EBS-CFL: Efficient and Byzantine-robust Secure Clustered Federated Learning

Zhiqiang Li[1], **Haiyong Bao**[1*], **Menghong Guan**[1], **Hao Pan**[1], **Cheng Huang**[2], **Hong-Ning Dai**[3]

[1] East China Normal University
[2] Fudan University
[3] Hong Kong Baptist University
{51255902170,mhguan,51255902119}@stu.ecnu.edu.cn, hybao@sei.ecnu.edu.cn,
chuang@fudan.edu.cn, hndai@ieee.org

Abstract

Despite *federated learning* (FL)'s potential in collaborative learning, its performance has deteriorated due to the data heterogeneity of distributed users. Recently, *clustered federated learning* (CFL) has emerged to address this challenge by partitioning users into clusters according to their similarity. However, CFL faces difficulties in training when users are unwilling to share their cluster identities due to privacy concerns. To address these issues, we present an innovative Efficient and Robust Secure Aggregation scheme for CFL, dubbed **EBS-CFL**. The proposed EBS-CFL supports effectively training CFL while maintaining users' cluster identity confidentially. Moreover, it detects potential poisonous attacks without compromising individual client gradients by discarding negatively correlated gradients and aggregating positively correlated ones using a weighted approach. The server also authenticates correct gradient encoding by clients. EBS-CFL has high efficiency with client-side overhead $\mathcal{O}(ml + m^2)$ for communication and $\mathcal{O}(m^2 l)$ for computation, where m is the number of cluster identities, and l is the gradient size. When $m = 1$, EBS-CFL's computational efficiency of client is at least $\mathcal{O}(\log n)$ times better than comparison schemes, where n is the number of clients. In addition, we validate the scheme through extensive experiments. Finally, we theoretically prove the scheme's security.

Code — https://github.com/Lee-VA/EBS-CFL

Introduction

Federated learning (FL) (McMahan et al. 2016), as a distributed machine learning paradigm, decentralizes model training across multiple local devices, thereby preserving data privacy. However, traditional FL methods operate under the assumption of independently and identically distributed (i.i.d.) data, which limits their effectiveness in real-world scenarios with non-i.i.d. data. To address this challenge, Clustered Federated Learning (CFL) (Sattler, Müller, and Samek 2021) has been proposed. CFL enhances model performance on heterogeneous datasets by grouping participants into clusters based on data similarity, effectively extending FL's applicability to non-i.i.d. scenarios.

*Corresponding author.

Most CFL methods, such as recent researches (Ghosh et al. 2022; Zhang et al. 2023a; Vahidian et al. 2023), mainly focus on improving accuracy or performance efficiency. For example, *Iterative Federated Clustering Algorithm (IFCA)* (Ghosh et al. 2022) is the first to provide convergence guarantees, which demonstrates its effectiveness in non-convex problems of neural networks. Despite numerous studies on CFLs, they still face challenges incurred by Byzantine attacks and privacy leakage. Particularly, they do not consider protecting against clients' cluster identities, which could potentially leak clients' important privacy.

As highlighted in DLG (Zhu, Liu, and Han 2019), the vulnerability of FL is exposed when gradient inference attacks target unencrypted gradients. Though Differential Privacy (DP) methods provide some protection, their noise can hinder model convergence. The integration of DP with secure aggregation in researches (Chen et al. 2022; Chen, Ozgur, and Kairouz 2022) lessens DP's security requirements and tackles differential attacks, but efficiency is still problematic. Secure aggregation methods (Bonawitz et al. 2016; Choi et al. 2020; Kadhe et al. 2020; Jahani-Nezhad et al. 2023) can protect plaintext gradients, yet they fall short in dealing with Byzantine attacks. Robust algorithms, as achieved in PEFL (Liu et al. 2021), rely on interaction between the server and the cloud platform, but this approach assumes no collusion between the two, which is a relatively weak security assumption. Lastly, SAFELearning (Zhang et al. 2023b) operates under the assumption of a single cloud server and supports only the FedAvg algorithm, overlooking compatibility with other federated learning algorithms. The current research primarily faces two core challenges:

Lack of a scheme to protect clients' cluster identities. Existing solutions require the acquisition of clients' cluster identities for training. However, there is no privacy preservation scheme for cluster identities. In this way, attackers may use the clients' cluster identities as additional information to launch inference attacks.

Lack of a comprehensive secure aggregation solution. Despite the advent of secure aggregation, the comprehensive solution of ensuring both efficiency and Byzantine-robustness in a single cloud server setting with compatibility of diverse FL schemes is still largely elusive.

To address these challenges, we propose a novel *Efficient and Byzantine-robust Secure Clustered Federated Learning*

Approach	Malicious Adversary	Defend Against Byzantine Attacks	Single-Server	Compatible Framework
CCESA (Choi et al. 2020)	×	×	✓	FedAvg
FastSecAgg (Kadhe et al. 2020)	×	×	✓	FedAvg
SwiftAgg+ (Jahani-Nezhad et al. 2023)	×	×	✓	FedAvg
PEFL (Liu et al. 2021)	✓	✓	×	FedAvg
SAFELearning (Zhang et al. 2023b)	✓	✓	✓	FedAvg
EBS-CFL	✓	✓	✓	FedAvg, IFCA

Table 1: Comparing related secure federated learning schemes.

	Communication		Computation	
Approach	Server	Per-client	Server	Per-client
SecAgg	$\mathcal{O}(nl + sn^2)$	$\mathcal{O}(l + sn)$	$\mathcal{O}(n^2 l)$	$\mathcal{O}(nl + sn^2)$
CCESA	$\mathcal{O}(nl + sn\sqrt{n \log n})$	$\mathcal{O}(l + s\sqrt{n \log n})$	$\mathcal{O}(nl \log n)$	$\mathcal{O}(l\sqrt{n \log n} + sn \log n)$
FastSecAgg	$\mathcal{O}(nl + n^2)$	$\mathcal{O}(l + n)$	$\mathcal{O}(l \log n)$	$\mathcal{O}(l \log n)$
EBS-CFL (m=1)	$\mathcal{O}(nl)$	$\mathcal{O}(l)$	$\mathcal{O}(n^2 + nl \log n)$	$\mathcal{O}(l)$
EBS-CFL	$\mathcal{O}(nml + n)$	$\mathcal{O}(ml + m^2)$	$\mathcal{O}(m^2(nml + nm^2 + n^2 + nl \log n))$	$\mathcal{O}(m^2 l)$

Table 2: Communication and computation overhead. For better clarity, we have included the case with the number of clusters $m = 1$ in the table, comparing it with other schemes that do not incorporate the concept of multiple clusters.

