mathematics

MDPI

Article

CVFL: A Chain-like and Verifiable Federated Learning Scheme with Computational Efficiency Based on Lagrange Interpolation Functions

Mengnan Wang [1], Chunjie Cao [2,3,*], Xiangyu Wang [4,*], Qi Zhang [5], Zhaoxing Jing [2,3], Haochen Li [2,3] and Jingzhang Sun [2,3,*]

[1] School of Computer Science and Technology, Hainan University, Haikou 570228, China; 21210812000014@hainanu.edu.cn
[2] School of Cryptology, Hainan University, Haikou 570228, China; 21110839000003@hainanu.edu.cn (Z.J.); 22220854000199@hainanu.edu.cn (H.L.)
[3] School of Cyberspace Security, Hainan University, Haikou 570228, China
[4] School of Network and Information Security, Xidian University, Xi'an 710126, China
[5] Faculty of Data Science, City University of Macau, Macau SAR, China; qizhang@cityu.mo
* Correspondence: caochunjie@hainanu.edu.cn (C.C.); wangxiangyu01@xidian.edu.cn (X.W.); jingzhangsun@hainanu.edu.cn (J.S.)

Abstract: Data privacy and security concerns have attracted significant attention, leading to the frequent occurrence of data silos in deep learning. To address this issue, federated learning (FL) has emerged. However, simple federated learning frameworks still face two security risks during the training process. Firstly, sharing local gradients instead of private datasets among users does not completely eliminate the possibility of data leakage. Secondly, malicious servers could obtain inaccurate aggregation parameters by forging or simplifying the aggregation process, ultimately leading to model training failures. To address these issues and achieve high-performance training models, we have designed a verifiable federated learning scheme called CVFL, where users exist in a serial manner to resist inference attacks and further protect the privacy of user dataset information through serial encryption. We ensure the secure aggregation of models through a verification protocol based on Lagrange interpolation functions. The serial transmission of local gradients effectively reduces the communication burden on cloud servers, and our verification protocol avoids the computational overhead caused by a large number of encryption and decryption operations without sacrificing model accuracy. Experimental results on the MNIST dataset demonstrate that, after 10 epochs of training with 100 users, our solution achieves a model accuracy of 90.63% for MLP architecture under IID data distribution and 87.47% under non-IID data distribution. For CNN architecture, our solution achieves a model accuracy of 96.72% under IID data distribution and 93.53% under non-IID data distribution. Experimental evaluations corroborate the practical performance of the presented scheme with high accuracy and efficiency.

Keywords: deep learning; federated learning; privacy protection; verifiable aggregation

MSC: 68T07

check for updates

Citation: Wang, M.; Cao, C.; Wang, X.; Zhang, Q.; Jing, Z.; Li, H.; Sun, J. CVFL: A Chain-like and Verifiable Federated Learning Scheme with Computational Efficiency Based on Lagrange Interpolation Functions. *Mathematics* **2023**, *11*, 4547. https://doi.org/10.3390/math11214547

Academic Editors: Jianguo Chen, Jinyu Tian, Ji Wang and Jonathan Blackledge

Received: 13 September 2023
Revised: 17 October 2023
Accepted: 2 November 2023
Published: 4 November 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Deep learning (DL) mainly utilizes neural networks to process and analyze data, while it typically requires a large amount of data for training the model [1]. The use of DL is usually constrained by issues such as data privacy concerns, commercial information confidentiality between companies and users' individual datasets varying significantly in terms of quantity and features [2]. In order to address the above-mentioned challenges, federated learning (FL), a novel distributed machine learning framework, was first proposed by Google in 2016 [3].

High-performance deep learning networks require the participation and support of a large number of participants and substantial amounts of data. In centralized neural networks, each user needs to upload their entire personal dataset to the server for unified training to obtain a deep learning model. However, in real-world environments, there is a significant amount of data that involves sensitive user information, such as medical or commercial data. With the advancement of digitization, intelligence and networking, the importance of data privacy and security has become increasingly prominent. Centralized deep learning models are no longer suitable for the current data security environment. In federated learning, when multiple participants have their own datasets, model training is accomplished by aggregating the local model parameters of each party without sharing the raw data [4,5]. This distributed collaborative framework helps protect data privacy and can improve model performance through cooperative training. When data are scattered among multiple organizations, federated learning avoids the problems of data transmission and centralized storage. It only requires training local models on each terminal device, performing lightweight model updates and uploading the local model updates (such as gradients or weights) to the central server for aggregation and updates. This reduces communication and computational overhead and is equally applicable in resource-constrained environments. This privacy-preserving mechanism gives federated deep learning an advantage in applications involving sensitive data and high compliance requirements.

Nevertheless, sharing only the model gradient information still poses a high risk of data privacy leakage [6]. Malicious attackers can use inference attacks [7] to obtain the original data information of participating parties. Furthermore, compared to centralized deep learning methods, federated deep learning may involve users with different data distribution, feature sets and class distributions. By aggregating these heterogeneous data, federated deep learning can obtain a more comprehensive data perspective, making the model more robust and capable of generalizing, which is beneficial for overcoming overfitting and improving model performance.

On the other hand, the central server of the model is not entirely trustworthy. In more realistic research scenarios [8,9], it has been found that a malicious server may obtain inaccurate intermediate parameters by forging aggregation parameters or simplifying the model aggregation processes and then returning them to the participants. To address this issue, it is necessary to ensure the correctness of the aggregation parameters during the iteration process, i.e., ensuring the data integrity of the model aggregation parameters. Therefore, protecting the privacy of local gradients and verifying the correctness of aggregated gradients are crucial for implementing a secure and reliable FL model.

Currently, there have been many privacy-preserving federated learning (PPFL) methods [10,11] proposed. However, FL involves the iteration and synchronization of model parameters across multiple devices, as well as the exchange of large amounts of data. Further, secure aggregation also requires complex computation and verification of encrypted information. Therefore, the time-consuming and communication overhead of FL is relatively huge [12,13], especially in cases with a large number of devices. Such overhead is often impractical for FL scenarios with plenty of resource-constrained users collaborating in training. It is of great significance to effectively optimize computing system performance and improve computational efficiency for FL.

