EAI Endorsed Transactions
on Scalable Information Systems
Research Article EAI.EU

An Efficient Privacy-Preserving Secure Aggregation Scheme for Federated Learning with Input Verification and Dropout Resistance

Zijun Guo[1*], Yuteng Sun[1], Xinyue Zhang[1] and Lingling Wu[1]

[1]College of Cryptography Engineering, Engineering University of People's Armed Police, Xi'an 710086, Shaanxi, China

Abstract

Federated learning, as a distributed machine learning paradigm, allows multiple participants to collaboratively train a shared model without sharing their local data. However, the increasing demand for privacy protection during data aggregation within distributed systems underscores the persistent challenge of ensuring both security and efficiency. Many existing Privacy-Preserving Machine Learning (PPML) schemes relying on homomorphic encryption introduce substantial computational overhead during aggregation, rendering them impractical for large-scale PPML applications involving resource-constrained participant devices. Moreover, device dropout events and data poisoning attacks perpetrated by malicious clients adversely affect the integrity of the aggregated results. To address these challenges, this paper proposes an efficient privacy-preserving secure aggregation scheme capable of tolerating participant dropout at arbitrary stages and securing data against both semi-honest and malicious participants. By integrating input verification protocols and applying gradient masking techniques, the scheme enhances its resilience against malicious attacks while ensuring user data privacy. Leveraging the additive homomorphic property of Shamir's secret sharing enables efficient global mask recovery, significantly optimizing the scheme's efficiency. Experimental results demonstrate that the proposed scheme significantly outperforms baseline methods in computational efficiency, communication overhead, and security robustness. By effectively balancing high privacy protection with practical feasibility, this scheme presents a promising solution for secure multi-party aggregation in large-scale distributed systems.

Keywords: Secure Aggregation, Privacy Preservation, Secret Sharing, Zero-Knowledge Proofs

Received on 15 September 2025 , accepted on 17 October 2025, published on 16 March 2026

doi: 10.4108/eetsis.11991

1. Introduction

In the contemporary digital era, the traditional centralized machine learning paradigm faces many challenges with the explosive growth of data volume and the increasing awareness of data privacy protection. Federated Learning (FL) [1], as an emerging distributed machine learning paradigm, solves the problem of training global models without being able to centralize the training data and can effectively address the dual challenges of privacy protection and model performance in the era of big data.

The core idea of federated learning is to achieve multi-party collaborative joint modeling by training models on local devices and sharing only model parameters instead of raw data. This framework not only effectively reduces the risk of privacy leakage caused by data aggregation in traditional centralized machine learning but also provides a feasible solution for cross-organizational and cross-domain data collaboration. In recent years, with the increasingly stringent data privacy protection regulations (e.g., GDPR [2]) and the popularization of edge computing devices [3], federated learning has shown great potential for application in healthcare, fintech, smart IoT, and other domains with stringent data privacy requirements [4][5][6]. However, the technique still faces a series of challenges such as communication efficiency [7], heterogeneity handling, and privacy-utility tradeoffs, the solution of which will drive federated learning in a more mature direction.

*Corresponding author. Email: g18833211087@163.com

Although federated learning reduces the privacy risk of raw data circulation through the parameter-sharing mechanism, recent studies have shown that the model parameters themselves may still expose sensitive information through reverse engineering [8] or membership inference attacks [9]. This contradiction has given rise to the research of Secure Aggregation (SA) techniques, whose core goal is to construct encrypted global model update channels, so that the server can only obtain the overall parameter updates after the aggregation of encrypted model updates from each client, and cannot trace the contribution value of a single client, thus protecting the client's local data.

Typically, secure aggregation schemes in federated learning scenarios often combine privacy-preserving techniques such as Homomorphic Encryption (HE), Differential Privacy (DP), and Secure Multi-Party Computation (MPC). For example, in a secure aggregation scheme constructed based on homomorphic encryption [10][11], the client uploads the encryption results of the training model parameters to a central server, which algebraically computes the set of protected gradient information uploaded by the user and returns the results, and the client carries out the decryption operation or the ensuing ciphertext computation. The scheme constructed in this way can guarantee information theory security, but faces high computational overhead, which is less effective in practical use and deployment. In secure aggregation schemes constructed using differential privacy, noise obeying a specific distribution is added to the gradient information before it is uploaded by the client to achieve privacy protection [12][13]. However, adding random noise can adversely affect the availability of data to some extent and also make the model convergence require more rounds, increasing the communication overhead. Therefore, utilizing MPC techniques such as secret sharing has a clear advantage in an increasing number of secure aggregation schemes [14], where the client of such schemes splits the secret information into multiple shares and distributes them to the rest of the different participants, and recovering the data requires cooperation from multiple parties, avoiding the risk of a single point of failure. In addition, secret sharing does not require encryption and decryption operations on the data, with less computational overhead, while no additional noise is introduced to ensure the accuracy and validity of the aggregation results.

Nowadays, a large number of secure aggregation protocols and systems applied in federated learning scenarios have been proposed [15][16][17][18], but based on the specificity of federated learning, the real-world application of secure aggregation still faces some challenges that are difficult to control: adversary collision attacks [19], massive users, high-dimensional input vectors, and unstable devices with unstable connectivity (e.g., some devices may drop out of the network before the end of the protocol [14]), the occurrence of these problems can affect the aggregation effectiveness of security aggregation protocols to varying degrees.

Input validation is a key technical means to ensure the effectiveness of global model aggregation, and the ability to perform correct and effective aggregation operations in the face of the presence of malicious users is another challenge in the actual deployment process. Poisoning attacks from malicious clients aim to impair the robustness of the federated learning system, disrupt and interfere with the normal operation of the federated learning system, and greatly reduce the usability of the trained models. Common attacks include model poisoning attacks (Byzantine attacks [20]) and backdoor attacks, where the former attacker disrupts the training process of the model by injecting malicious samples into the training data, thus degrading the model's performance. The latter attacker implants specific triggers (e.g., specific patterns or features) in the training data, so that the model performs well under normal inputs, but outputs the attacker's predetermined target results when the input contains specific triggers, which is highly stealthy.

A comprehensive and reliable federated learning security aggregation scheme needs to focus on the following aspects:

1 **Security:** In real-world applications, we need to consider poisoning attacks by malicious attackers to ensure that user privacy and model integrity, and usability can be protected despite the malicious collaboration of some participants.

2 **Efficiency:** The overall impact of the computational overhead and communication overhead involved in the execution of the scheme and the scalability of the scale of the number of participating users.

3 **Robustness:** This is primarily reflected in the ability to tolerate random dropouts of honest clients, ensuring secure vector aggregation even when a certain proportion of users go offline.

Overall, in this paper, we propose an efficient secure aggregation scheme in federated learning. Our scheme is based on the single mask mechanism to hide secret information and uses Shamir's secret-sharing approach to reconstruct the results of aggregation masks, which can satisfy the client's need for data aggregation without revealing private information. In comparison with existing work, our contributions are as follows:

1 Our scheme innovatively proposes a distributed mask correctness verification protocol that integrates it into the overall scheme flow, and introduces an input validation mechanism to resist poisoning attacks by malicious participants. This not only ensures the reliability of the aggregation results but also avoids a significant increase in communication burden.

2 Our scheme is highly tolerant to dropped clients, does not need to rely on trusted third parties, and can correctly execute the protocol process when some users are dropped, thus ensuring the robustness and continuity of the aggregation process.