(EBS-CFL) scheme. In particular, we devise a *Robust Federated Clustering Algorithm (RFCA)* with the integration of a Byzantine robust algorithm based on cosine similarity (Cao et al. 2021) into the aggregation process of IFCA (Ghosh et al. 2022). Additionally, we protect clients' gradients and cluster identities by developing a secure aggregation scheme. This scheme allows the server to perform clustered aggregation without accessing the clients' gradients or clusters' identities while filtering out malicious gradients. We also introduce compression scheme to guarantee the scalability of communication and computation costs. Particularly in terms of communication overhead, our scheme is more advantageous compared to other schemes. When the number of clusters is 1, the server's overhead is linearly related to the number of clients, while the overhead for an individual client is independent of the total number. Our main contributions are summarized as follows.

- Our scheme can protect the privacy of client gradients and cluster identities when training models. Specifically, we designed the Verifiable Orthogonal Matrix Confusion for Aggregation (VOMCA), which enables clients to encode both their cluster identities and gradient information simultaneously. This innovative design allows for the aggregation of gradients according to set cluster identities, without exposing clients' identities.
- We designed a secure aggregation scheme to achieve high efficiency, Byzantine-robustness, a single cloud server assumption, and compatibility with CFL schemes. We use the assumption of a single secure cloud server, which is more secure than those based on multiple cloud servers. Specifically, we designed RFCA, which realizes Byzantine-robust CFL. Meanwhile, we designed the Secure ReLU Function Computation Mechanism (SRFC) to filter out malicious gradients in secure aggregation. Lastly, we designed a compression scheme to significantly enhance the overall efficiency of the system.
- We conduct theoretical analysis and extensive experiments to evaluate our proposed EBS-CFL. Through theoretical analysis, we explain the communication and computational complexity of our scheme. Moreover, we conducted experiments involving multiple variables related to communication and computation overhead, conducting detailed data analysis of the entire process of aggregation stage. And we incorporate experiments involving typical Byzantine attacks in FL to validate the robustness of our proposed EBS-CFL. Finally, we demonstrate the security of the EBS-CFL through security analysis.

Table 1 compares EBS-CFL with other state-of-the-art works in terms of key characteristics. Table 2 compares EBS-CFL with other state-of-the-art works in terms of communication and computational complexity. Note that for detailed illustration of theoretical analysis of complexity, please refer to the appendix.

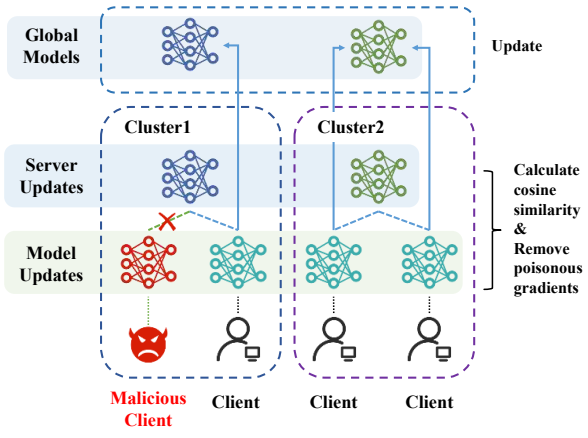


Figure 1: Overview of RFCA.

Related Work

We provide an overview of relevant research from two perspectives: **efficient secure aggregation** and **defense against Byzantine attacks**.

In the realm of **efficient secure aggregation**, PSA (Bonawitz et al. 2016) introduced a lightweight protocol that allows the server to aggregate model updates from multiple clients without accessing individual updates. CCESA (Choi et al. 2020) presented a low-complexity scheme using sparse random graphs to ensure data privacy through a secret-sharing node topology. FastSecAgg (Kadhe et al. 2020) developed a protocol, which leverages the Fast Fourier Transform (FFT) to create FastShare, a novel multi-secret sharing scheme that enables more efficient computation and communication. SwiftAgg+ (Jahani-Nezhad et al. 2023) proposed a secure aggregation protocol that significantly reduces communication overhead while preserving privacy, achieving optimal communication loads and security guarantees. However, these schemes do not address Byzantine attacks.

In the realm of **defense against Byzantine attacks**, PEFL (Liu et al. 2021) proposed a framework that uses homomorphic encryption and penalizes poisoning behavior by extracting gradient data with logarithmic functions, addressing the problems of both privacy leakage and poisonous attack. SAFELearning (Zhang et al. 2023b) introduced technique that supports secure aggregation through Unintentional Random Grouping (ORG) and Partial Parameter Disclosure (PPD), effectively preventing malicious attacks. ShieldingFL (Wan et al. 2022) proposed an adaptive client selection strategy to combat Byzantine attacks by filtering out honest clients, but it does not encrypt or obfuscate gradients.

All existing solutions build upon the traditional FL framework, which struggles with handling non-i.i.d. data. As the demand for handling non-i.i.d. data grows, the recently proposed CFL has provided a solution. Therefore, there is an urgent need for a secure aggregation framework that can be compatible with CFL.

System Model

Our system involves three entities: the **Clients**, the **Server**, and the **Key Distribution Center (KDC)**. The functions of each entity are as follows.

- **Clients**: Clients encode gradients obtained from local training using encoding keys and upload the encoded gradients to the server. Malicious clients may attempt to bypass the robustness algorithm by altering encoded gradients or generating them incorrectly, thereby impacting the global model.
- **Server**: The server broadcasts the model, collects client-uploaded gradients for validation, and performs weighted clustering aggregation using our proposed EBS-CFL. Malicious servers may improperly execute the robustness algorithm to obtain target gradients.
- **Key Distribution Center (KDC)**: The KDC generates and distributes keys.