In this study, we proposed a verifiable FL scheme, a chain-like and verifiable federated learning with computational efficiency (CVFL), that is similar to the zero-noise in serial [14]. Firstly, during the local model training process, all users train on their respective datasets simultaneously, obtain their respective local gradients for the current round and then pass the local gradients. A mask is added to the first user's local gradient to pass to the next user, who continues to pass backwards as a mask for the next user's local gradient and so on until the last user passes the sum of the local gradients mixed with a mask to the cloud server. Local gradient passing to the server starts at the end of training for all users, so the next user's training does not have to wait for the previous user's local gradient;

this form of crosstalk only serves to protect the local gradient during its passing and does not affect the training of the users' respective models. The server removes the mask to obtain the accurate local gradient and iterate, which ensures the privacy of each local model gradient in the sharing process but also does not come at the cost of reducing the model accuracy. In addition, the use of the Lagrangian interpolation verification method reduces the computational cost and time overhead in the model training process as much as possible under the premise of guaranteeing the correctness of the aggregated gradients, which can be better applied in real-world environments. The main contributions of this paper are as follows:

1. We proposed a verifiable chain-like FL scheme based on Lagrange interpolation, which significantly reduced the communication and computation overhead as compared to existing verifiable FL schemes. It was suitable for real-world environments where device computing resources were limited but there were multiple data providers.
2. The proposed FL model protected the privacy of users' local data while verifying the data integrity of the server's aggregated gradients. The protocol used a pseudo-random generator that generated a random sequence based on timestamps to ensure its reliability. Moreover, the protocol did not affect the performance of the original FL model.
3. This algorithm involves a trusted agent (TA) and serialized user communication. A time-dependent mask is sent to the first user and carried over along the communication chain to obtain a masked global gradient, which is sent to the TA. Lagrange interpolation is then conducted at the TA, and the approximated function is sent to the central server to verify its computation ability. The mask is then removed from the aggregated gradients before the next learning iteration.
4. We conducted an effectiveness analysis of the protocol and implemented the model. We performed experimental evaluations on the MNIST dataset using a multilayer perceptron (MLP) and convolutional neural network (CNN) to assess computation overhead and accuracy. The results demonstrate that our model achieves high accuracy and efficiency.
5. This approach saves the need for traditional encryption and decryption steps by leveraging Lagrange interpolation. Compared with other approaches, such as differential privacy, there is no need to sacrifice the global model performance by adding noises to the gradients.

The remaining sections of this article are organized as follows: In the second section, we introduce related work. In the third section, we present preliminary work. In the fourth section, we provide a detailed description of the proposed protocol scheme. In the fifth and sixth sections, we analyze the security of the model and the system's performance, respectively. In the seventh section, we discuss the results of our experiments. In the eighth section, we conclude and discuss future work.

2. Related Works

Federated learning, an emerging distributed learning approach replacing traditional centralized learning, consists of two components: a central server and users. After training their respective models on local datasets, users only upload their local gradients to the central server. The server aggregates all the local gradients using an algorithm, such as Federated Averaging (FedAVG) [15], which is widely used in federated learning. FedAVG is based on the principles of stochastic gradient descent and generates a global gradient by averaging the gradients uploaded by participants. This algorithm is efficient, scalable and suitable for large-scale and heterogeneous federated learning environments. It has been widely applied in various fields, including natural language processing, computer vision and edge computing. This distributed iterative gradient aggregation process allows participants to collectively improve the performance of the global model while partially avoiding the privacy issues associated with sharing private data in centralized learning. Clients ideally train their local models while the server aggregates and distributes the global model [16].

However, in real network environments, ensuring the trustworthiness of the central server is challenging. Compromised servers can extract local data from gradients, enabling attacks like inversion and inference attacks [7]. Attackers can indirectly infer dataset membership by analyzing client-uploaded gradients. Therefore, it is clear that relying solely on model updates is insufficient for user privacy protection [9]. To enhance the privacy of user data in federated learning frameworks, several federated learning solutions have been proposed to protect data privacy. Most of these solutions for preserving data privacy can be categorized into three implementation technologies: homomorphic encryption (HE), secure multi-party computation (SMC), and differential privacy (DP) techniques. Zhang et al. [17] proposed a FL scheme using multiplicative HE and secret sharing for the ElGamal cryptosystem. Li et al. [18] introduced a multi-key system based on fully HE. However, these schemes are limited to semi-honest servers and involve expensive exponential computations during encryption and decryption, leading to low efficiency. HE [7] technology is applied to the process of transmitting local gradients. It enables gradient aggregation and model updates to be performed while the data are encrypted, ensuring privacy protection for gradient information. However, the complex cryptographic primitives, encryption/decryption operations, homomorphic computations and the increase in ciphertext size associated with HE can impose additional computational burden, impacting the efficiency of federated learning. Furthermore, HE may impose certain limitations on the required computation operations and model update methods in federated learning. Tueno et al. [19] introduced a non-interactive proof encryption protocol based on secret sharing, demonstrating that these protocols can be employed to construct an effective privacy-preserving system for releasing aggregated statistical data while ensuring security in malicious environments. Secure multi-party computation (SMC) [20] allows multiple participants to collectively participate in model training and computation operations without revealing their private inputs. By using encryption protocols, each participant can obtain only the computed results during the training process without gaining knowledge of other participants' inputs. Through the use of SMC, federated learning can achieve a higher level of data privacy protection. However, it also faces challenges in terms of computational efficiency, communication complexity and protocol selection. In practical applications, a trade-off between privacy protection and computational overhead needs to be carefully considered. To protect the privacy of users' local training data in the FL framework, researchers have proposed the NbAFL [21] framework. This adjusts noise variance based on differential privacy with better security, but has computational challenges. DP [22] aims to add noise to the private data of each participant. Subsequently, participants conduct local model training based on the perturbed data, and the aggregated parameters are used to complete the model aggregation. The individual data of participants is obfuscated by the added noise, thereby achieving data privacy protection. However, the introduction of noise can introduce a certain level of model bias, leading to a reduction in model accuracy. Therefore, when balancing privacy protection and model accuracy, it is necessary to carefully select differential privacy parameters and appropriate noise levels to meet specific application requirements.

While the aforementioned methods for protecting data privacy can prevent attackers from obtaining dataset information by stealing participants' uploaded local gradients, there is still a security vulnerability. A malicious server could potentially influence model training by forging aggregated gradients. To enhance the security of federated learning, it is necessary to verify whether the server in the distributed learning environment has successfully performed the aggregation process [23], ensuring data verifiability. Verifiability [23] refers to the ability of one party in the federated learning protocol to prove to other parties that it has correctly executed the intended task without deviation. VerifyNet [24], proposed by Xu et al., ensures secure updates of gradient information through HE and secret sharing. However, the security protocols used by VerifyNet, which combine HE and SMC, require additional computational resources or bandwidth assistance to implement the protocol. This imposes significant computational and communication overhead on the

entire training process of the model, which continues to increase with more participants. In federated learning, there is a consideration for resource limitations due to the communication and computation between multiple participants. Each participant can only utilize their limited local computing resources for model training. The substantial computational and communication overhead of VerifyNet may lead to issues such as communication latency, insufficient resources and prolonged training time, thereby affecting the effectiveness of model training. As a result, it is challenging to apply VerifyNet in federated deep learning environments. To address a limited bandwidth and high-dimensional gradient, Guo et al. proposed the verifiable aggregation protocol Verifl [25] for efficient and verifiable aggregation. However, Ren et al. demonstrated that VerifyNet and Verifl protocols still have security vulnerabilities [26]. While these solutions enhance model security, most of them come at the cost of model performance and efficiency, making them less practical under resource constraints. In situations with limited resources, these methods may not be suitable for federated deep learning. Additionally, the complexity of these verification methods can increase model training time and impose additional computational overhead, conflicting with the requirements of certain real-time application scenarios that demand quick model updates and responsiveness.