3 The construction of our scheme is based on adding individual masks, and the addition and restoration of masks are simpler and more efficient, we utilize the additive homomorphism shared by Shamir's secret to reconstruct the result of mask aggregation instead of reconstructing them as one by one, which outperforms most of the traditional schemes in terms of computational efficiency and communication efficiency.

This paper is organized as follows: In Section 2, we mainly review recent related similar works and briefly introduce the preparatory cryptography knowledge related to this paper. The scheme construction is then elaborated in Section 3, followed by security proofs in Section 4 and relevant experimental designs in Section 5. Finally, in Section 6, we give the conclusions.

2. Related Work and Backgrounds

In this section, a review of existing research on secure aggregation schemes in federated learning is presented, followed by a brief overview of the core cryptographic tools involved in our scheme.

2.1. Related Work

In this section, we review the secure aggregation schemes in the context of the current mainstream technologies and compare and analyze their core mechanisms.

In conventional federated learning architectures, a single aggregation server is responsible for summarizing the local training model updates uploaded by each client as a way to train to get the global model. However, this model carries the risk of data leakage, which is since the shared update information may be exploited maliciously, and such data leakage-based attacks are known as inference attacks [21][22]. Even if the server is semi-honest, it may infer sensitive information about private training data by analyzing local model updates uploaded by clients. Secure aggregation techniques are widely used as a common defense against inference attacks.

Homomorphic encryption provides a generalized solution for privacy protection in federated learning scenarios. In the solution proposed by Lessage et al. [23] to enhance privacy protection in medical image analysis, model weights in federated learning are encrypted by full homomorphic encryption (FHE) to ensure a secure state of the data in the transmission and aggregation process. However, the encryption, decryption, and serialization operations involved in the solution are time-consuming, and the global model security will be compromised if a single secret key is leaked by a client. Hosseini et al. [24] proposed a secure aggregation method based on multi-party homomorphic encryption (MPHE), and the solution introduces a secret sharing mechanism in the setup phase so that decryption can be done by some clients, which can accommodate the random dropping of clients in federated learning. The scenario of random client dropouts and partial participation in federated learning, but the overhead of this scheme is still high in large-scale client (e.g., N>1000) scenarios.

Liu et al. [25] proposed an efficient secure aggregation scheme SASH based on a seeded homomorphic pseudo-random generator (SHPRG), which takes advantage of the homomorphic nature of SHPRG to simplify the process of mask generation and elimination, and the design of separating the computation process from the communication significantly optimizes the efficiency of computation and communication in federated learning scenarios. However, it does not incorporate security mechanisms such as input validation within the entire process, which leads to its inability to effectively defend against model poisoning attacks initiated by malicious clients. Specifically, an attacker can contaminate the global model parameters without triggering anomaly detection by forging local model updates or manipulating mask seeds, which can lead to degradation of model performance or trigger backdoor injection risks.

Verifynet [26] innovatively combines privacy protection with verifiability, effectively protects users' private data through a double-masking protocol, and tolerates some users' dropouts during training, with good performance on real data. However, this scheme relies on a trusted third party to generate public-private key pairs for all the clients participating in the training, leading to a single point of failure risk in the system. Similarly, Kadhe et al. [27] proposed FastShare, a multi-secret sharing scheme based on the Fast Fourier Transform (FFT), which can resist adaptive attackers and at the same time has a certain degree of robustness to the user dropout problem, the scheme requires a trusted key distribution center, which may be difficult to implement in some practical application scenarios, limiting its fully decentralized environment This may be difficult to fulfill in some practical application scenarios, limiting its application in a fully decentralized environment. In addition, further security enhancement mechanisms are needed to face the possible existence of malicious attackers.

In addition, other studies such as [28][29] combine federated learning with differential privacy techniques. The scheme proposed by Han et al. [28] adds different levels of noise according to the importance coefficients of the model parameters to reduce the impact of noise on the performance of the global model by introducing a differential noise addition mechanism. The scheme of Zhang et al. [29] effectively enhances data privacy protection in federated learning through a two-stage differential privacy protection mechanism, which effectively enhances data privacy protection in federated learning. The first stage perturbs the original feature data through a randomized response mechanism, and the second stage adds noise to the local model through an edge server to further protect model privacy.

2.2. Preliminaries

In this section, we briefly introduce the related concepts of LWE hard problems, Shamir secret sharing, and zero-knowledge proofs.

2.2.1 Learning with Errors

In this scheme used for mask generation, we use a technique based on the LWE hard problem, the LWE problem [30] is a difficult problem that is widely used in cryptography, and its security is based on the complexity of solving the problem. It can be used to construct a variety of cryptographic schemes. Arithmetic is relatively easy to implement, computationally efficient, and can meet the requirements of cryptographic algorithm efficiency in practical applications. The LWE problem is usually defined over a finite field $\mathbb{F}_q$ (of order q, usually denoted as $GF(q)$). Let a secret vector $s \in$

$\mathbb{F}_q^n$ be given. An LWE sample consist of a pair (a, b), where a is uniformly randomly chosen from $\mathbb{F}_q^n$, and b is obtained by computing $a \cdot s + \mathrm{e}$, where $a \cdot s$ denotes the inner product over $\mathbb{F}_q$, and e is is an error term sampled from the appropriate error distribution over $\mathbb{F}_q$.

There are usually two versions of the LWE problem, i.e., the LWE decision problem and the LWE search problem, which are based on the same mathematical model, and in some cases, it is possible to reduce the search problem to a decision problem, i.e., to solve the search problem by solving the decision problem several times. What is needed in this paper's scheme is the difficulty of the LWE decision problem, i.e., it is difficult to determine whether a given sample (a, b) it is an LWE sample pair or randomly generated from $\mathbb{F}_q^n \times \mathbb{F}_q$.

2.2.2 Shamir Secret Sharing Scheme and its Homomorphic Properties

The Shamir secret-sharing scheme [31] is a cryptographic technique for the decentralized storage of secrets. The scheme prevents the secret information from being too centralized by splitting a secret value S into n parts and storing them separately by n participants. Meanwhile, when the threshold value is $t(1 \le t \le n)$, i.e., at least t secret shares are required to reconstruct the secret value S, any subset consisting of less than t secret shares cannot be reconstructed. Such a scheme is called a (t, n) threshold scheme.

Specifically, for a (t, n) threshold scheme, in a finite domain $\mathbb{F}_q$, where the prime q satisfies $0 < t \le n < q$, the secret value S of the secret holder u_i is partitioned into $\{S^1, \ldots, S^n\}$, each of which is kept by a participant $u_j (j \in (1, n))$ respectively. Its mathematical expression is as follows.

(1) Secret Splitting and Sharing: The secret holder u_i randomly selects $t-1$ positive integers $a_1, \ldots, a_{t-1}$ in $\mathbb{F}_q$ and constructs a polynomial of degree $t-1$, i.e., $f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{t-1} x^{t-1} \bmod q$, where $a_0 = S$. Subsequently, u_i randomly selects n points $x_1, \ldots, x_n$ and computes $\{x_j, f(x_j)\}$ where $j \in \{1,2, \ldots, n\}$, and sends them to each of the n participants.

(2) Secret Reconstruction: For any t participants holding $\{x_j, f(x_j)\}$, the secret holder u_i can use Lagrange interpolation to compute the coefficients $a_1, \ldots, a_{t-1}$ of the polynomial $f(x)$, and subsequently, obtain the secret value $S = f(0) = a_0$.

