was first sorted by digit label, divided into 200 shards of size 300, and each of the 100 clients was assigned 2 shards. This partitioning ensures that most clients have samples from only two digit classes, thoroughly testing the algorithm's robustness

DOI: 10.54254/2755-2721/2026.36451

under extreme Non-IID conditions [1]. Federated Hyperparameters: Total clients K=100, fraction of clients per round C=0.1 (i.e., 10 clients randomly selected per round). Local epochs E=1, local batch size B=10, learning rate η=0.1, with default momentum settings. Number of servers n=3, with one randomly selected as the leader each round and the rest as auxiliary. Total global rounds = 30.

Baseline: The classic FederatedAveraging algorithm FedAvg [1] was chosen as the baseline. For a fair comparison, FedAvg and our scheme were run on the same dataset, with identical hyperparameters and hardware environment.

Evaluation Metrics: (1) Model Accuracy: Classification accuracy (%) evaluated on a unified test set (10,000 images) after each global round; (2) Efficiency Overhead: Average time per global round (seconds) to measure the computational and communication latency introduced by the protocol.

5.2. Experimental results

5.2.1. Accuracy comparison

Figure 2 presents the test accuracy curves of our scheme (FedShare) and FedAvg over 30 global rounds. It can be clearly observed from the diagram that the two accuracy curves almost completely overlap, and the accuracy difference of each round is within 0.03 percentage points, which is completely within the random fluctuation range of the experiment. As the iteration progresses, the accuracy of the two increases steadily. In the 30th round, FedShare reached 86.13 % and FedAvg was 86.12 %, with a difference of only 0.01 percentage points. All rounds in the detailed data showed highly consistent results.

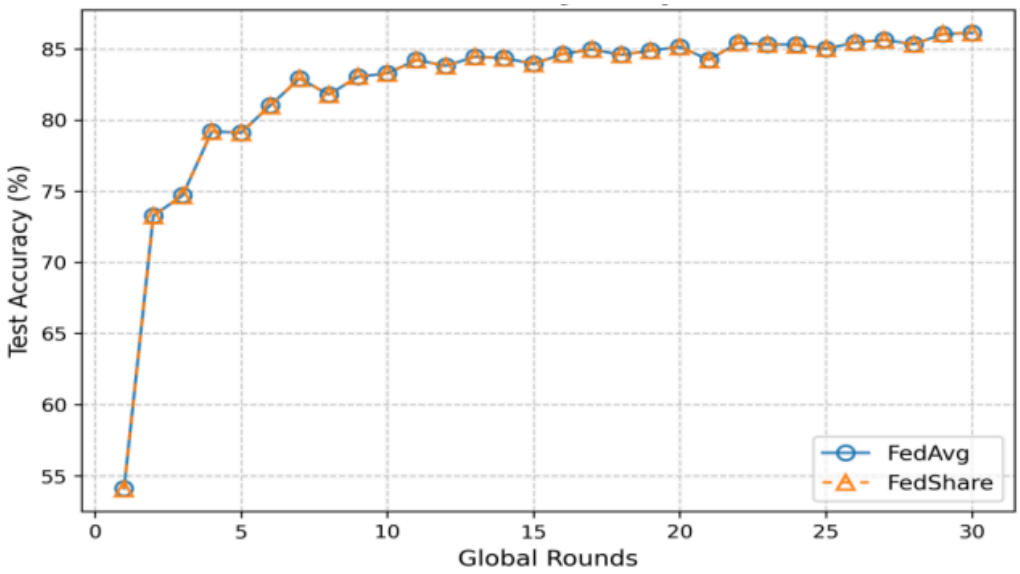

Figure 2. Comparison of classification accuracy between FedShare and FedAvg on MNIST (30 rounds)

The experimental results strongly show that the aggregation mechanism based on additive secret sharing is mathematically equivalent to the weighted average process of standard FedAvg, and does not introduce the accuracy deviation caused by share splitting, truncation or floating-point conversion. In other words, under the premise of strictly protecting the privacy of client model update, FedShare achieves the same model utility as the original federated learning algorithm, and successfully solves the core contradiction between privacy protection and model accuracy.

DOI: 10.54254/2755-2721/2026.36451

5.2.2. Efficiency comparison

Figure 3 shows the time consumption of each round of communication. FedAvg 's time consumption per round is stable at between 3.2 and 3.5 seconds, with an average of about 3.33 seconds. FedShare 's time-consuming per round is stable at between 3.3 and 3.8 seconds, with an average of about 3.52 seconds, basically the same as FedAvg, with a relative overhead of about 6 %. The first round takes slightly longer (about 7.34 seconds). This is because the algorithm needs to initialize the system environment, establish the connection between servers, and generate and distribute the secret sharing share for the first time when it runs for the first time, which belongs to the one-time cold start overhead. The round of data has been distinguished in Figure 2 .

After eliminating the first round of cold start data, FedShare 's average time per round is 3.46 seconds, which is only about 0.13 seconds more than FedAvg 's 3.33 seconds. In general, the overhead of this scheme mainly comes from the share transmission and addition aggregation between three servers (only $O(mn)$ traffic is required per round, where m = 10 is the number of participating clients and n = 3 is the number of servers). This overhead is much lower than the order of magnitude of homomorphic encryption and other schemes, which verifies that this scheme maintains a high degree of system efficiency while achieving strict privacy protection.