Therefore, for practical applications of federated learning, it is important to consider resource limitations and security requirements. It is necessary to choose appropriate methods that strike a balance between security and performance/efficiency. Some researchers have also focused on improving model performance and efficiency and proposed joint learning frameworks, such as VFChain [27], which is based on blockchain systems; a robust and lightweight verification-based secure aggregation scheme, VSA [28]; and a verifiable and privacy-preserving federated learning approach, VFL [29], which is specifically for big data in industrial Internet of Things. However, they still have significant computational and communication overhead.

In this study, we aimed to implement a lightweight, verifiable FL solution based on Lagrange interpolation with minimal time and computational overhead. We also proposed improvements to make it suitable for FL models with a larger number of users. By utilizing a serial transmission approach for aggregation, we could reduce the communication frequency between users and the server, enhance model generalization and minimize the number of model iterations. This model is well-suited for practical scenarios with limited resources.

3. Preliminary

In this section, we provide a brief overview of the serial transmission-based federated learning and Lagrange interpolation methods used in our proposed solution. We then introduce the threat model and corresponding design objectives.

3.1. Federated Learning (FL)

The proposed FL framework consists of a central server, a trusted third party, and n users $U_i(i = 1, 2, \ldots, n)$. Each user has their own dataset $D_i(i = 1, 2, \ldots, n)$ and trains their local model using the stochastic gradient descent method [30]. All users are stored in a serial manner within the FL framework. Under the masking generated by the trusted third party, each user sends their local gradient information to the cloud server. The cloud server aggregates the local gradients uploaded by users using the FedAVG algorithm [31] to obtain a global model:

$$g_t = \sum(D_i * w_i) / \sum N_i \quad (1)$$

where g_t is the global model of the t-th round and w_i is the local model uploaded by the i-th user.

The central server (CS) returns the gradient parameters to the user. The user then uses optimization algorithms to calculate the loss function based on the global gradient obtained in the previous round:

$$L_i(w_i) = \sum(y_i - f(x_i, w_i))^2 \quad (2)$$

where y_i is the label of the i-th user, x_i is the input of the i-th user and $f(x_i, w_i)$ is the output for the i-th participant.

The participants update their local model parameters using the formula below until the entire model tends to converge.

$$w_{i+1} = w_i - \eta * \nabla L_i(w_i) \tag{3}$$

where η is the learning rate and $\nabla L_i(w_i)$ is the gradient of the loss function with respect to the i-th user.

3.2. Lagrange Interpolation

Mathematically, given $k + 1$ data points: $(x_0, y_0), (x_1, y_1), \ldots, (x_k, y_k)$, the Lagrange interpolation method [32] can construct a polynomial function:

$$L(x) = \sum_{m=0}^{k+1} y_m h_m(x) \tag{4}$$

In this formula, each $h_m(x)$ is the Lagrange basic polynomial, whose expression is:

$$h_m(x) = \prod \frac{x - x_m}{x_m - x_n} = \frac{x - x_0}{x_m - x_0} \cdot \frac{x - x_1}{x_m - x_1} \cdot \ldots \cdot \frac{x - x_k}{x_m - x_k} \tag{5}$$

$h_m(x)$ could further expressed by:

$$h_m(x) = \begin{cases} 1, x = x_m \\ 0, x = x_n (n \neq m) \end{cases} \tag{6}$$

Using the Lagrange basic polynomial makes it easy to obtain the Lagrange interpolation polynomial, which can approximate an unknown function based on discrete data without the need to solve equations by supplementing and interpolating continuous functions. This allows the function to pass through a set of given discrete data points.

In practical applications, the Lagrange interpolation polynomial suffers from the problem of ill-conditioning when using high-degree interpolation [33]. Intuitively, we might think that using more interpolation points and higher-degree polynomials would lead to more accurate predictions of a function. However, experiments have shown that high-degree interpolation not only results in complex calculations but also leads to less accurate predictions, known as the Runge phenomenon [34].

3.3. Threat Model

We consider an original federated learning model, as shown in Figure 1. Although the original federated learning model, where participants do not share their respective datasets but train models locally, provides some level of privacy protection for participants' data, it still has security vulnerabilities. Multiple users train their own deep learning models on their local datasets and then upload the resulting local model parameters to a central server. If an attacker manages to steal the gradient information during the process of uploading their local gradients, they can still infer specific data information through inference attacks, posing a security vulnerability (Q1). After receiving all the local gradient information, the central server aggregates all the local gradients using principles such as stochastic gradient descent and weighted averaging to obtain global model parameters. The central server then sends the updated global model parameters back to the users for the next round of model training. However, if a malicious server fabricates false aggregation parameters and returns them to all users for the next round of model iteration, it can lead to failed model training, posing a security vulnerability (Q2). We assume that there is no possibility of collusion between the server and users, and that users will not intentionally tamper with

the gradient information of their local models. Therefore, collusion attacks between the server and users, as well as user poisoning attacks, are not considered in this study.

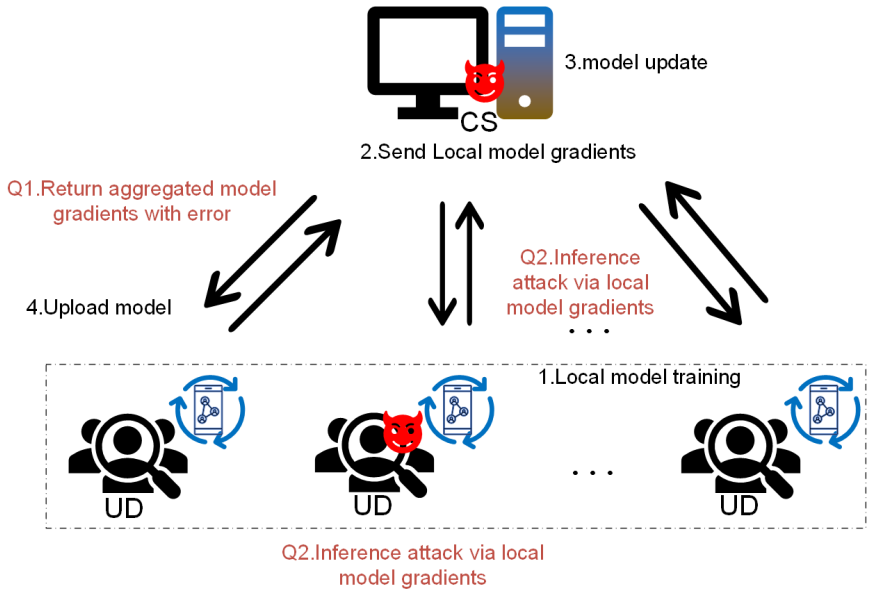


Figure 1. Security risks in federated learning. One of them is that the local gradient shared by federated learning may still have the risk of leaking user data information due to inference attacks. The second is that the central server may provide wrong aggregated gradients by faking or simplifying the model, which ultimately leads to model convergence failure.