For simplicity, we use $hamirSS(t, n, S)$ in the subsequent scheme to denote the n secret shares $\{x_j, f(x_j)\}$ generated from the secret value S. The reconstruction function is denoted as $\mathcal{F}_{rec}$ and is used to reconstruct the secret value S based on any t shared shares. Furthermore, the Shamir secret-sharing scheme has an additive homomorphism. For a (t, n)-threshold scheme, the multiple secret holders are denoted as the set $\mathcal{U}$, $\mathcal{U} = \{u_1, \ldots, u_n\}$, u_i holds the secret value S_i,where $i \in \{1, \ldots, n\}$.

During the secret splitting and sharing phase, the secret holder u_1 generates a $t-1$ degree polynomial $f(x) = a_0 + \sum_{i=1}^{t-1} a_i x^i$, where $a_0 = S_1$, to share the secret value S_1 among n participants. The shares $\{x_i, f(x_i)\}$ are calculated and sent to the corresponding participants. Similarly, another secret holder u_2 generates a $t-1$ degree polynomial $g(x) = b_0 + \sum_{i=1}^{t-1} b_i x^i$, where $b_0 = S_2$, to share the secret value S_2 among the n participants. The shares $\{x_i, g(x_i)\}$ are calculated and distributed accordingly.

Furthermore, the additive homomorphism of Shamir's Secret Sharing scheme is introduced as follows: In the secret reconstruction phase, participant. u_3 receives the secret shares from u_1 and u_2 and performs an addition operation to obtain $\{x_3, f(x_3) + g(x_3)\}$. Subsequently, u_3 collaborates with any $t-1$other participants to use the sum of t shares to reconstruct the secret value $S_1 + S_2$ using Lagrange interpolation. This process allows the reconstruction of the sum of the secret values from the sum of the secret shares, thereby achieving the additive operation of multiple secrets.

2.2.3 Zero-Knowledge Proofs

A commitment scheme in cryptography is a two-stage interaction protocol involving two parties, who are the committing party and the receiving party. Pedersen commitment [32] is a homomorphic commitment protocol that satisfies perfect hiding and computational binding, and its security is based on the discrete logarithm assumption. G is a q-order cyclic multiplicative group, and g, h are the generators of G. The Pedersen commitment of the committing party concerning the message vector m is expressed as $C = g^m h^r \bmod q$, where the random number r is the blinding factor, and the commitment is denoted in the scheme as $\mathrm{com} = \mathrm{Commit}(m; r)$. In the zero-knowledge proof part, we utilize Bulletproofs [33], which are currently recognized as the most efficient proofs of zero-knowledge ranges.

Currently, zero-knowledge proofs are widely used in cryptography, and by using zero-knowledge proofs, the prover $\mathcal{P}$ can convince the verifier $\mathcal{V}$ that he or she knows or possesses a certain piece of information without providing any useful information to the verifier $\mathcal{V}$. At the end of the proof process, the verifier $\mathcal{V}$ only obtains the information that "the prover $\mathcal{P}$ possesses this knowledge" without obtaining any relevant information about the knowledge itself. Zero-knowledge proofs can be categorized into interactive and non-interactive ones. Interactive zero-knowledge proof protocols include multiple rounds of interactions between the prover and the verifier, whereas non-interactive zero-knowledge proofs require only a single piece of proof information from the prover.

3. Our Scheme

In this section, we propose a scheme that realizes the need for multi-party secure aggregation services without a trusted third-party server. Briefly, each participant generates a mask based on the Learning with Errors (LWE) hard problem to hide the user's local private data, which can be regarded as a kind of encryption to prevent data leakage, and realizes the data aggregation based on the completion of the security

authentication through multiple rounds of interactions. Figure 1 highly summarizes the general flow of this scheme.

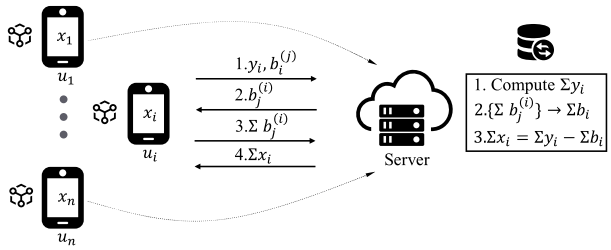

Figure 1. The overall workflow of our scheme

3.1. Threat Model

This system involves two parties: the clients and the server. The client consists of a large number of users holding private data, training local models and uploading parameters, while the server is responsible for aggregating and reconstructing the collected data. In this paper we identify the server side as honest but curious, i.e., it will strictly follow the protocol process but will try to infer the model parameters uploaded by the users. We consider two main types of threat-touching models based on the client's characteristics:

Honest but curious model: In this type of model, all clients are recognized as semi-honest; they do not initiate attacks or deliberately disrupt the protocol process, and complete the steps according to the rules set by the protocol. However, they are curious about the private data of other users and will use the legitimate information obtained from interactions with the server or other clients to perform inferential analysis. Therefore, in this scheme, we take the way of adding mask vectors to protect the privacy of each client's model data.

Malicious model: In the malicious model, there are some malicious attackers in the clients, who tend to use malicious means to disrupt the normal aggregation process of the protocol, and we consider data poisoning attacks by malicious participants [34][35]. Specifically, some clients submit vector information that does not match the real results, and these vectors tend to have too large values or deviate too much from the direction of the global aggregation model, which in turn affects the validity and usability of the aggregation results. Therefore, we introduce an input validation protocol in our scheme to check and validate the vector data uploaded by clients, stripping out malicious forged data, and thus resisting poisoning attacks.

In addition, under both threat models, we also take into account the client's dropout scenario, which will lose part of the user's critical information data and thus adversely affect the recovery of aggregation results. Therefore, the scheme should have a certain tolerance for user dropouts to ensure that the protocol process is carried out effectively.

3.2. Masking Protocol

In this paper, an innovative masking protocol is used to reduce the amount of communication between the client and the server. The core process of this protocol can be summarized in two steps, i.e., encrypted transmission of gradients and mask aggregation transmission. In gradient encryption and transmission, each participating training user randomly generates a one-time mask based on some rule, whose size is the same as the size of the local gradient vector, masks and hides its gradient information with the mask and sends the encrypted gradient information to the server side. In mask aggregation and transmission, each user generates its mask vector locally and subsequently integrates it with the gradient vector and sends it to the server side. The server will receive the mask gradient and perform the integration operation to prepare for the final recovery of the aggregated information.

To realize this, a matrix $A \in \mathbb{F}_q^{m \times n}$ is first generated as a common parameter shared by all users, and the matrix A is generated from m vectors randomly selected from the finite field $\mathbb{F}_q^n$, which are combined to obtain. Then each client i generates the secret vector. $s_i \in \mathbb{F}_q^n$ and the error vector $e_i \in \mathbb{F}_q^m$, both taken from the same distribution $\mathcal{X}$, and then the client obtains the mask vector b_i by computing $b_i = As_i + e_i$. We know that an LWE sample consists of a pair (a, b), and then (A, b) represents the set of m LWE samples. According to the properties of the LWE decision problem, the vector b_i generated by this method is statistically indistinguishable from the vector consisting of randomly selected elements from $\mathbb{F}_q$. Equation (1) indicates that the encrypted gradient vector y_i can be obtained after encrypting the gradient using the randomly generated mask vector:

$$y_i = x_i + b_i \tag{1}$$

In practice, each user can upload y_i to the server with confidence because the mask vector as well as the privacy information of the original gradient vector is masked, and as long as the user does not disclose the locally generated mask vector, the gradient vector will not be easily disclosed. After receiving y_i from all users, the server simply performs vector addition to get y_{sum}:

$$y_{sum} = x_{sum} + b_{sum} \tag{2}$$

In Equation (2) y_{sum}, x_{sum} and b_{sum} represent the result of summing the encrypted gradient vector, local gradient vector, and mask vector of the N users who participated in the training and did not drop out respectively, and this operation needs to be done on the server side. The mask vector sum b_{sum} is eliminated in a later step and the gradient vector aggregation result is returned to the user.