Algorithm 1: ClusteredModelUpdate($\{\theta_i\}_{i=1}^m$, D, b, η, T)

Input: initialization $\{\theta_i\}_{j=1}^m$; training datasets D; batch size b; learning rate η; number of iterations T.

$\{\theta_i^0\}_{i=1}^m \leftarrow \{\theta_i\}_{i=1}^m$
$Loss(D;\theta_j^0) \leftarrow \min(\{Loss(D_b;\theta_i^0\}_{i=1}^m))$
for $t = 1$ to T **do**
    Randomly sample a batch D_b from D.
    $\theta_j^t \leftarrow \theta_j^{t-1} - \eta \nabla Loss(D_b;\theta_j^{t-1})$
end for
return $j, \theta_j^T - \theta_j^0$

Proposed Scheme

This section introduces our proposed EBS-CFL. First, we present a Byzantine-robust clustered federated learning algorithm, RFCA. Based on this, we designed a secure aggregation scheme for privacy preservation. Finally, we improved the overall efficiency of the system through an efficient compression scheme.

Robust Federated Clustering Algorithm (RFCA)

We first design Algorithm 1 to train clustered gradient, and then design Robust Federated Clustering Learning Algorithm (RFCA). The overview of the RFCA scheme is illustrated Figure 1.

The specific steps of RFCA are as follows.

- The server sets m clusters and initializes m models. Then the server collects a small and clean public dataset and trains m models to obtain m server updates. Finally, m server updates are obtained, represented as $\{g_0^j\}_{j=1}^m$.
- The clients run Algorithm 1 to train local model.
- The server calculates cosine similarities between server updates and clients' gradients, and removes poisonous gradients. Then, for $1 \le i, j \le n$ and $1 \le k \le m$, the server calculates

$$g^k = \frac{1}{\sum_{j=1}^n \mathrm{ReLU}(c_j^k)} \sum_{i=1}^n \mathrm{ReLU}(c_i^k) \frac{\|g_0^k\|}{\|g_i\|} \cdot g_i, \quad (1)$$

where $c_i^k = s_{i,k} \frac{\langle g_i, g_0 \rangle}{\|g_i\| \cdot \|g_0\|}$, k represents the cluster identity, $\{s_{i,k}\}_{k=1}^m$ represents a one-hot encoding vector ($s_{i,k} = 1$, when k matches the target identity), and g_i represents the gradient submitted by the i-th client.

Note that for detailed illustration of the pseudo code of RFCA, please refer to the appendix.

Verifiable Orthogonal Matrix Confusion for Aggregation (VOMCA)

VOMCA is a privacy-preserving aggregation technology. Similar to secret sharing, individual data is only decryptable after aggregation. It also ensures that clients cannot bypass robustness checks and submit malicious gradients.

Definition 1. *Given a set of matrices $\{M_i\}_{i=1}^l$, they are called mutually orthogonal matrices (Zheng et al. 2022) if*

they satisfy the following condition:

$$M_i^T M_j = \begin{cases} I, & i = j, \\ O, & i \neq j, \end{cases} \tag{2}$$

where I and O respectively represent an identity matrix and a zero matrix, for $1 \leq i, j \leq l$.

For $1 \leq i \leq n$, the i-th client can encode the secret x_i as follows.

$$\chi_i = x_i M_i + R_i M_{i+n}, \tag{3}$$

where $\sum_{i=1}^{n} R_i = O$ and $\{M_i\}_{i=1}^{2n}$ are Mutually Orthogonal Matrices. Construct verification key vk as $\{\sum_1^n \mathcal{V}_i M_{i+n}, \{\mathcal{V}_i \cdot R_i^T\}_{i=1}^n\}$. The verification function is as follows.

$$\mathcal{V}(\chi_i, vk) = \begin{cases} 1, & vk \cdot \chi_i^T = \mathcal{V}_i \cdot R_i^T, \\ 0, & vk \cdot \chi_i^T \neq \mathcal{V}_i \cdot R_i^T, \end{cases} \tag{4}$$

where $\mathcal{V}_i$ is a random matrix that has the same shape with R_i. And construct decode key dk as $\sum_1^{2n} M_i$, which satisfies the following relations,

$$\chi_i \cdot dk^T = x_i + R_i, \quad \text{for } 1 \leq i \leq n,$$
$$(\sum_{i=1}^{n} \chi_i) \cdot dk^T = \sum_{i=1}^{n} x_i.$$

Theorem 1. *For any $1 \leq i \leq n$, $a_i \in \{\chi_i, \chi_i'\}$, where χ_i' represents a ciphertext tampered by an adversary. The $\mathcal{V}$ satisfies*

$$Pr[\sum_{i=1}^{k} a_i dk^T \neq \sum_{i=1}^{k} x_i \wedge \forall i \in [n], \mathcal{V}(\chi_i, vk) = 1] \leq \epsilon, \tag{5}$$

where $\epsilon = \frac{1}{\lambda^l}$, λ represents the probability of correctly predicting pseudo-random numbers without additional information, and l represents the size of x_i.

Proof. For detailed illustration of the proof process, please refer to the appendix. □

Secure ReLU Function Computation Mechanism (SRFC)

We designed SRFC to exclude gradients with negative cosine similarity, while ensuring security.