3.4. Design Goal

Our goal is to design a secure and efficient verifiable FL solution to address the two security vulnerabilities. This solution aims to protect the privacy of user gradients and ensure secure server aggregation while minimizing the computational overhead of the overall implementation and reducing the communication burden on the central server without compromising accuracy. The specific design objectives are as follows:

1. Security: By protecting the locally aggregated gradient information uploaded by users, it prevents adversaries from inferring users' data privacy through inference attacks. The verification of the aggregation result ensures the correctness of the training process.
2. Accuracy: Our approach balances security guarantees and training result accuracy. Aggregation and validation protocols must maintain the original training accuracy, preserving model availability and utility.
3. Efficiency: Our approach minimizes costs while prioritizing the above objectives. We aim to minimize cryptographic primitive usage due to computational and communication overhead.

4. The Proposed CVFL Scheme

In this section, we first provide an overview of our proposed solution, CVFL, and its functionalities. Then, we present the specific technical details of our solution.

4.1. Overview of CVFL Scheme

As shown in Figure 2, there is a trusted agency (TA), a cloud server (CS) and n users (UD) in the proposed framework [35].

1. TA: The TA is responsible for generating a pseudo-random number (PRN) using a pseudo-random generator as a mask to obfuscate the user gradient information and pass it to the first user. In order to achieve secure aggregation, the TA also needs to generate a pseudo-random backup sequence $a = [a_1, a_2, \ldots, a_m]$ and broadcast it to

all users and the cloud server. In addition, the TA is responsible for splitting the sum of locally computed gradients under the mask before verification and for sending the mask to the cloud server only after user verification is trusted. The TA is a completely trusted third party, responsible only for the process of generating and transmitting the PRN, and remains closed throughout the model training process.

2. CS: The cloud server is responsible for generating and delivering the initial information of the model. During each round of training, if the user verification is successful, the cloud server uses the FedAVG to obtain the new global model for the next round and sends its gradients to the users to continue the next round of model training until the model converges.
3. UD: A set of participants (n users) concurrently train a DL model locally through feedforward neural networks and obtain their local gradients. Taking the example of MLP on the MNIST and CIFAR-10 datasets, each sample in the MNIST dataset is a 2828 grayscale image. Therefore, the number of input layer nodes in the feedforward neural network is 784, where each pixel value is treated as a feature input to the MLP model. The network consists of two hidden layers, each with 200 units, and ReLU is used as the activation function. The output layer has 10 nodes, representing the ten possible classes (0–9) in the dataset. Similarly, in the CIFAR-10 dataset, each sample is a 3232-pixel image with RGB channels. Hence, the number of input layer nodes in the feedforward neural network is 3072, where each pixel value is treated as a feature input to the MLP model. The network consists of two hidden layers, each with 200 units, and ReLU is used as the activation function. The output layer has ten nodes, representing the ten possible classes (zero to nine) in the dataset [35]. Then, users send the local gradient information and validation request with mask to TA and use a PRN generated by timestamping as seed to validate the information. After validation is successful, users receive the aggregated model gradient information from the server and continue the local model training update in the next round.

The entire process can be divided into four stages: initialization, local model training, secure aggregation and global update. In the following sections, we will provide detailed explanations of the technical details for each stage.

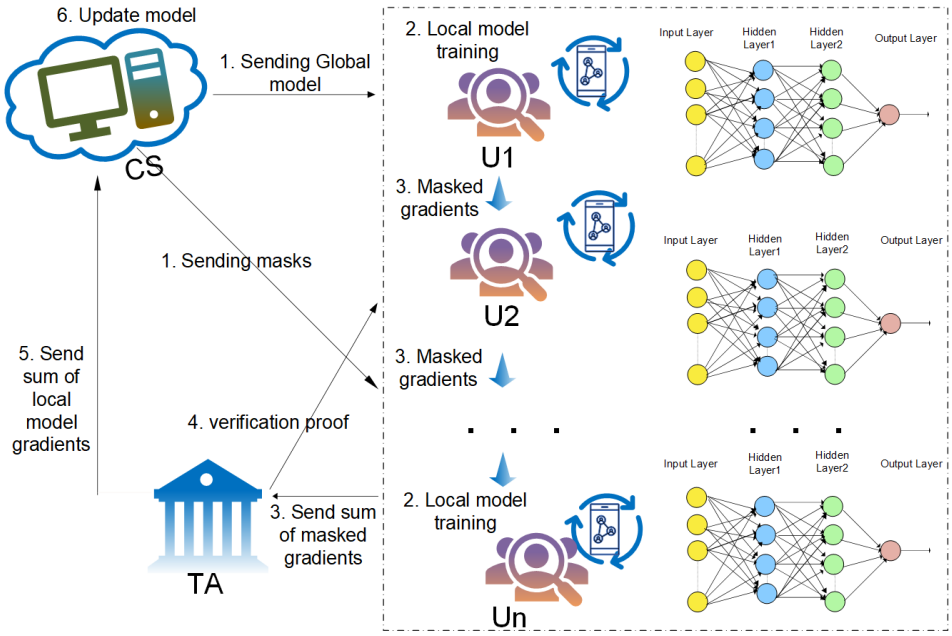


Figure 2. The architecture and system model of CVFL scheme. The CVFL model consists of two parts: passing the encrypted local model gradient and verifying the central server aggregation gradient. The model consists of the user UD, the central server CS and the trusted third party TA together.

4.2. Initialization Phase

In the initialization stage, FL needs to complete the following main tasks: First, based on factors such as the distance between users or data distribution, all users are connected in a user chain. Second, the TA is responsible for using a PRN generator to generate a PRN as a mask and a backup sequence "a", ensuring the confidentiality of the mask to the CS. In our proposed scheme, the TA is a fully trusted third party and is only responsible for generating and transmitting the mask PRN. Additionally, for simplicity, we assume there are n users represented by U_i $(i = 1, 2, \ldots, n)$ (where $n \geq 2$). In the initialization stage, the CS needs to pass the initial parameters g_0 of the model to the first user, U_1. This way, all the users participating in federated learning are connected in series according to certain conditions, and the task of transferring the gradient information is controlled between two users and the cloud server, which reduces the large amount of communication burden between the cloud server and the users due to the interaction of the gradient information and also reduces the training time of the model.

4.3. Local Training Phase

First, the CS broadcasts the initial model parameters g_0 to all users. The TA also sends the mask PRN^1 for the first round of training to the user U_1. Each user U_i trains their local model parameters ω_i^1 using backpropagation and stochastic gradient descent on their own local dataset D_i. Next, as shown in Figure 2, the encrypted local gradient is obtained by adding the mask to the local gradient:

$$\omega_1^{1'} = \omega_1^1 + PRN^1 \tag{7}$$

These encrypted gradients are then passed on to the next user as the mask for the gradient information. This process continues, with each user transmitting the masked local gradient information to the next user:

$$\omega_i^{1'} = \omega_i^1 + PRN^1 \tag{8}$$

until the last user obtains the encrypted local gradient information under the mask:

$$\omega_n^{1'} = \omega_1^1 + \omega_2^1 + \ldots + \omega_n^1 + PRN^1 = \sum_{i=1}^{n} \omega_i^1 + PRN^1 \tag{9}$$

The last user sends $\omega_n^{1'}$ and a verification request to the TA. All users send a pseudo-random number generated with the same timestamp as the seed to the TA for the CS's verification. We will explain the specific verification details in the next section. Algorithm 1 provides the technical details for the chained FL. Storing the FL framework of the users in a chained manner not only protects the privacy of each dataset from leakage, but the presence of PRN^t also avoids the risk of leakage of local gradients during the transmission process. Meanwhile, the server removes PRN^t, which is equivalent to a differentially private mechanism with ε equal to zero. This ensures secure aggregation without compromising the overall performance of the model.