3.3 Vector Aggregation

Similar to many secure data aggregation schemes, this proposal also involves two types of key participants: First, there are the clients who hold private data models locally, denoted as $u_i \in \mathcal{U}$. Second, there is a central server, denoted as $\mathcal{S}$, which aims to aggregate inputs from n clients. Both types of players work together to ensure that the security and privacy of the data during the aggregation process are properly protected. Our scheme strives to achieve secure aggregation of client secret vectors $x_0 \ldots x_{n-1}$, with the core goal of accomplishing this process without revealing any

private data. To this end, each step in the scheme is carefully designed to ensure that the security of the data is maximized. In particular, by interspersing authentication interactions between the client and the server, we provide double security for the aggregation and recovery process of masks. This mechanism not only ensures the correctness of the aggregated masks, but also strictly guarantees that the recovered aggregated masks are the same as the summation results of each user's private local masks, thus realizing the accurate aggregation of data while protecting user privacy. We provide the complete detailed protocol flow in Figure 2, where additional steps involving authentication have been marked in red.

Participants: A set of clients $\mathcal{U}$ and a central server $\mathcal{S}$.

Public inputs: The total number of clients $n = |\mathcal{U}|$, the finite field $\mathbb{F}_q$, the input vector length ℓ, the input domain $\mathbb{R}^\ell$, the threshold $t < n$, the reconstructing function $\mathcal{F}_{rec}$ and the signature function $\mathcal{F}_{sig}$.

Client inputs: Each participated client's private vector $x_i \in \mathbb{R}^\ell$.

Sever outputs: The aggregated results of online clients' models $\sum x_i \in \mathbb{R}$.

Round 1:Commitments

1. Client i generates keypair $(sk_i, pk_i) \leftarrow KeyGen(1^\lambda)$ and sends pk_i to the server. And then executes the first step of the distributed mask correctness protocol.
2. The server commits to the public key pk_i using a Merkle tree and sends the root hashes h_r to the corresponding client. And then executes the second step of the distributed mask correctness protocol.
3. The server sends $\{j, pk_j, \pi_j\}_{j\in\mathcal{U}}$ to client i, where π_j is Merkle inclusion proofs in regard to h_r.
4. Client i aborts if the Merkle inclusion proofs fail to be verified.

Round 2: Mask sharing

5. Each client $u_1 \in \mathcal{U}_1$: ($\mathcal{U}_1$ denotes the set of online clients after Round 1)
   a) Generates a secret vector $s_i \in \mathbb{F}_q^n$ and computes its mask vector $b_i = As_i + e_i$.
   b) Computes a set of shares $[b_i] = \left\{b_i^{(j)}\right\}_{j\in\mathcal{U}_1} = ShamirSS(t, n, b_i)$.
   c) Sends the message $m_j = (j, a_{i,j})$ for each $j \in \mathcal{U}_1$to the server, where $\mathrm{Enc}\left(k_{i,j}, i||j||b_i^{(j)}\right) \rightarrow a_{i,j}$ for $k_{i,j} = KA.Agree(sk_i, pk_j)$.
6. The server aborts if it receives fewer than t messages from each client. Otherwise, it sends message $m_j = (j, a_{i,j})$ to the corresponding client j. The set of clients whose message received by the server is denoted by $\mathcal{U}_2$.

Round 3: Masking

7. Each client $u_i \in \mathcal{U}_2$:
   a) Computes the input masked by b_i, i.e., $y_i = x_i + b_i$.
   b) Forms signature $\mathcal{F}_{sig}(y_i, sk_i) \rightarrow \sigma_i$ to the server $\mathcal{S}$.
   c) Forms commitments $\mathrm{Com}(b_i, r_i) \rightarrow com_{b_i}$ and $\mathrm{Com}(x_i) \rightarrow com_{x_i}$ to its key an input.
   d) Computes proofs $\pi^{0\le x_i\le t}, \pi^{dir}, \pi^{valid(x_i)}$ of smallness, direction and validity.
   e) Executes the third step of the distributed mask correctness protocol to form K_i.
   f) Sends packed messages $\mathrm{M} \rightarrow \{y_i, \sigma_i, com_{b_i}, com_{x_i}, K_i, \pi^{0\le x_i\le t}, \pi^{dir}, \pi^{valid(x_i)}\}$ to the server $\mathcal{S}$.

Round 4: Verifying

8. The server aborts if it receives fewer than t packed messages from each client. Otherwise, The set of clients whose packed message is received by the server is denoted by $\mathcal{U}_3$. If $\mathcal{F}_{vrfy}(y_i, pk_i, \sigma_i) = 0$, remove u_i from $\mathcal{U}_3$. And then executes the fourth step of the key verification protocol.
9. Each online client i executes the fifth step of the distributed mask correctness protocol. Then send m_i and a_i to the server.
10. The server aborts if it receives fewer than t responses. It verify all received proofs $\pi^{0\le x_i\le t}, \pi^{dir}, \pi^{valid(x_i)}$ and aborts if any of them fails. Otherwise, it sends the set of online client names $\{j\}$ (denoted by $\mathcal{U}_4$) to clients.

Round 5: Unmasking

11. Each client $u_i \in \mathcal{U}_4$:
    a) Checks that if $|\mathcal{U}_4| < t$, aborts.
    b) Computes the sum of shares of $b_j^{(i)}$ from the clients $u_j \in \mathcal{U}_2$, i.e., $b_g^{(i)} = \sum_{j\in\mathcal{U}_2} b_j^{(i)}$.
    c) Sends $b_g^{(i)}$ with the signature $\mathcal{F}_{sig}(b_g^{(i)}, sk_i) \rightarrow \sigma_i^2$ to the server $\mathcal{S}$.
12. Server $\mathcal{S}$:
    a) Receive $b_g^{(i)}$from the clients (This set of clients denote with$\mathcal{U}_5$).If $\mathcal{F}_{vrfy}\left(b_g^{(i)}, pk_i, \sigma_i^2\right) = 0$, remove client i from $\mathcal{U}_5$. Proceed until $|\mathcal{U}_5| > t$.
    b) Reconstructs the aggregated mask b_g for unmasking, i.e., $\mathcal{F}_{rec}\left(\left\{b_g^{(i)}\right\}_{u_i\in\mathcal{U}_4}\right) \rightarrow b_g \mod p$.
    c) Executes the final step of the distributed mask correctness protocol.
    d) Computes and outputs the aggregated inputs $\sum x_i = \sum y_i - b_g$.

Figure 2. A detailed description of the scheme

As described in Section 3.1, to protect the client's model, we adopt the approach of using a one-time pad to mask the client's local model before uploading it to the central server. At the same time, the mask vector b_i used by each user cannot be an identical value, considering that there exists the possibility that an adversary can easily reconstruct the global model to infer the local data uploaded by the client. The final model vector uploaded by the user is represented in the form of equation (1) as part of the content of the message sent by the client to the server during the process.

The aggregated encrypted gradient vector $y_{sum} = \sum_{u_i \in \mathcal{U}} y_i$ can be obtained by the server by performing a simple summation operation on the whole y_i received, and if it is assumed that the server then obtains the aggregated mask vector $b_g = \sum_{u_i \in \mathcal{U}} b_i$, it can be straightforward to recover the aggregated gradient vector $x_{sum} = \sum_{u_i \in \mathcal{U}} x_i$. The most direct way to get b_g is for individual clients to upload their respective mask vectors b_i along with y_i to the server, but a semi-honest server will try to infer the data uploaded by the user and get x_i easily, so this method is not secure.