We define the function $\mathcal{F}$ to square each element in the matrix, preserving their original signs. Meanwhile, we define $\mathcal{F}^{-1}$ to calculate the square root of each element's absolute value, also preserving their original signs. The definitions of $\mathcal{F}$ and $\mathcal{F}^{-1}$ are as follows.

$$\mathcal{F}(A) = (\frac{(A)_{ij}^3}{|(A)_{ij}|}),\ \mathcal{F}^{-1}(A) = (\frac{(A)_{ij}\sqrt{|(A)_{ij}|}}{|(A)_{ij}|}),$$

where $A \in \mathbb{R}^{a \times b}$. For $1 \leq i \leq n$, input $x_i \in \mathbb{R}$, define the encode of x_i as

$$E(x_i) = x_i M_i + \frac{1}{2}\zeta_i M_{i+n} + R_i.$$

Construct $\{\alpha_i\}_{i=1}^n$ as follows.

$$\alpha_i = x_i \cdot M_i^T R_i' M_i + M_{i+n}^T R_i' M_{i+n},$$

where $\forall M \in \{M_i\}_{i=1}^{2n}, M \in \mathbb{R}^{lm \times 2lmn}$, and $\{M_i\}_{i=1}^{2n}$ are Mutually Orthogonal Matrices. And construct β, τ_1, τ_2, and τ_3 as follows.

$$\begin{cases} \beta = \sum_{i=1}^{2n} M_i, \\ \tau_1 = \sum_{i=1}^{n} M_i^T {R_i'}^{-1} M_i + M_{i+n}^T {R_i'}^{-1}(R_i^{(0)} + \zeta_i \cdot M_{i+n}), \\ \tau_2 = \sum_{i=1}^{n} M_i^T {R_i'}^{-2}(M_i \circ M_i) \\ \qquad + M_{i+n}^T {R_i'}^{-2}(R_i^{(1)} \circ R_i^{(1)} + R_i^{(2)}), \\ \tau_3 = \sum_{i=1}^{n} M_i^T {R_i'}^{-2}\mathcal{F}(2 \cdot M_i \circ R_i^{(1)}) - M_{i+n}^T {R_i'}^{-2}\mathcal{F}(R_i^{(2)}), \end{cases} \tag{6}$$

where $\circ$ denotes hadamard product, $\zeta_i \in \mathbb{R}^{lm \times lm}$, $\sum_{i=1}^n \zeta_i = 0$, $R_i' \in \mathbb{R}^{lm \times lm}$, R_i' is an invertible matrix, $R_i, R_i^{(0)}, R_i^{(1)}, R_i^{(2)} \in \mathbb{R}^{lm \times 2lmn}$, $R_i = R_i^{(0)} + R_i^{(1)}$, $\sum_{i=1}^n R_i = 0$.

For $1 \leq j \leq lm, 1 \leq k \leq 2lmn$, $r_{jk}^{(1)}$ and $r_{jk}^{(2)}$ are random numbers, the $R_i^{(1)}$ and $R_i^{(2)}$ satisfy that if $(M_i)_{jk} \leq 0$,

$$\Pr[(R_i^{(1)})_{jk} = r_{jk}^{(1)} \wedge (R_i^{(2)})_{jk} = 0] = 1, \tag{7}$$

where $0 < r_{jk} < \max(\{|x_i|\}_{i=1}^n)$; if $(M_i)_{jk} > 0$,

$$\Pr[(R_i^{(1)})_{jk} = 0 \wedge (R_i^{(2)})_{jk} = r_{jk}^{(2)}] = 1, \tag{8}$$

where $0 < r_{jk}^{(2)} < 2 \cdot \max(\{|x_i|\}_{i=1}^n)^2$.

The principle of SRFC is as follows. Firstly, by squaring the matrix elements and then taking the square root, we can obtain the absolute value of the corresponding elements. Then, by adding this absolute value to the original element, we can obtain the output of the ReLU function.

The α is the encoding of x, and when it performs matrix multiplication operation with itself, the decoding can yield x^2. The function of β is to perform decoding, while τ_1, τ_2, and τ_3 help to compute the square of the target element and then compute the square root. Since the entire computation process requires privacy preservation, a random number factor is introduced and incorporated into all parameters.

The $\sum_{i=1}^n E(ReLU(x))$ and $\sum_{i=1}^n ReLU(x_i)$ can be calculated as theorem 2.

Theorem 2. *The $\sum_{i=1}^n E(ReLU(x))$ can be calculated as*

$$\sum_{i=1}^{n} E(ReLU(x_i)) = \frac{1}{2}(\beta \cdot (\sum_{i=1}^{n} \alpha_i) \cdot \tau_1 + \sum_{i=1}^{n} \mathcal{F}^{-1}(\beta \cdot \alpha_i^2 \cdot \tau_2 + \mathcal{F}^{-1}(\beta \cdot \alpha_i^2 \cdot \tau_3))), \tag{9}$$

and the $\sum_{i=1}^n ReLU(x_i)$ can be calculated as

$$\sum_{i=1}^{n} ReLU(x_i) = \frac{1}{l} tr(\sum_{i=1}^{n} E(ReLU(x_i) \cdot \beta^T). \tag{10}$$

Proof. For detailed illustration of the proof process, please refer to the appendix. □

Secure Aggregation Scheme

Based on RFCA, we design our secure aggregation scheme. In each round of training, the number of participating clients is denoted by n, the number of cluster identities is denoted by m, and the dimension of the gradients is denoted by l.

Initialization: KDC pre-generates multiple sets of Mutually Orthogonal Matrices $\{A_i|A_i \in \mathbb{R}^{l\times lm}\}_{i=1}^{m}$, $\{M_i|M_i \in \mathbb{R}^{lm\times 2lmn}\}_{i=1}^{2n}$, and $\{M_i'|M_i' \in \mathbb{R}^{lm\times 3lmn}\}_{i=1}^{3n}$. The server sends the update gradient $\{g_0^i\}_{j=1}^{m}$ to KDC. Then KDC generates β, τ_1, τ_2, and τ_3 as Eq.(6). And KDC generates the encode keys $\{ek_i\}_{i=1}^{n}$, $\{\alpha_i'\}_{i=1}^{2lmn}$, vk, and dk as follows.