This chained form, to some extent, simplifies the complex encryption and decryption processes, reducing the complexity of ciphertext computation. It also ensures the privacy of gradient information between users and avoids leakage of gradient information during transmission. This achieves privacy-preserving gradient uploading.

Algorithm 1: Local Training Phase

input : $UD = \{U_1, U_2, \ldots, U_n\}$, $Datasets = \{D_1, D_2, \ldots, D_n\}$, random number of $t-th$ round PRN^t, number of global iterations T, initial model g_0, maximum number of epochs E, learning rate η, loss function $L(g, D)$

output: sum of encrypted local gradient information $\omega_n^{t'}$

CS sends g_0 to U_1
for $t = 1, 2, \ldots, T$ **do** // global iteration
  TA sends PRN^t to U_1 **for** $U_1 \in UD$ **do** // UD
    for $r = 1, 2, \ldots, E$ **do** // epoch
      $\omega_i^{r+1} \leftarrow \omega_i^r - \eta \Delta L(g_r, D_i)$ // SGD
    end
    $\omega_1^{t'} = \omega_1^t + PRN^t$ // masking
    $\omega_{i+1}^{t'} = \omega_i^{t'} + \omega_{i+1}^t \; \omega_n^{t'} = \omega_{n-1}^{t'} + \omega_n^t = \omega_1^t + \omega_2^t + \ldots + \omega_n^t + PRN^t$
  end
end

4.4. Secure Aggregation Phase

During the t-th round of model training, the TA receives the verification request from the user U_i. Then, the received value is randomly split into m-1 vectors $[v_1, v_2, \ldots, v_m - 1]$ such that the sum of the vectors is equal to vector $\omega_n^{t'}$. These numbers, along with the pseudo-random number $v_m^t = PRN^t$ sent by the user, a vector of the same dimension as v_i, form a sequence $v = [v_1^t, v_2^t, \ldots, v_m^t]$. This sequence, together with the auxiliary sequence $[a_1, a_2, \ldots, a_m]$ generated during the initialization phase, forms a set of m points. According to the Lagrange interpolation principle, there exists one and only one function equation that satisfies the above set of points. To avoid the Runge phenomenon from making the verification method impractical in a FL environment with a large number of users, we appropriately reduce the degree of the function equation to obtain the verification function $Q(x)$. The TA sends the verification function $Q(x)$ to the cloud server, which computes $Q(a_1) + Q(a_2) + \ldots + Q(a_(m-1))$ and $Q(a_m)$ and sends $Q(a_m)$ to the verifying user U_i. The CS aggregates the masked local gradients to obtain the global gradient G_1^t. The verifying user then verifies $Q(a_m)$ and v_m^t.

$$Q(a_m) = v_m^t \tag{10}$$

If the verification is successful, the CS aggregates the local gradients after removing the mask to obtain the global gradient G_2^t. The user uses (ω_n^t, G_1^t) and $(\omega_n^{t'}, G_1^t)$ to construct two Lagrange base functions for verification, obtaining a Lagrange interpolation polynomial $F(x)$. The user verifies by substituting $\omega_n^(t)$ to obtain the predicted value of the aggregated gradient and compares it with G_2^t. If they are consistent, the aggregation of this round of training is successful, and the model continues to the next iteration. If they are not consistent, the training of this round is terminated. The technical details of secure aggregation are provided in Algorithm 2.

Algorithm 2: Secure Aggregation Phase

input : integer sequence $a = \{a_1, a_2, \ldots, a_m\}$, random number of $t - th$ round PRN^t, number of global iterations T, sum of encrypted local gradient information $\omega_n^{t'}$

output: a secure global gradient G_n^t

for $t = 1, 2, \ldots, T$ **do** `// global iteration`
  TA split $\omega_n^{t'}$ to $\{v_1, v_2, \ldots, v_{m-1}\}$
  $v_m^t = PRN^t$
  $v = \{v_1, v_2, \ldots, v_{m-1}\}$
  Lagrange interpolation $Q(x) \leftarrow ((a_1, v_1^t), (a_2, v_2^t), \ldots, (a_m, v_m^t))$
  if $Q(a_m) == v_m^t$ **then**
    $\omega_n^{t'} = Q(a_1) + Q(a_2) + \ldots + Q(a_m - 1)$
    $y_2^t = FedAVG(\omega_n^{t'})$
    TA sends PRN^t to CS
    $\omega_n^t = \omega_n^{t'} - PRN^t$
    Lagrange interpolation $F(x)$
    $y_1^t = F(\omega_n^t)$
    $y_1^{t'} = FedAVG(\omega_n^t)$
  else
    break
  end
  if $y_1^{t'} == y_1^t$ **then**
    $G_n^t = FedAVG(\omega_n^t)$
  else
    break
  end
end

4.5. Global Update Phase

In the $t + 1$ round of iteration, user U_i received the updated aggregated model parameters. They repeat tasks 4.3 and 4.4, continuing the training and updating of their local models, and obtain a new round of local model gradients with the added mask PRN^t. After successful verification by user U_i, the cloud server performs aggregation of the global model until the global model converges.

5. Security Analysis

5.1. Protection of Gradient Information

Our CVFL scheme is secure against both honest-but-curious users and the cloud server. Neither an individual user who is honest but curious nor a collusion between users and the cloud server can extract personal data information of participating users from the local gradient information transmitted during the model training process.

Theorem 1. *Assuming that the local gradient information of users is masked to obtain $\omega_n^{'}$, neither an individual user nor the cloud server can obtain the personal gradient information of any user except themselves from $\omega_n^{'}$.*

Proof of Theorem 1. According to Equations (7) and (8), the local gradient information of users is masked by adding a mask in the t-th round of the iteration, resulting in:

$$\omega_n^{t'} = \omega_n^t + PRN^t = \omega_1^t + \omega_2^t + \ldots + \omega_n^t + PRN^t = \sum_{i=1}^{n} \omega_i^t + PRN^t \tag{11}$$

$$\omega_i^{t'} = \omega_i^t + PRN^t \tag{12}$$

□

Therefore, we can conclude that, after the local gradient information of users is masked, neither a curious but honest user nor a colluding cloud server can extract personal data information from the local gradient information transmitted during the model training process. Our CVFL scheme is secure against both curious but honest users and cloud servers.