Therefore, briefly in this scheme we employ Shamir's secret sharing scheme to split b_i into multiple shares, and the client uploads the data with the summation share of b_i. This approach prevents the server from directly reconstructing b_i, effectively protecting user privacy. Meanwhile, it enhances the overall robustness of the system to some extent, allowing it to tolerate client dropouts at any stage of the aggregation process due to various reasons. As long as the number of online users in the final recovery phase is greater than the threshold value t, the server can recover b_g based on the share of the summation secret shared by the users in the previous phase.

As can be seen from Section 3.1, client u_i generates the secret vector s_i based on the LWE hard problem, which in turn computes the mask vector b_i, and subsequently generates y_i based on b_i and x_i. At the same time, each client u_i partitions b_i into a series of secret shares by using the Shamir threshold scheme, i.e., $[b_i] = Shamir(t, n, b_i)$. Each share corresponds to an online client so that the server can send the share to the corresponding client after data upload. Assuming that there are n clients participating in the aggregation, correspondingly, each client u_i divides b_i into n shares before uploading the data, i.e., $[b_i] = \left\{b_i^{(1)}, \ldots b_i^{(n-1)}\right\}$. After a round of forwarding from the server, each online client u_i receives n secret shares, denoted as the set $\mathcal{M} = \{b_1^{(i)}, \ldots b_{n-1}^{(i)}\}$. At this point, the online users are denoted as the set $\mathcal{U}'$, considering the case of users dropping out, there are $|\mathcal{U}'| \leq |\mathcal{M}|$. After completing a series of related verification operations, u_i takes out the share of $\mathcal{M}$ that corresponds to the serial number of the user in $\mathcal{U}'$ and sums it up to get $b_g^{(i)} = \sum_{u_j \in \mathcal{U}'} b_j^{(i)}$, and sends $b_g^{(i)}$ and the corresponding signatures to the server side. The server verifies the received signatures and gets the set of verified users $\mathcal{U}''$ until $|\mathcal{U}''| \geq t$ is satisfied and then continues the execution. At this point, the server side has the set of summed mask vector shares $\mathcal{M}' = \left\{b_g^{(i)}\right\}_{u_i \in \mathcal{U}''}$, and secret reconstruction using the additive homomorphism of Shamir's secret sharing scheme yields the aggregated mask vector b_g, i.e., $b_g = \sum b_i \ (u_i \in \mathcal{U}')$. Finally, x_{sum} can be obtained after the elimination mask operation at the server side.

3.4 Input Validation

In this section, we will focus on the input validation part involved in the scheme. Consider that during the training process, a malicious client under the control of an attacker can upload malicious data during the input phase, e.g., in the case of sharing random vectors that are inconsistent with the locally generated masks to influence the global aggregation results or manipulating the direction of the vectors uploaded with the local model updates to cause the aggregation results to tend to a random direction that the attacker expects. All these scenarios can undermine the usability and validity of the aggregation results to some extent. For this reason, we provide a method for the server to check whether the model parameters uploaded by the client satisfy some specific criteria, which include proof of the correctness of the client's distributed aggregation mask and proof of the validity of the input vectors.

Distributed Aggregate Mask Correctness Proofs

Regarding the part of the aggregation mask correctness proof, we adopt an interactive proof approach in our scheme, in which we integrate the specific steps in multiple rounds of interactions between the client and the server throughout the protocol. Figure 3 illustrates the entire verification interaction flow.

The client's input will not expose the user's private data after mask obfuscation, and the server cannot know the client's private data while it cannot determine whether the client's uploaded mask share corresponds to the locally generated mask. Therefore to verify:

$$b_g = \sum b_i \tag{3}$$

the clients are required to jointly provide proof, which is to ensure that the aggregated input result X after server recovery is consistent with the actual aggregated input result $\sum x_i$ from the client. Throughout the verification process, each client does not need to communicate with other clients, but only needs to provide a partial proof π^{b_i}. Together, this proof information can prove the correctness of the aggregation mask. Specifically, each client u_i commits to the mask vector b_i utilizing the random value r_i. Here, the choice is made to establish a Pedersen commitment, i.e.:

$$C_i = g^{b_i} h^{r_i} \tag{4}$$

and then send it to the server along with the rest of the validation information. After several rounds of interaction protocols, the server will eventually have explicit knowledge of the aggregated mask vector b_g and the product of all the commitments $\prod C_i$, which can be regarded as a commitment to b_g. In the whole process, the server acts as a verifier, and the correctness of the aggregated input results after

eliminating the mask in the recovery phase can be guaranteed after successfully passing the verification protocol.

Global parameters:

A cyclic group G with generators g_1 and g_2.

Participant Inputs:

- **Client i: local mask** b_i, randomness r_i, and commitment $C_i = \text{Commit}(b_i; r_i)$.
- **Server:** Vector b_g, and client commitments$\{C_i\}$.

Protocol Execution:

1. Each client i randomly selects a_i and sends $g_{2_i} = {g_2}^{a_i}$ to the server.
2. The server generates a random challenge e. For each client i, it computes a commitment $com_{i,chl} = g_1^e \, {g_2}^{d_i}$ with randomness d_i, then sends $com_{i,chl}$ to client i.
3. Each client i generates $k_i \in |G|$ and sends $K_i = {g_2}^{k_i}$ to the server.
4. The server reveals the challenge e and randomness d_i to client i.
5. Client i checks that $com_{i,chl} = g_1^e \, {g_2}^{d_i}$. If not, it aborts. Otherwise it computes $m_i = r_i e + k_i$ and sends m_i and a_i to the server.
6. The server checks that $g_{2_i} = {g_2}^{a_i}$. If pass, it Computes aggregated commitment $C = \prod C_i / g_1^{b_g}$ and aggregated response $m = \sum m_i$. Then it checks that ${g_2}^m = C^e \prod K_i$. Outputs 1 if all checks pass, otherwise 0.

Figure 3. Distributed mask correctness protocol

Zero-Knowledge Proofs of Norm Bounds

The integrity of model aggregation is often used to refer to whether the aggregated model is properly constructed, and in general, the integrity of model construction is susceptible to data poisoning attacks [36]. If the model parameters submitted by users in the federated learning paradigm are vectorized and analyzed, the model vectors of honest clients, which account for the majority, are statistically consistent (both in terms of vector direction as well as scale size), while the model vectors submitted by malicious clients deviate significantly [37]. For this part, we use Non-Interactive Zero Knowledge Proof (NIZK).

In the scheme, each client needs to provide a zero-knowledge proof about the input vector paradigm bounds (denoted as B) to ensure that their input vectors y_i about the update are correctly formed. In the notation of Camenisch and Stadler [38], a client needs to provide the server with proofs with the following relation:

$$NIZK(x_i, b_i)\{c_i = (g_1^{x_i} g_2^{b_i}, g_1^{b_i}) \wedge \left|\left|x_i\right|\right|_p < B\} \quad (5)$$

For zero-knowledge proof systems applying such proofs there are ACORN [39] and ROFL [40], where we have made a similar setup. In the scheme, we use the criteria of L_∞ norm and L_2 norm. under the L_∞ norm criterion, the client only needs to prove to the server that each entry of the input vector x_i lies in the range of the interval [0,B), as a way of guaranteeing that the vector is correctly generated; under the L_2 norm criterion, the client needs to prove that the input vector x_i satisfies:

$$\sum_{i=1}^{l} x_i^2 < B_{L_2}^2 \quad (6)$$

Specifically, the client provides a Pedersen commitment on x_i, namely:

$$C_i = Com(x_i^2, r_i) = g_1^{x_i^2} g_2^{r_i} \quad (7)$$

and provides a proof that every entry about x_i lies within the interval $[0, B_{L_2}^2)$.