The $(M)_i$ represents the i-th row of matrix M, for $1 \le i \le n$ and $1 \le j \le m$,

$$ek_i = \{ek_i^0, ek_i^1\} = \{M_i', \sum_{j=1}^{m} R_{ij}'' A_j M_{i+n}' + \mu_i M_{i+2n}'\}, \tag{11}$$

where $\sum_{i=1}^{n} R_{ij}'' = 0$, $||R_{ij}''|| = 1$, $\sum_{i=1}^{n} \mu_i = 0$, and $||\mu_i|| = 1$. And for $1 \le i \le 2lmn$,

$$\begin{aligned}\alpha_i' =& \sum_{j=1}^{n} {M_j'}^T (\sum_{k=1}^{m} A_k^T (g_0^k)^T)(M_j^T R_j' M_j)_i \\ &+ {M_{j+2n}'}^T {\mu_j}^{-1} (M_{j+n}^T R_j' M_{j+n})_i.\end{aligned} \tag{12}$$

$$\begin{cases} vk = \{\sum_{i=1}^{n} \mathcal{V}_i \sum_{j=1}^{m} A_j M_{i+n}', \{\mathcal{V}_i \cdot \sum_{j=1}^{m} {R_{ij}''}^T\}_{i=1}^{n}\}, \\ dk = \sum_{i=1}^{n} M_i^T M_i' + M_{i+n}^T \sum_{j=1}^{n} (M_{j+n}' + M_{j+2n}'), \end{cases} \tag{13}$$

where $\{\mathcal{V}_i\}_{i=1}^{n}$ are random matrices generated by KDC.

Broadcast Phase: In the first round, the KDC sends ek_i and $\{A_i\}_{i=1}^{m}$ to the i-th client. For subsequent rounds, the KDC updates ek_i^1 by updating R_{ij}'' and sends the updated ek_i^1 to the i-th client. The server first classifies the current models using the public dataset, then sends the global model to the selected clients.

Training Phase: The training phase follows the same procedure as RFCA. The cluster corresponding to the i-th client is denoted as j, the gradient is encoded as $\delta_i = \frac{g_i}{\|g_i\|} A_j M_i' + \sum_{j=1}^{m} R_{ij}'' A_j M_{i+n}' + \mu_i M_{i+2n}'$. The encoded gradient δ_i is then uploaded.

Aggregation Phase: The KDC sends $\{\alpha_i\}_{i=1}^{lm}$, β, τ_1, τ_2, dk, and vk to the server. The server then verifies each client as described in Eq.(4), and checks if $\|\delta_i\|^2 = 3$ for $1 \le i \le n$. If the verification in Eq.(4) passes, it can be conclusively demonstrated that the client is unable to manipulate the ciphertext. If $\|\delta_i\|^2 = 3$, it can be inferred that the client has performed a normalization operation $\frac{g_i}{\|g_i\|}$. If the verification fails, the server will request the failed clients to resend their encoded gradients, or the server will restart the process. For $1 \le i \le n$, the server calculates α_i as follows,

$$(\alpha_i)_j = \delta_i \cdot \alpha_j'. \tag{14}$$

The server calculates $\sum_{i=1}^{n} E(ReLU(w_i))$ and $\sum_{i=1}^{n} ReLU(w_i)$ as Eq.(9) and Eq.(10).

Finally, the server calculates the aggregated gradient g as follows,

$$dk' = \frac{\sum_{i=1}^{n} E(ReLU(w_i)) \cdot dk}{\sum_{i=1}^{n} ReLU(w_i)}, \tag{15}$$

$$g = (\sum_{i=1}^{n} \delta_i) \cdot {dk'}^T. \tag{16}$$

Update Phase: For $1 \le j \le m$, the server updates as follows.

$$\theta_j^{(t)} = \theta_j^{(t-1)} - \eta \cdot ||g_0^j|| \cdot g A_j^T. \tag{17}$$

Secure Key Transformation Mechanism (SKT)

For $1 \le i \le n$, $1 \le j \le m$, and converse function $\mathcal{C} : i \to j$, construct Transformation key tk_i as $M_i^T M_{\mathcal{C}(i)}' + M_{i+n}^T (M_{\mathcal{C}(i+n)}' + R_i^{-1} \cdot R_i')$ for the key transformation of χ_i, and the Transformation function is as follows.

$$\mathcal{T}(\chi_i, tk_i) = \chi_i \cdot tk_i = \chi_i' + R_i', \tag{18}$$

where c represents a constant, $\{M_i\}_{i=1}^{n}$ and $\{M_j'\}_{j=1}^{m}$ are Mutually Orthogonal Matrices, $\chi_i = x_i M_i + R_i M_{i+n}$, and $\chi_i' = x_i M_{\mathcal{C}(i)}' + R_i M_{\mathcal{C}(i+n)}'$. It satisfies

$$(\sum_{i=1}^{m} \chi_i') \cdot (dk')^T = \sum_{i=1}^{n} c \cdot x_i, \tag{19}$$

where $dk' = \sum_{i=1}^{m} M_i'$.

Compression Scheme

We design gradient segmentation to reduce gradient dimensionality for individual aggregations and layered aggregation to minimize the number of clients involved in each aggregation.

Gradient Segmentation: The i-th client flattens the local gradients into a vector by rows, and concatenates them layer by layer to form a vector g_i. Let the vector g_i be divided into s segments, denoted as $\{g_{ij}\}_{j=1}^{s}$, each segment has a length of t, which satisfies that $s \cdot t \ge l$. For $1 \le i \le n$, $1 \le j \le m$, and $1 \le k \le s$, the R'' and μ satisfy

$$\sum_{k=1}^{s} ||R_{ijk}''||^2 = 1, \ ||\mu_{ijk}||^2 = \frac{1}{s}.$$

For $1 \le i \le n$, $\mathcal{A}(i)$ represents the i-th client, the corresponding inner products are calculated as follows.

$$g_i {g_0^{\mathcal{A}(i)}}^T = \sum_{j=1}^{s} g_{ij} {g_{0,j}^{\mathcal{A}(i)}}^T. \tag{20}$$

For $1 \le i \le 3mnt$, $1 \le j \le n$, and $1 \le k \le s$, following the gradient segmentation, Eq.(12) should be reformulated as follows.

$$\begin{aligned}\alpha_{ik}' =& \sum_{j=1}^{n} {M_j'}^T (\sum_{k=1}^{m} A_k^T (g_0^k)^T)(M_j^T M_j)_i \\ &+ \frac{1}{s} {M_{j+2n}'}^T {\mu_{jk}}^{-1} (M_{j+n}^T R_j' M_{j+n})_i.\end{aligned}$$

Finally, aggregate the corresponding segments and concatenate the segments back to the original gradients.

Layered Aggregation: The total number of clients is denoted as n, and the clients are divided into ξ groups, each of which is with a maximum size of $\lceil \frac{n}{\xi} \rceil$.