5.2. Verifiability and Correctness

Theorem 2. *Each user can verify the aggregation result. If a malicious or dishonest cloud server attempts to forge the aggregation result, it will fail the verification process. And if the security protocol can guarantee the secure aggregation of gradients by the server, users can obtain the correct global gradients to proceed with the next round of model training. The general process is as follows: After receiving and successfully verifying the authentication information from the cloud server, assuming the cloud server is trusted and not malicious, a trusted third party can send the combined local gradients and pseudo-random numbers to the cloud server. The trusted cloud server can then obtain the aggregated gradient information using the FedAVG algorithm based on the principles of stochastic gradient descent. The cloud server returns the correct aggregated gradient to the users, who can continue updating their models based on this correct aggregated gradient.*

Proof of Theorem 2. Assuming that the cloud server securely aggregates the local gradients to obtain the correct global model gradient in the t-th round of iteration, we can verify the user's validity by using Equations (4) and (5) to obtain $Q(x)$ [32].

$$Q(x) \leftarrow ((a_1, v_1^t), (a_2, v_2^t), \ldots, (a_{m-1}, v_{m-1}^t), (a_m, v_m^t)) \tag{13}$$

Substituting $a_1, a_2, \ldots, a(m-1)$ for $Q(x)$,

$$\begin{aligned} \sum_{i=1}^{m-1} Q(a_i) = Q(a_1) + Q(a_2) + \ldots + Q(a_{m-1}) = v_1^t + v_2^t + \ldots + v_{m-1}^t \\ = \omega_n^{t'} = \omega_1^t + \omega_2^t + \ldots + \omega_n^t + PRN^t \end{aligned} \tag{14}$$

If the cloud server forges the aggregated gradient, we should have

$$\begin{aligned} \omega_1^t + \omega_2^t + \ldots + \omega_n^t + PRN^t + \Delta\omega = \omega_n^{t'} + \Delta\omega \\ = v_1^t + v_2^t + \ldots + v_{m-1}^t + \Delta\omega = \textstyle\sum_{i=1}^{m-1} Q'(a_i) \end{aligned} \tag{15}$$

According to Equation (9), in order to deceive the user and achieve successful verification, it is necessary to

$$Q'(a_m) = v_m^t = Q(a_m) \tag{16}$$

where v_m^t is a PRN as the seed with strong randomness and kept confidential from all entities except for itself, in order for the above equation to always hold, it is necessary that

$$Q'(x) = Q(x) \tag{17}$$

This is obviously contradictory to the Lagrange interpolation principle, which states that there exists only one n-degree Lagrange interpolation function that passes through the given $n + 1$ data points. Therefore, the cloud server cannot deceive the user's verification to achieve the goal of forging the aggregation result. In addition, each user can independently verify the correctness of the aggregated gradient returned by the cloud server.

If the verification is not successful, the training is terminated; otherwise, the model training is continued.

$$w_i = \Delta_\theta L_i(\theta_i) \tag{18}$$

Substituting (18) with

$$g_t = \frac{1}{n}\sum_{i=1}^{n}\omega_i = \nabla L_i(w_i) \tag{19}$$

Substituting (19) with

$$W_{i+1} \leftarrow W_i - \eta g_t \leftarrow W_i - \eta \nabla L_i(w_i) \tag{20}$$

□

According to the above equations, we can conclude that if the server forges the aggregated gradient, our security protocol can identify it and terminate the model training for that round. However, if all users, the TA and the cloud server faithfully follow the agreed-upon process for training the model in the FL model, then users can obtain the correct aggregated gradient information returned by the cloud server to update the model.

6. Performance Evaluation

In this section, we will evaluate the effectiveness of our proposed CVFL scheme based on its performance, computational complexity and communication overhead under the premise of completing the DL task.

6.1. Dataset and Experimental Setup

We implemented the CVFL scheme using Python. Our experiments were conducted on a PC with an RTX 3090 GPU, 16 VCPU Intel(R) Platinum 8350C CPU, 2.60 GHz and 42 GB RAM. The MNIST [36] and CIFAR-10 datasets were used for experiments with MLP and CNN. The MNIST dataset consists of 70,000 labeled grayscale images of handwritten digits with a size of 28 × 28 pixels. It contains 60,000 training samples and 10,000 testing samples. The CIFAR-10 dataset is a standardized dataset for object recognition, consisting of 10 classes of colored images with a size of 32 × 32 pixels. It contains 50,000 training samples and 10,000 testing samples. In the experiments, both datasets were randomly divided among users using two methods: IID (independent and identically distributed) and non-IID. To evaluate the effectiveness of our approach, we utilized the following metrics for experimental evaluation:

1. Accuracy: We set a fixed number of epochs and users. We compare the accuracies and observe the impact of the number of users on the convergence of the proposed approach and some advanced approaches.
2. Convergence speed: With a fixed number of epochs and users, we compare the convergence speed of our proposed CVFL with some advanced approaches when the model tends to converge.
3. Communication frequency: We compare the number of training rounds required for our proposed CVFL and some advanced approaches to converge the model. Clearly, a lower communication frequency implies lower communication overhead to a certain extent.
4. Computation overhead: We measure the time required for the CVFL approach to converge the model and compare it with some advanced approaches. This comparison allows us to observe the scalability of our approach in terms of computational overhead.

6.2. Evaluation of Model Accuracy

Our CVFL scheme employs masking techniques to protect gradient privacy. After successfully verifying the aggregation result, masks are removed using Equation (9) to obtain accurate local gradient information. The FedAVG algorithm is then used to compute precise aggregated gradients, achieving $\varepsilon = 0$ differential privacy. Unlike traditional differential privacy methods, our approach does not compromise accuracy.

According to Figure 3, on the MNIST dataset, CVFL gradually converges after 10 epochs. Under the MLP architecture with an IID data distribution, the model achieves an accuracy

of 90.63%. Under a non-IID data distribution, the model achieves an accuracy of 87.47%. Under the CNN architecture with an IID data distribution, the model achieves an accuracy of 96.72%. Under a non-IID data distribution, the model achieves an accuracy of 93.53%. It can be observed that CVFL exhibits significantly higher model accuracy compared to G-VCFL and is roughly comparable to the original federated learning model (FedAVG) for deep learning classification tasks. Additionally, CVFL demonstrates faster convergence compared to G-VCFL.

As shown in Figure 4, on the CIFAR dataset, we found that the average loss per round during the training process of CVFL is similar to FedAVG, and its convergence speed and model accuracy are also close to the original federated learning model.