4. Security Analysis

In this section, we provide a detailed analysis for the security of the protocol, assuming a scenario where the environment is an honest but curious client with the presence of a malicious adversary.

Theorem 1: Under a given secure aggregation protocol, a participant's private data remains private during the execution of the protocol, i.e., no participant can access the inputs of other participants.

PROOF: Suppose there are n participants, the set of participants is denoted as $\mathcal{U}$ and each participant has an input vector x_i, where $i \in [0, n-1]$. We need to prove that no participant can infer the inputs of other participants during the execution of the protocol, i.e., for any two participants u_i and $u_j (i \neq j)$, assuming that the input of P_i is x_i, u_j cannot infer x_i from the aggregation result. Analyzing the overall flow of the protocol, firstly, the protocol protects the input privacy by introducing a mask vector b_i in the input vector x_i. input privacy. Specifically, the client locally generates the mask vector b_i and combines the input vector x_i with the mask vector to generate the encrypted input $y_i = x_i + b_i$. At this time, the input of u_i is masked, and since the mask vector b_i is generated based on the LWE hard problem, even though the semi-honest client $u_j \in \mathcal{U} \backslash \{u_i\}$ or the semi-honest aggregation server can obtain y_i, they cannot be able to know b_i because it not being able to know b_i and thus x_i. In addition, this scheme uses the Shamir secret-sharing mechanism to slice the mask vector. All the shared information received by the server is the aggregation result locally computed by each client, and it is not possible to directly know the specific value of individual mask vectors. In the reconstruction phase, the server can only obtain the aggregated mask value b_g, thus realizing the recovery of

the aggregation result Σx_i without directly accessing the input of each participant. Meanwhile, the protocol introduces interactive and non-interactive zero-knowledge proof mechanisms to ensure that the server can verify the validity of the input vector x_i and the correctness of the aggregated mask vector b_g without knowing the participants' inputs. The zero-knowledge proof mechanism ensures the consistency of each participant's inputs with the final aggregation result without leaking any additional information to the third party. In summary, the design of the mask vector is based on the LWE hard problem, which ensures input privacy; the secret sharing mechanism prevents single-point leakage; and the zero-knowledge proof mechanism verifies the correctness of the inputs and the aggregation result. As a result, a single participant cannot recover the masks or input vectors of other participants through the obtained information during the protocol execution process, so the privacy of participants' data can be better ensured throughout the protocol process.

Theorem 2: Under the given secure aggregation protocol, even if multiple malicious participants collude to launch an attack, they are unable to obtain or tamper with the private data of other participants.

Proof: Assume that there are n participants, t of which are malicious, and the set of colluding attackers is denoted as $T \subseteq \{P_1, P_2, \ldots, P_n\}$, where $|T| = t$. Each participant, P_i, has an input x_i. We need to show that even if the malicious participants conspire to carry out the attack, they are not able to infer or tamper with the private data of other participants.

According to the protocol, the encrypted input of each participant is denoted as $y_i = x_i + b_i$, where b_i is a mask vector generated based on the LWE hard problem. Due to the indistinguishability of the LWE decision problem, the mask vector b_i is statistically indistinguishable from a random vector. A malicious participant $P_j \in T$ has access to its own mask b_j but not to the masks b_k or real inputs $x_k (k \neq j)$ of other participants. Therefore, the malicious participant cannot directly infer the inputs of other participants through local information. Without direct access to the other participants' encrypted inputs, the amount of information that a colluding attacker can obtain is limited. Suppose the colluding attacker attempts to recover the private data of other participants by manipulating the aggregation result $y = \Sigma_{i=1}^{n} y_i$. Since each input x_i is masked by an independent random mask b_i, the aggregation result y is actually $\Sigma_{i=1}^{n}(x_i + b_i)$. The protocol achieves mask aggregation and recovery through additive homomorphisms shared by Shamir secrets. Specifically, each client divides the mask b_i into multiple secret shares and distributes these shares to other participants. Only when the number of online participants exceeds a threshold value, the server can reconstruct the aggregated mask $\Sigma_{i=1}^{n} b_i$ via Lagrangian interpolation. Since a single mask b_i is partitioned into multiple shares and an attacker does not have access to all of the necessary shares, they are unable to recover a single mask or input. Zero Knowledge Proof (ZKP) is introduced in the protocol to verify the correctness of the aggregation result while ensuring that no additional information about the inputs is disclosed. Even if multiple malicious participants collude, they can only verify the aggregation results related to their own inputs and cannot infer the private data of other participants through the proof process.

During the communication process, we employ a series of encryption algorithms to ensure the confidentiality of the transmitted data, effectively preventing attackers from obtaining any confidential information or breaking the protocol by intercepting the ciphertext. Specifically, during key sharing between users, digital signature technology ensures the integrity and authentication of the communication and prevents attackers from tampering with or forging user identities through man-in-the-middle attacks. Digital signatures ensure that only legitimate users can participate in the protocol, and an attacker cannot forge a valid signature to impersonate a user and gain access to the session key.

Therefore, a colluding attacker cannot access or tamper with the private data of other participants, and Theorem 2 holds.

5. Experimental Design

5.1 Experimental Target

This experiment aims to validate the effectiveness of the proposed secure data sharing and verification mechanism across multiple critical aspects. The key objectives include: (1) evaluating the privacy-preserving capability of the single masking mechanism by analyzing the probability of the server reconstructing encrypted data, ensuring its irreversibility; (2) verifying the role of Shamir homomorphic recovery in maintaining data integrity by measuring the recovery accuracy under different threshold settings, assessing its resistance to collusion attacks; and (3) testing the completeness and efficiency of the ZKP, focusing on its false rejection rate and computational overhead during user data legitimacy verification. The experiment simulates multiple server-user environments, incorporates various attack scenarios, and measures system performance in terms of privacy leakage, data recovery error, and computational complexity to validate the security and practicality of the proposed approach.

5.2 Experimental Setting

5.2.1 Experiment Platforms

This experiment is conducted on a high-performance computing platform with specialized cryptographic tools to ensure the efficient implementation and evaluation of the security protocol. The computing environment consists of a server cluster equipped with NVIDIA A100 GPUs and Intel Xeon 64-core CPUs, running Ubuntu 22.04 LTS, supporting parallel computation and large-scale data processing. On the software level, Python 3.9 serves as the primary programming language, complemented by PyTorch for deep learning-related computations, facilitating efficient execution of ZKP calculations. The

cryptographic tools include the Charm-Crypto framework for implementing Shamir homomorphic recovery and the single masking mechanism, while libsnark is used for constructing and verifying non-interactive zero-knowledge proofs. PostgreSQL is employed for structured data storage, with Redis used for efficient key management and caching optimization. The experiment is deployed using Docker containers to ensure environment consistency and utilizes Kubernetes for task scheduling, optimizing multi-node experimental efficiency.