KDC generates ek, vk, α'_i, and transformation keys for each group. Then, KDC generates β, τ_1, τ_2, and τ_3. In Addition, ζ, μ, R'', and $\{A_i\}_{i=1}^m$ are shared across all clients. The specific steps for aggregation are as follows.

- First, the steps before aggregation for each group are the same as the secure aggregation scheme. In the aggregation phase, the server will calculate $\sum_{i=1}^{n} E(ReLU(w_i))$ as Eq.(9). However, since each group cannot eliminate the random matrices and lacks a corresponding dk prior to key transformation, the decoding is not performed.
- Second, the server performs keys transformation in each group.
- Finally, the server decodes the encoded gradients and subsequently updates the global model.

Each transformation can be regarded as the aggregation of each group. The number of transformation rounds can be denoted as x. The total size $\mathcal{S}(n, \{\xi_i\}_{i=1}^x)$ of transformation keys can be expressed as follows.

$$\mathcal{S}(n, \{\xi_i\}_{i=1}^x) = 2lm\xi_x + \sum_{i=1}^{x-1} 4l^2m^2\xi_i\xi_{i+1} \prod_{j=i+1}^{x} \xi_j,$$

where the $\{\xi_i\}_{i=1}^x$ satisfies that $\prod_{i=1}^x \xi_i \geq n$. For $1 \leq i \leq n$, $1 \leq j \leq x$, the key transformation is as follow.

$$\delta_i^j = \mathcal{T}(\delta_i^{j-1}, \sum_{k=1+\nu}^{\xi_j+\nu} tk_k^j), \tag{21}$$

where $\nu = i - (i \mod \xi_j)$, and $\delta_i^0 = \delta_i$. The α_i has the same transformation process as Eq.(21).

Convergence Guarantees

In this section, we provide convergence guarantees for RFCA. Our analysis builds upon the assumptions of IFCA, with additional considerations specific to the Byzantine-robustness algorithm.

Theorem 3. *Let all assumptions hold, and the step size γ be chosen as $\gamma = \frac{1}{L}$. Suppose that at a certain iteration of the RFCA algorithm, the parameter vector θ_j satisfies: $\|\theta_j - \theta_j^*\| \leq \left(\frac{1}{2} - \alpha\right) \frac{\lambda}{L}\Delta$, where $0 < \alpha < \frac{1}{2}$. Let θ_j^+ denote the next iterate, m is the total number of clients, k is the number of clusters, and $w = \sum_{i=1}^m w_i$. For any fixed $j \in [k]$ and $\delta \in (0, 1)$, the following holds with probability at least $1 - \delta$,*

$$\|\theta_j^+ - \theta_j^*\| \leq \left(1 - \frac{p\lambda}{8L}\right) \|\theta_j - \theta_j^*\| + \frac{c_0\sqrt{2v^2+1}}{\delta L\sqrt{pmn'}} + c_1 \frac{\eta^2 m}{\delta\alpha^2\lambda^2\Delta^4 wn'} + c_2 \frac{\eta k\sqrt{2kmv^2+km}}{\delta^{\frac{3}{2}}\alpha\lambda L\Delta^2 wn'}. \tag{22}$$

This theorem guarantees the convergence of RFCA. Note that for details on assumptions, parameter definitions, theoretical derivations, and the proof, please refer to the appendix.

Performance Evaluation

In this section, we will evaluate the performance of our proposed EBS-CFL, including both efficiency and Byzantine-robustness. We carry out experiments on MNIST (Deng 2012), CIFAR10, and CIFAR100 (Krizhevsky and Hinton 2009). Note that more detailed experiments will be presented in the appendix.

Efficiency

In this section, we will evaluate the efficiency of our scheme by contrasting it with solutions that do not incorporate privacy preservation, thereby demonstrating the optimization we have achieved in our privacy-preserving scheme.

We evaluate the average client communication, as shown in Figures 2 and 3. Experiments showed that the communication overhead does not increase with the number of clients, and it has a linear relationship with the size of the data vector. Under normal values of m, the communication overhead does not significantly increase with changes in other variables, proving the practicality of the method.

We conducted similar experiments for server-side computational overhead, with the number of cluster identities set to 2, as shown in Figure 4. We measured the efficiency by calculating the consumed time. Experiments showed that the time consumption of our scheme is almost equivalent to FedAvg, corresponding to the theoretical analysis of computational overhead.

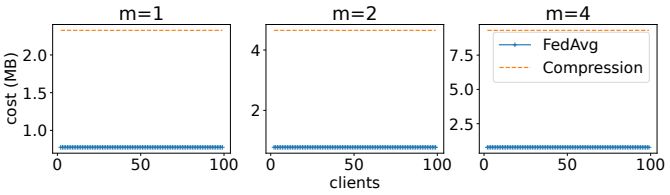


Figure 2: With a fixed model size of 794KB, the impact of other factors on individual client communication overhead.

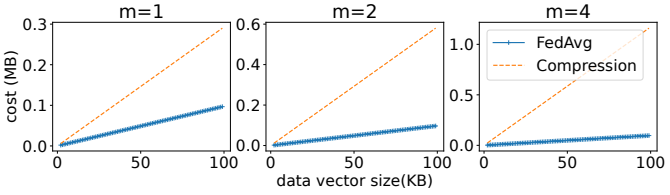


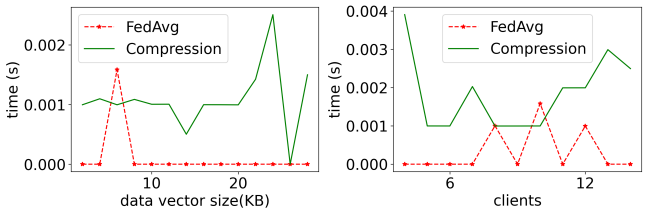
Figure 3: With a fixed number of clients at 20, the impact of other factors on individual client communication overhead.