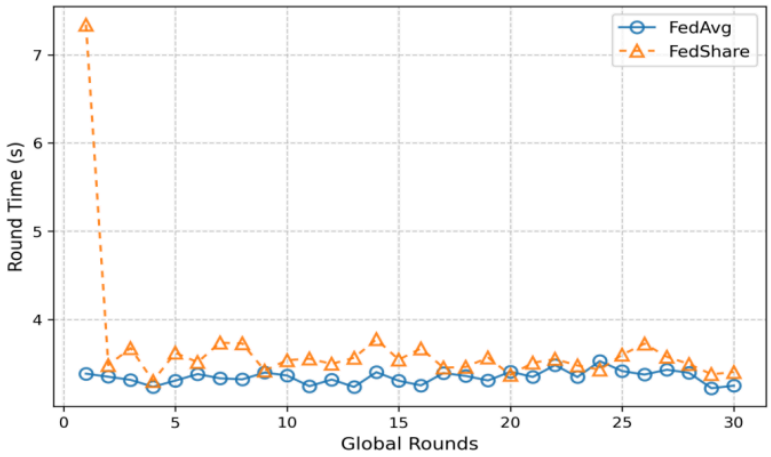

Figure 3. Comparison of per-round time between FedShare and FedAvg on MNIST (30 rounds)

Combining the accuracy and efficiency results, the experimental outcomes demonstrate that under the reasonable security assumptions of "at least one server does not collude" and "at least two clients do not collude," our scheme not only strictly protects clients' gradient privacy but also guarantees, through mathematical provability, that the aggregation result is exactly equivalent to standard FedAvg. Meanwhile, its computational and communication overhead remains within acceptable limits, offering a feasible solution for practical deployment of federated learning that achieves an excellent balance among privacy, accuracy, and efficiency.Partial data are shown in Table 1.

Table 1. Selected round accuracies and times

round	master_server	fedshare_accuracy	fedavg_accuracy	fedshare_time	fedavg_time
1	1	54.1	54.13	7.016543	3.112729
6	3	81.01	81.03	3.402333	3.166665

DOI: 10.54254/2755-2721/2026.36451

Table 1. (continued)

11	2	84.22	84.22	3.290119	3.147545
16	2	84.64	84.64	3.26051	3.206255
21	2	84.24	84.24	5.44531	3.464013
26	3	85.46	85.45	3.283747	3.551623

6. Conclusion

Aiming at the core contradiction between privacy protection and model accuracy and system efficiency in Federated Learning, this paper proposes a multi-server secure aggregation scheme based on additive secret sharing. The scheme uses the information-theoretic security of additive secret sharing to split the client model update into multiple random shares and distribute them to multiple non-collusion servers, and achieves strict client-level privacy protection under the reasonable assumptions of ' at least one server does not collude ' and ' at least two clients do not collude '. At the same time, using the linear homomorphism of the additive secret sharing, it is mathematically proved that the aggregation result of the scheme is completely equivalent to the standard FedAvg, which ensures that the model accuracy is lossless. The experimental results on MNIST dataset show that the proposed scheme achieves 86.13 % test accuracy after 30 rounds of global iteration, which is completely equal to FedAvg (86.12 %), and the extra time overhead per round is only 6 %, which verifies the effectiveness and efficiency of the scheme. Future work will expand the scheme to resist active attacks in malicious adversary models , explore the integration with differential privacy to provide quantifiable privacy protection, and verify the generalization ability of the scheme on larger data sets and more complex model architectures.

References

[1] McMahan, H. B., Moore, E., Ramage, D., Hampson, S., & Agüera y Arcas, B. (2017). Communication-efficient learning of deep networks from decentralized data. *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 1273–1282.
[2] (2024). A survey of security strategies in federated learning: Defending models, data, and privacy.
[3] Shamir, A. (1979). How to share a secret. *Communications of the ACM, 22*(11), 612–613.
[4] Blakley, G. R. (1979). Safeguarding cryptographic keys. *1979 International Workshop on Managing Requirements Knowledge (MARK)*, 313–318.
[5] Dwork, C., & Roth, A. (2014). The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science, 9*(3–4), 211–407.
[6] Phong, L. T., Aono, Y., Hayashi, T., Wang, L., & Moriai, S. (2018). Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Transactions on Information Forensics and Security, 13*(5), 1333–1345.
[7] Mohassel, P., & Zhang, Y. (2017). SecureML: A system for scalable privacy-preserving machine learning. *2017 IEEE Symposium on Security and Privacy (SP)*, 19–38.
[8] Mohassel, P., & Rindal, P. (2018). ABY3: A mixed protocol framework for machine learning. *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 35–52.
[9] Khojir, H. F., Alhadidi, D., Rouhani, S., & Mohammed, N. (2023). FedShare: Secure aggregation based on additive secret sharing in federated learning. *Proceedings of the 27th International Database Engineered Applications Symposium (IDEAS 2023)*, 25–33.
[10] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE, 86*(11), 2278–2324.