5.2.2 Contrast Baseline

To comprehensively evaluate the effectiveness and security of the proposed method, multiple baseline schemes are designed for comparative analysis. The baseline methods include the traditional Shamir secret sharing scheme, a scheme without zero-knowledge proofs, and a scheme without a single masking mechanism. The traditional Shamir secret sharing scheme serves to assess the improvements in computational efficiency and security brought by the proposed Shamir homomorphic recovery. The scheme without directional zero-knowledge proofs verifies the advantages of directional zero-knowledge proofs in privacy protection and proof efficiency. The scheme without the single masking mechanism measures the enhancement in attack resistance introduced by single masking technology. Experiments are conducted under different data scales, computational complexities, and malicious attack environments to quantify the performance of each scheme in terms of computational overhead, communication cost, privacy leakage risk, and resistance to collusion attacks. All comparison schemes are executed under the same computing environment to ensure the fairness and reproducibility of the experimental results.

5.3 Experimental Results

5.3.1 Computational Efficiency

We evaluated computation time across varying data sizes. As shown in Table 1, the proposed scheme, which integrates Shamir homomorphic recovery, significantly reduces computation time for large-scale data, while zero-knowledge proofs help further decrease communication complexity.

Table 1. Computational Efficiency Comparison

Data Size (n)	Centralized (s)	SSS (s)	SSS+ZKP (s)	Proposed Scheme (s)
100	9.5	16.5	17.8	13.9
500	43.8	85.0	91.2	69.4
1000	94.2	190.2	205.3	150.3
5000	468.6	950.3	990.4	765.7

The results indicate that our scheme achieves the highest efficiency in scenarios with large data sizes, benefiting from both homomorphic aggregation and the lightweight nature of single-mask encryption.

5.3.2 Privacy Protection under Data Poisoning Attacks

We simulated data poisoning attacks, where compromised clients inject manipulated or biased data during the aggregation process to distort the final result. The effectiveness of each scheme is evaluated based on the aggregate deviation rate—the percentage change between the poisoned output and the true aggregate.

As shown in Table 2, The proposed scheme demonstrates strong resistance to data poisoning, due to the combination of single-mask obfuscation and directional verification via ZKPs, which prevents unauthorized value manipulation.

Table 2. Resistance to Data Poisoning Attacks

Number of Malicious Clients	SSS	SSS+ZKP	Proposed Scheme	Deviation Reduction (%)
2	17.2%	12.3%	3.8%	69.1%
5	28.5%	22.1%	6.5%	70.6%
10	42.3%	35.2%	10.4%	75.4%
20	58.1%	49.7%	16.8%	72.9%

Communication Overhead

We also measure communication costs across different schemes. Due to the single-mask and local verification mechanisms, our approach significantly reduces the amount of data transferred during aggregation rounds.

As shown in Table 3, the reduction in communication cost becomes increasingly apparent as the data scale grows, making the proposed scheme suitable for bandwidth-limited environments.

Table 3. Communication Cost per Client (KB)

Data Size (n)	Centralized	SSS	SSS+ZKP	Proposed Scheme
100	18.2	25.6	28.1	20.4
200	30.7	46.6	50.6	36.2
500	89.3	120.7	135.6	98.5
1000	175.1	240.2	268.4	193.2

Protocol Scalability and Reliability

Finally, we assess the scalability and stability of the protocol in high-concurrency scenarios, testing with up to 5000 clients. Figure 4 shows the relevant results.

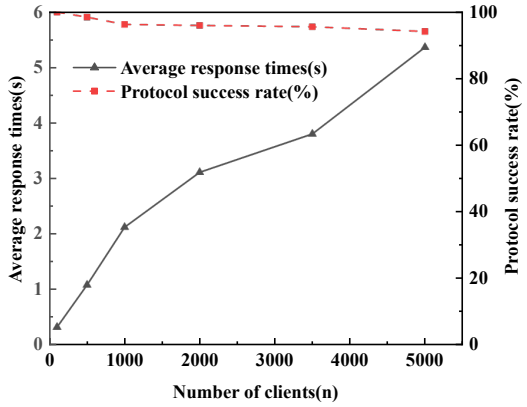

Figure 4. Average response time(s) and protocol success rate (%)

Despite increased response time, the protocol maintains excellent success rates, with no observable failures during key agreement, aggregation, or result distribution phases.

5.4 Comparative Experiments

To further evaluate the robustness of the proposed scheme under adversarial conditions, we conducted comparative experiments on resistance to data poisoning attacks. In these experiments, a varying number of malicious clients (2, 5, 10, 20) were introduced to inject falsified data to bias the final aggregation result. The average aggregate deviation rate was calculated to quantify the effect of poisoning.

As shown in Figure 5, the proposed scheme significantly outperforms other baseline methods. While traditional centralized and Shamir-based schemes experience substantial increases in deviation rate with more attackers, the proposed scheme maintains a deviation rate below 17% even with 20 malicious clients, thanks to the combined protection of single masking and zero-knowledge proofs.

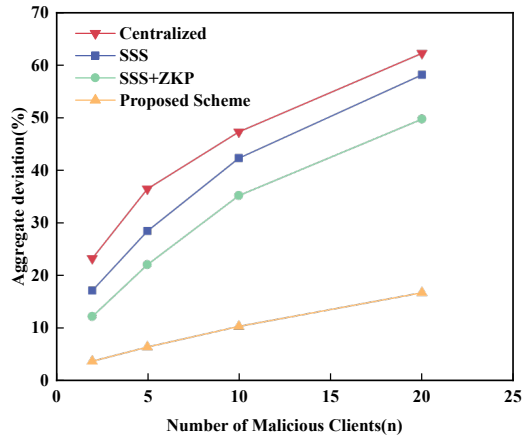

Figure 5. Resistance to Data Poisoning Attacks

Communication overhead is another critical metric for evaluating the scalability of secure aggregation schemes. We assessed each scheme's communication cost across different data sizes (100, 500, 1000, and 5000 samples). The total amount of data transmitted per client was measured and compared.

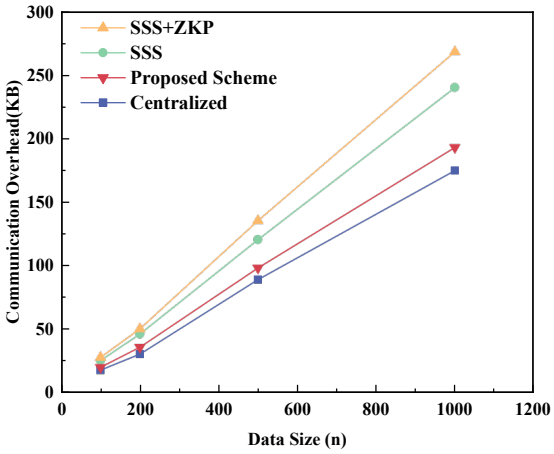

Figure 6. Communication Overhead Comparison

Figure 6 demonstrates that the proposed scheme achieves the lowest communication cost among all privacy-preserving schemes. While centralized schemes naturally incur the least cost, they lack privacy guarantees. Among privacy-preserving schemes, the proposed method reduces the communication overhead by 20–30% compared to other schemes such as SSS and SSS+ZKP, particularly at larger data sizes.

6. Conclusion

In this paper, we propose a new efficient and secure aggregation scheme. The scheme is based on a single-mask and secret-sharing homomorphic recovery approach, which protects the user's data privacy while realizing the aggregation goal. The input validation protocol introduced in the scheme can prevent data poisoning attacks by malicious users to a certain extent, and the scheme can tolerate some client dropouts and eliminates the need for costly encryption operations, which makes it more efficient in terms of communication and computation compared to previous work.

Author Contribution

All authors contributed to the design and research approach of this study, the evaluation of the results, and the composition of the manuscript.