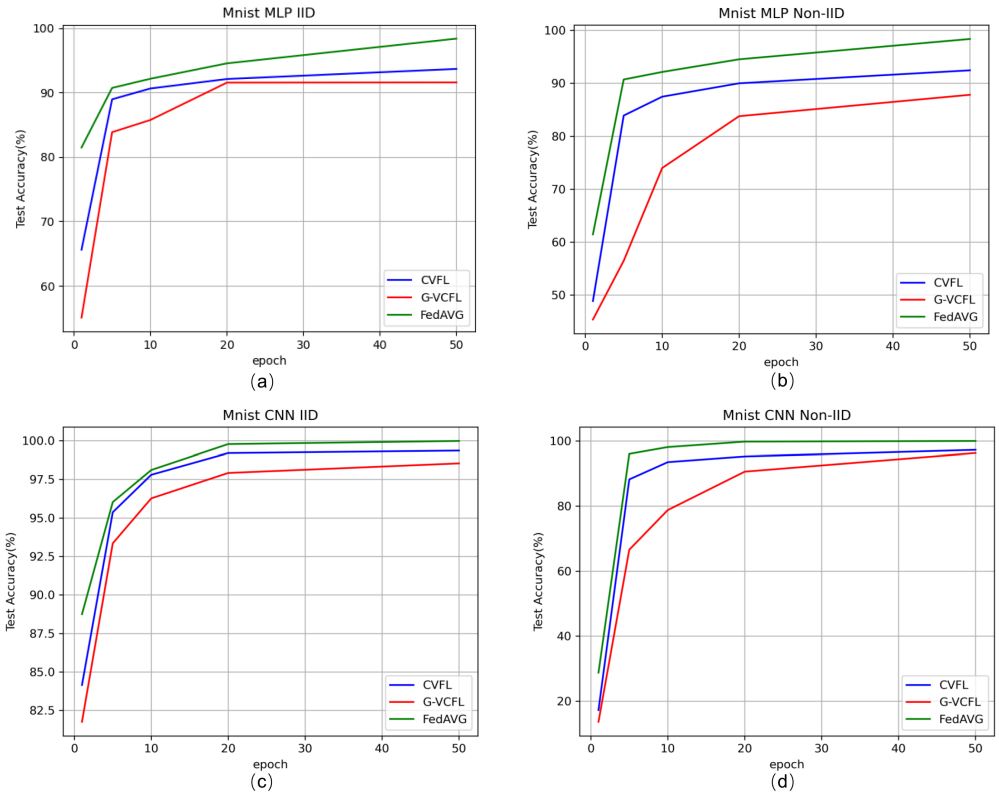


Figure 3. Model accuracy comparison on MNIST dataset. Figures (**a**–**d**) show the accuracy comparison of the CVFL model with G-VCFL and the traditional federated learning model FedAVG for deep learning based on MLP and CNN models with independently and non-independently distributed MNIST datasets, respectively, for participant user = 100.

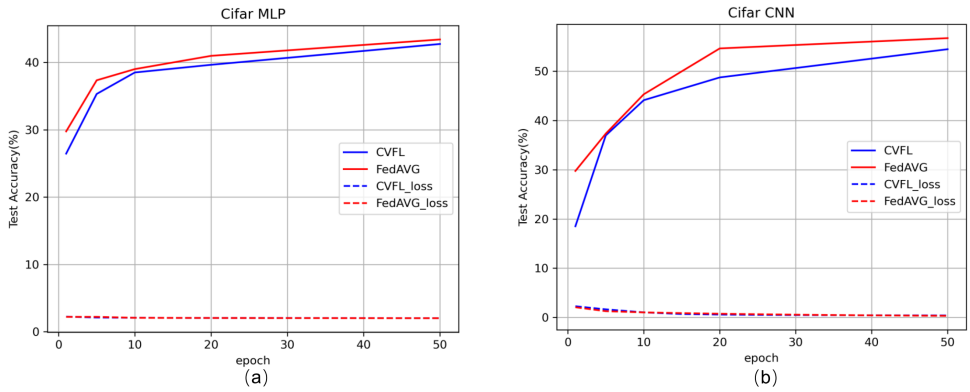


Figure 4. Model accuracy comparison on Cifar-10 dataset. Figures (**a**,**b**) show the comparison of accuracy, training loss between CVFL model and traditional federated learning model FedAVG for deep learning based on MLP and CNN models on Cifar-10 dataset for participating users = 100, respectively.

Therefore, the experimental results indicate that our CVFL model addresses the security vulnerabilities of the original federated learning model without sacrificing its original performance. It enhances the security of the original federated learning model and does not result in a loss of classification accuracy or convergence speed in deep learning tasks. Thus, it can be applied to deep learning tasks in real-world environments.

In Figure 5, we observed that, as the number of users participating in CVFL model training increased, the convergence speed and classification accuracy of the model also improved. This is advantageous for our model to handle large-scale data distributed across different user devices or nodes. Additionally, the involvement of a large number of users accelerated the convergence of the model and reduced data transmission costs between users and servers.

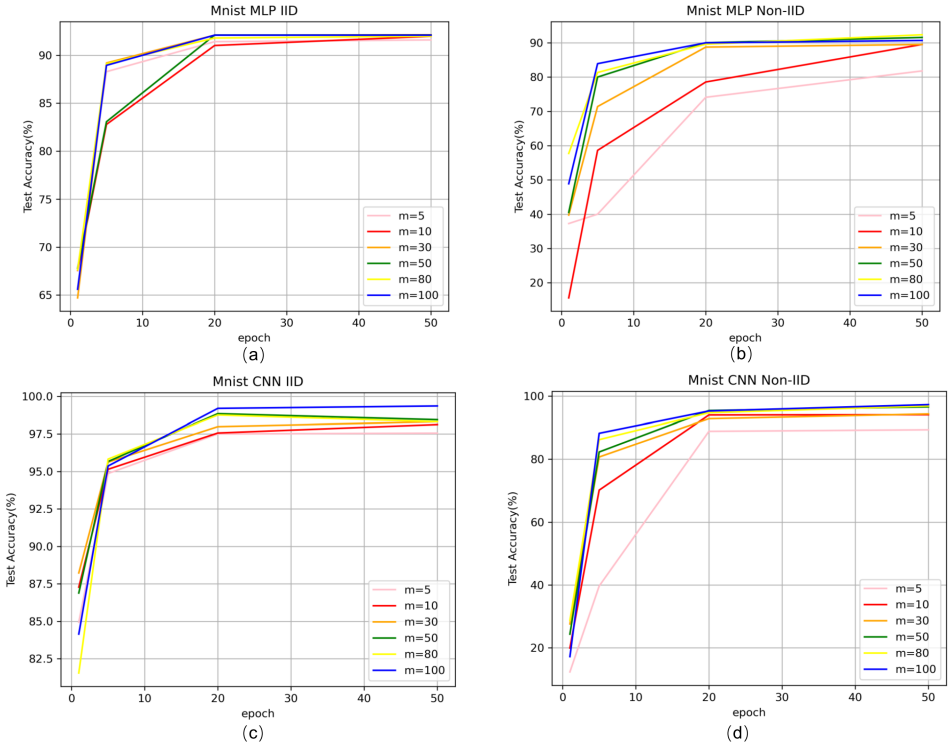


Figure 5. Relationship among accuracy and number of users. Figures (**a**–**d**) show the accuracy of the CVFL model for deep learning based on MLP and CNN models on IID and non-IID MNIST datasets with different numbers of users, respectively.

6.3. Evaluation on Execution Time

The total execution time of our CVFL model includes data sample partitioning, user model training and secure aggregation on the server. The computational overhead is mainly in secure aggregation. Our approach eliminates traditional encryption and decryption steps, reducing verification costs during aggregation. In comparison, HE-based schemes like VerifyNet and VFChain require decryption for verification, risking wasting time if verification fails. Thus, theoretically, CVFL has an advantage in execution time. The experiment compared the execution times of the CVFL, VFL, VFChain, VerifyNet and Blockchain models using CNN training (Figure 6). All models showed increased computational cost with more epochs. However, our CVFL model had a significantly lower computational cost compared to the other four models. The masking design in our approach corresponds to differential privacy with $\varepsilon = 0$, resulting in substantial improvements in computational overhead. And we found that the computational cost of the CVFL model increases roughly linearly with the number of epochs. For instance, the time cost for ten users participating in five epochs is approximately 48.63 s, while the time cost for

ten users participating in fifty epochs is approximately 463.12 s, which is about ten times longer than the case with ten users.