Byzantine-robustness

In this section, we evaluate the Byzantine robustness of our scheme. To demonstrate the effectiveness of the proposed

Approach	Attack	Data Heterogeneity α = 0.1	α = 0.5	α = 0.9
FedAvg	**LF**	38.82 / 39.53 / 100.00 / 15.29	45.36 / 48.40 / 100.00 / 14.77	41.77 / 43.25 / 100.00 / 18.29
	Sine	12.78 / 40.22 / 100.00 / 26.52	14.28 / 43.71 / 100.00 / 28.21	43.61 / 48.86 / 100.00 / 15.51
FLtrust	**LF**	49.49 / 49.49 / 49.50 / 6.15	55.07 / 55.20 / 38.00 / 8.56	57.88 / 57.88 / 35.50 / 6.33
	Sine	47.28 / 47.28 / 100.00 / 8.11	57.95 / 57.95 / 100.00 / 6.45	59.59 / 59.59 / 100.00 / 5.55
IFCA	**LF**	48.18 / 49.01 / 100.00 / 15.83	37.57 / 42.50 / 100.00 / 20.04	49.63 / 52.03 / 100.00 / 13.05
	Sine	12.31 / 25.48 / 100.00 / 36.72	10.29 / 20.49 / 100.00 / 38.48	41.52 / 42.93 / 100.00 / 19.31
EBS-CFL	**LF**	62.13 / 64.58 / 7.00 / 1.73	65.29 / 66.30 / 0.00 / 0.25	65.88 / 65.90 / 0.50 / 2.10
	Sine	60.56 / 61.95 / 9.00 / 3.38	65.45 / 66.17 / 31.50 / 0.74	66.74 / 67.09 / 37.00 / 1.85

Table 3: Comparing the performance of different approaches under Byzantine attacks in a non-i.i.d. setting. Each cell in the table includes the final accuracy (FA), maximum accuracy during training (MA), Attack Success Rate (ASR = $\frac{\text{successful attackers}}{\text{total attackers}}$), and Attack Impact Rate (AIR = $\frac{2\cdot\text{NA}-\text{FA}-\text{MA}}{2\cdot\text{NA}}$), separated by "/". NA represents the accuracy achieved during training without any attacks. For the sake of convenience in presentation, the percentage symbols in the table are omitted.



(a) n=5 (b) l=10 KB

Figure 4: Study the impact of other variables on computational overhead while fixing the number of clients and model size.

EBS-CFL, we compare it against FLTrust, a state-of-the-art Byzantine-robustness algorithm. For attacks, we consider the representative Label-Flipping (LF) Attack (Biggio, Nelson, and Laskov 2012) and the advanced Sine Attack (Kasyap and Tripathy 2024). Additional comparisons with other methods, such as those in (Yin et al. 2018; Blanchard et al. 2017; Fang et al. 2020), are detailed in the appendix.

To simulate data heterogeneity under a non-i.i.d. setting, we use Dirichlet's distribution (Hsu, Qi, and Brown 2019), where α controls data dispersion (smaller α indicates higher heterogeneity). We evaluate both attacks at a high adversarial rate of 40%. The Sine Attack leverages cosine similarity to generate gradients that align closely with the central benign gradient but diverge from other client gradients. Our scheme protects server updates, rendering the AK-BSU assumption of Sine ineffective. Therefore, we conduct experiments under the realistic AK assumption and PC capability defined in Sine Attack. Results are shown in Table 3. Greater data heterogeneity reduces training accuracy and increases attack success rates. However, since heterogeneity inherently affects training accuracy, the relative impact of attackers does not rise significantly. Malicious gradients introduced by attackers are classified as low-quality and assigned minimal weight, reducing their influence. FLTrust exhibits weakened defense under heterogeneous data. Against Sine Attack, FLTrust can be fully compromised as data heterogeneity increases, making it harder to distinguish malicious gradients from legitimate but highly divergent ones. LF Attack follows a similar trend but achieves a lower success rate than Sine Attack, despite causing a greater overall impact. In contrast, our scheme significantly improves defense by effectively clustering and identifying malicious gradients in heterogeneous data. While accuracy declines when $\alpha = 0.1$, this is primarily due to the extreme heterogeneity rather than interference from Byzantine adversaries.

Security Analysis

In the security proof, we define the perturbation functions $\phi = x \cdot M$ and $\psi(x, R) = x + R$, which safeguard the privacy of matrix and vector operations, respectively. We also define the function $\varphi(\mathcal{S}) = \sum_{M \in \mathcal{S}} M$, where $\mathcal{S} \subset \{M_i\}_{i=1}^n$ and $\{M_i\}_{i=1}^n$ are mutually orthogonal matrices. The sum of these orthogonal matrices does not reveal information about individual matrices.

Using this framework, via theorems we prove that specific encryption mechanisms, such as VOMCA and SRFC, ensure the confidentiality and immutability of individual variables due to the incorporation of randomness. In secure aggregation and compression schemes, multiple randomizations and key transformations protect gradients and cluster identities from being leaked. Additionally, the SKT mechanism guarantees that neither keys nor variables can be inferred during transformations. Note that for detailed illustration of the proof process, please refer to the appendix.

Conclusion

In this paper, we have proposed an efficient and robust clustered federated learning secure aggregation framework, EBS-CFL. Specifically, we construct an encoding mechanism using matrix techniques, allowing the server to filter out poisonous gradients without knowing the clients' gradients and cluster identities. We then perform weighted aggregation based on the correlation between the gradients and the server updates. In addition, we provide detailed theoretical proofs of the correctness and the security of the approach, and analyze its efficiency. Finally, through extensive experiments, we demonstrate the efficiency, effectiveness, and robustness of our approach.

Acknowledgements

The authors would like to express their gratitude to Haiyong Bao for his insightful discussions and guidance. We also thank Cheng Huang and Hong-Ning Dai for their productive and stimulating conversations. Finally, our heartfelt thanks go to Menghong Guan and Hao Pan for their invaluable support and assistance.

The authors are very grateful as this work was supported in part by the National Natural Science Foundation of China (62072404); and in part by the Natural Science Foundation of Shanghai Municipality (23ZR1417700).

References

Biggio, B.; Nelson, B.; and Laskov, P. 2012. Poisoning Attacks against Support Vector Machines. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ICML'12, 1467–1474. Madison, WI, USA: Omnipress. ISBN 9781450312851.