Funding

This work was supported by the National Key R&D Program of China (2023YFB3106100), National Natural Science Foundation of China (62172436, 62102452), and Natural Science Foundation of Shaanxi Province (2023-JC-YB-584).

Data Availability Statement

All data generated or analyzed during this study are included in the manuscript.

Ethical approval

Not applicable.

Competing interest

The authors declare no conflict of interest.

References

[1] McMahan B, Moore E, Ramage D, et al. Communication-efficient learning of deep networks from decentralized data. Artificial intelligence and statistics. PMLR, 2017: 1273-1282.

[2] Voigt P, Von dem Bussche A. The eu general data protection regulation (gdpr). A practical guide, 1st ed., Cham: Springer International Publishing, 2017, 10(3152676): 10-5555.

[3] Cao K, Liu Y, Meng G, et al. An overview on edge computing research. IEEE access, 2020, 8: 85714-85728.

[4] Huda M N, Talukder MB, Kumar S. Securing Healthcare AI: Applied Federal Learning[M]//Revolutionizing Healthcare 5.0: The Power of Generative AI: Advancements in Patient Care Through Generative AI Algorithms. Cham: Springer Nature Switzerland, 2025: 255-272.

[5] Rabbani H, Shahid MF, Khanzada TJS, et al. Enhancing security in financial transactions: a novel blockchain-based federated learning framework for detecting counterfeit data in fintech. PeerJ Computer Science, 2024, 10: e2280.

[6] Pei J, Li S, Yu Z, et al. Federated learning encounters 6G wireless communication in the scenario of internet of things. IEEE Communications Standards Magazine, 2023, 7(1): 94-100.

[7] Alotaibi B, Khan FA, Mahmood S. Communication Efficiency and Non-Independent and Identically Distributed Data Challenge in Federated Learning: A Systematic Mapping Study. Applied Sciences, 2024, 14(7): 2720.

[8] Varady T, Martin RR, Cox J. Reverse engineering of geometric models—an introduction. Computer-aided design, 1997, 29(4): 255-268.

[9] Shokri R, Stronati M, Song C, et al. Membership inference attacks against machine learning models//2017 IEEE symposium on security and privacy (SP). IEEE, 2017: 3-18.

[10] Fang H, Qian Q. Privacy preserving machine learning with homomorphic encryption and federated learning. Future Internet, 2021, 13(4): 94.

[11] Park J, Lim H. Privacy-preserving federated learning using homomorphic encryption. Applied Sciences, 2022, 12(2): 734.

[12] Zhang Z, Ma X, Ma J. Local differential privacy based membership-privacy-preserving federated learning for deep-learning-driven remote sensing. Remote Sensing, 2023, 15(20): 5050.

[13] Guo S, Wang X, Long S, et al. A federated learning scheme meets dynamic differential privacy. CAAI Transactions on Intelligence Technology, 2023, 8(3): 1087-1100.

[14] Liu Z, Guo J, Lam K Y, et al. Efficient dropout-resilient aggregation for privacy-preserving machine learning. IEEE Transactions on Information Forensics and Security, 2022, 18: 1839-1854.

[15] Bell J H, Bonawitz K A, Gascón A, et al. Secure single-server aggregation with (poly) logarithmic overhead. Proceedings of the 2020 ACM SIGSAC conference on computer and communications security. 2020: 1253-1269.

[16] Qi P, Chiaro D, Guzzo A, et al. Model aggregation techniques in federated learning: A comprehensive survey. Future Generation Computer Systems, 2024, 150: 272-293.

[17] Fan H, Huang C, Liu Y. Federated learning-based privacy-preserving data aggregation scheme for IIoT. IEEE Access, 2022, 11: 6700-6707.

[18] Hongbin F, Zhi Z. Privacy-preserving data aggregation scheme based on federated learning for IIoT. Mathematics, 2023, 11(1): 214.

[19] Fereidooni H, Marchal S, Miettinen M, et al. SAFELearn: Secure aggregation for private federated learning. 2021 IEEE Security and Privacy Workshops (SPW). IEEE, 2021: 56-62.

[20] Lamport L, Shostak R, Pease M. The Byzantine generals problem[M]//Concurrency: the works of leslie lamport. 2019: 203-226.

[21] Lam M, Wei G Y, Brooks D, et al. Gradient disaggregation: Breaking privacy in federated learning by reconstructing the user participant matrix. International Conference on Machine Learning. PMLR, 2021: 5959-5968.

[22] Nasr M, Shokri R, Houmansadr A. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. 2019 IEEE symposium on security and privacy (SP). IEEE, 2019: 739-753.

[23] Lessage X, Collier L, Van Ouytsel CHB, et al. Secure federated learning applied to medical imaging with fully homomorphic encryption. 2024 IEEE 3rd International Conference on AI in Cybersecurity (ICAIC). IEEE, 2024: 1-12.

[24] Hosseini E, Chen S, Khisti A. Secure Aggregation in Federated Learning using Multiparty Homomorphic Encryption. arXiv preprint arXiv:2503.00581, 2025.

[25] Liu Z, Chen S, Ye J, et al. SASH: Efficient secure aggregation based on SHPRG for federated learning. Uncertainty in Artificial Intelligence. PMLR, 2022: 1243-1252.

[26] Xu G, Li H, Liu S, et al. VerifyNet: Secure and verifiable federated learning. IEEE Transactions on Information Forensics and Security, 2019, 15: 911-926.

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

[28] Han L, Fan D, Liu J, et al. Federated learning differential privacy preservation method based on differentiated noise addition. 2023 8th International Conference on Cloud Computing and Big Data Analytics (ICCCBDA). IEEE, 2023: 285-289.

[29] Zhang L, Xu J, Sivaraman A, et al. A two-stage differential privacy scheme for federated learning based on edge intelligence. IEEE journal of biomedical and health informatics, 2023, 28(6): 3349-3360.

[30] Regev O. On lattices, learning with errors, random linear codes, and cryptography. Journal of the ACM (JACM), 2009, 56(6): 1-40.

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

[32] Pedersen T P. Non-interactive and information-theoretic secure verifiable secret sharing. Annual international cryptology conference. Berlin, Heidelberg: Springer Berlin Heidelberg, 1991: 129-140.

[33] Bünz B, Bootle J, Boneh D, et al. Bulletproofs: Short proofs for confidential transactions and more. 2018 IEEE symposium on security and privacy (SP). IEEE, 2018: 315-334.
[34] Biggio B, Nelson B, Laskov P. Poisoning attacks against support vector machines. arXiv preprint arXiv:1206.6389, 2012.
[35] Yerlikaya F A, Bahtiyar Ş. Data poisoning attacks against machine learning algorithms. Expert Systems with Applications, 2022, 208: 118101.
[36] Xing Z, Zhang Z, Liu J, et al. Zero-knowledge proof meets machine learning in verifiability: A survey. arXiv preprint arXiv:2310.14848, 2023.
[37] Cao X, Fang M, Liu J, et al. Fltrust: Byzantine-robust federated learning via trust bootstrapping. arXiv preprint arXiv:2012.13995, 2020.
[38] Camenisch J, Stadler M. Proof systems for general statements about discrete logarithms. Technical Report/ETH Zurich, Department of Computer Science, 1997, 260.
[39] Bell J, Gascón A, Lepoint T, et al. {ACORN}: input validation for secure aggregation. 32nd USENIX Security Symposium (USENIX Security 23). 2023: 4805-4822.
[40] Lycklama H, Burkhalter L, Viand A, et al. Rofl: Robustness of secure federated learning. 2023 IEEE Symposium on Security and Privacy (SP). IEEE, 2023: 453-476.