Blanchard, P.; El Mhamdi, E. M.; Guerraoui, R.; and Stainer, J. 2017. Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, 118–128. Red Hook, NY, USA: Curran Associates Inc. ISBN 9781510860964.

Bonawitz, K. A.; Ivanov, V.; Kreuter, B.; Marcedone, A.; McMahan, H. B.; Patel, S.; Ramage, D.; Segal, A.; and Seth, K. 2016. Practical Secure Aggregation for Federated Learning on User-Held Data. *CoRR*, abs/1611.04482.

Cao, X.; Fang, M.; Liu, J.; and Gong, N. Z. 2021. FLTrust: Byzantine-robust Federated Learning via Trust Bootstrapping. In *28th Annual Network and Distributed System Security Symposium, NDSS 2021, virtually, February 21-25, 2021*. The Internet Society.

Chen, W.-N.; Choo, C. A. C.; Kairouz, P.; and Suresh, A. T. 2022. The Fundamental Price of Secure Aggregation in Differentially Private Federated Learning. In Chaudhuri, K.; Jegelka, S.; Song, L.; Szepesvari, C.; Niu, G.; and Sabato, S., eds., *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, 3056–3089. PMLR.

Chen, W.-N.; Ozgur, A.; and Kairouz, P. 2022. The Poisson Binomial Mechanism for Unbiased Federated Learning with Secure Aggregation. In Chaudhuri, K.; Jegelka, S.; Song, L.; Szepesvari, C.; Niu, G.; and Sabato, S., eds., *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, 3490–3506. PMLR.

Choi, B.; Sohn, J.; Han, D.; and Moon, J. 2020. Communication-Computation Efficient Secure Aggregation for Federated Learning. *CoRR*, abs/2012.05433.

Deng, L. 2012. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6): 141–142.

Fang, M.; Cao, X.; Jia, J.; and Gong, N. Z. 2020. Local Model Poisoning Attacks to Byzantine-Robust Federated Learning. In Capkun, S.; and Roesner, F., eds., *29th USENIX Security Symposium, USENIX Security 2020, August 12-14, 2020*, 1605–1622. USENIX Association.

Ghosh, A.; Chung, J.; Yin, D.; and Ramchandran, K. 2022. An Efficient Framework for Clustered Federated Learning. *IEEE Trans. Inf. Theory*, 68(12): 8076–8091.

Hsu, T. H.; Qi, H.; and Brown, M. 2019. Measuring the Effects of Non-Identical Data Distribution for Federated Visual Classification. *CoRR*, abs/1909.06335.

Jahani-Nezhad, T.; Maddah-Ali, M. A.; Li, S.; and Caire, G. 2023. SwiftAgg+: Achieving Asymptotically Optimal Communication Loads in Secure Aggregation for Federated Learning. *IEEE Journal on Selected Areas in Communications*, 41(4): 977–989.

Kadhe, S.; Rajaraman, N.; Koyluoglu, O. O.; and Ramchandran, K. 2020. FastSecAgg: Scalable Secure Aggregation for Privacy-Preserving Federated Learning. *CoRR*, abs/2009.11248.

Kasyap, H.; and Tripathy, S. 2024. Sine: Similarity is Not Enough for Mitigating Local Model Poisoning Attacks in Federated Learning. *IEEE Transactions on Dependable and Secure Computing*, 21(5): 4481–4494.

Krizhevsky, A.; and Hinton, G. 2009. Learning multiple layers of features from tiny images. *Handbook of Systemic Autoimmune Diseases*, 1(4).

Liu, X.; Li, H.; Xu, G.; Chen, Z.; Huang, X.; and Lu, R. 2021. Privacy-Enhanced Federated Learning Against Poisoning Adversaries. *IEEE Trans. Inf. Forensics Secur.*, 16: 4574–4588.

McMahan, H.; Moore, E.; Ramage, D.; Hampson, S.; and Arcas, B. 2016. Communication-Efficient Learning of Deep Networks from Decentralized Data. *arXiv: Learning,arXiv: Learning*.

Sattler, F.; Müller, K.-R.; and Samek, W. 2021. Clustered Federated Learning: Model-Agnostic Distributed Multitask Optimization Under Privacy Constraints. *IEEE Transactions on Neural Networks and Learning Systems*, 32(8): 3710–3722.

Vahidian, S.; Morafah, M.; Wang, W.; Kungurtsev, V.; Chen, C.; Shah, M.; and Lin, B. 2023. Efficient Distribution Similarity Identification in Clustered Federated Learning via Principal Angles between Client Data Subspaces. In Williams, B.; Chen, Y.; and Neville, J., eds., *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, 10043–10052. AAAI Press.

Wan, W.; Hu, S.; Lu, J.; Zhang, L. Y.; Jin, H.; and He, Y. 2022. Shielding Federated Learning: Robust Aggregation with Adaptive Client Selection. In Raedt, L. D., ed., *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, 753–760. ijcai.org.

Yin, D.; Chen, Y.; Kannan, R.; and Bartlett, P. 2018. Byzantine-Robust Distributed Learning: Towards Optimal

Statistical Rates. In Dy, J.; and Krause, A., eds., *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, 5650–5659. PMLR.

Zhang, Y.; Liu, D.; Duan, M.; Li, L.; Chen, X.; Ren, A.; Tan, Y.; and Wang, C. 2023a. FedMDS: An Efficient Model Discrepancy-Aware Semi-Asynchronous Clustered Federated Learning Framework. *IEEE Transactions on Parallel and Distributed Systems*, 34(3): 1007–1019.

Zhang, Z.; Li, J.; Yu, S.; and Makaya, C. 2023b. SAFE-Learning: Secure Aggregation in Federated Learning With Backdoor Detectability. *IEEE Trans. Inf. Forensics Secur.*, 18: 3289–3304.

Zheng, Y.; Lu, R.; Zhang, S.; Guan, Y.; Shao, J.; and Zhu, H. 2022. Toward Privacy-Preserving Healthcare Monitoring Based on Time-Series Activities Over Cloud. *IEEE Internet of Things Journal*, 9(2): 1276–1288.

Zhu, L.; Liu, Z.; and Han, S. 2019. Deep Leakage from Gradients. In Wallach, H.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.