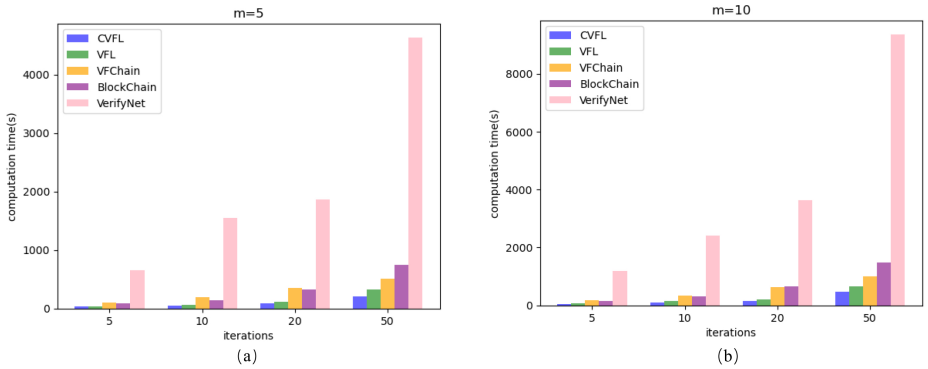


Figure 6. Execution time comparison on MNIST dataset. Figures (**a**,**b**) show the computational overhead comparison between the CVFL model and the four models, VFL, VFChain, BlockChain and VerifyNet, for deep learning on the MNIST dataset when the participating users = 5 and 10, respectively.

Additionally, we evaluated the execution time of training MLP and CNN models with different numbers of users, as shown in Figure 7. We found that the computational cost of the CVFL model increases roughly linearly with the number of users. For instance, the time cost for 10 users participating in 20 iterations is approximately 109.56 s, while the time cost for 80 users participating in 20 iterations is approximately 852.37 s, which is about eight times longer than the case with 10 users.

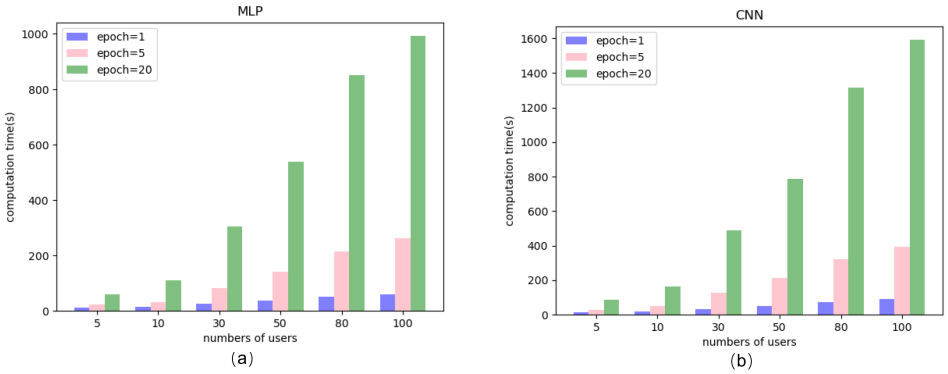


Figure 7. Relationship between execution time and number of users. Figures (**a**,**b**) show the comparison of the computational overhead when performing deep learning with epoch = 1, 5 and 20 for different numbers of users participating in the CVFL model based on the MNIST dataset, respectively.

6.4. Communication Overhead of CVFL

The total amount of data transmitted during the entire CVFL model training process increases with the number of participating users and also with the number of iterations. The users of our model are stored in the model in serial form, and the central server only communicates with the first and the last user, which eliminates the communication process between most of the users and the cloud server during the gradient information transfer process and greatly reduces the communication burden of the central server. In addition, we use the verification method based on the Lagrange interpolation principle in the verification part of the security aggregation process, and its method does not involve the transmission of ciphertexts, so the communication overhead in the verification part is only related to the number of users and the number of iterations.

7. Discussion

7.1. Model Accuracy

By conducting experiments on both the MNIST and CIFAR-10 datasets using MLP and CNN architectures, considering both independent and non-independent data distributions, we observed that the CVFL solution exhibited significantly better convergence in terms of model accuracy compared to G-VCFL [14]. Furthermore, it showed little difference compared to the original federated learning method, FedAVG [15]. The minor differences may be attributed to the additional computational and communication overhead introduced by the extra security measures implemented in the CVFL solution. These additional overheads could introduce some degree of error and have a slight impact on model training, resulting in a slight loss in model accuracy. However, this indicates that our solution does not negatively affect the performance and prediction accuracy of the model. This positive outcome can be attributed to our novel chained encryption local gradient approach, where we add masks in a serial manner to the users' local gradients. Unlike traditional differential privacy methods [21,22], our approach is similar to noiseless encryption [14] as it adds masks without introducing noise. Therefore, our solution maintains model accuracy while satisfying practical requirements.

7.2. Scalability

Federated deep learning [29] models often require diverse participants to provide large amounts of data, enabling federated deep learning to leverage abundant data resources. In federated deep learning, each user's local dataset may have different data distributions, creating a distributed dataset that contains more diverse and extensive information. By aggregating these heterogeneous data sources, federated deep learning can obtain a more comprehensive data perspective. These data can facilitate more comprehensive and accurate model training, making the model more robust and capable of generalization. This is beneficial for addressing overfitting and improving model performance, ultimately leading to higher-performing federated deep learning models. It is evident that the number of users can influence the training of federated learning in various ways, and it may even impact the model performance of our solution. Therefore, our solution needs to be applicable in a federated deep learning environment with a large number of users. As a result, we evaluate the scalability of CVFL, specifically assessing the impact of the number of users on model performance.

Specifically, we conducted experiments on the MNIST and CIFAR-10 datasets using MLP and CNN architectures, considering both independent and non-independent data distributions. We varied the number of users between 5, 10, 30, 50, 80 and 100. It was easy to observe that the number of users had a positive impact on accuracy, particularly for non-independent data distributions. This is because in non-independent data distributions, there are significant differences in the distribution characteristics among users. Each user trains their local model on different subsets of the dataset, which leads to collecting more diverse and extensive data information during the training process. Aggregating these heterogeneous data provides the federated deep learning model with a more comprehensive data perspective. The global model can be seen as an aggregation of various local models, enabling it to have a more complete and accurate model training process. Consequently, the global model becomes more robust and capable of generalization.

Furthermore, we observed that the number of users also had a positive impact on the convergence speed of the global model, especially for non-independent data distributions. This is because the heterogeneity in the distribution of non-independent datasets implies more diverse data features. The heterogeneous characteristics provide more extensive and comprehensive information for model training. The gradients from these diverse local models contribute to the global model's ability to quickly select data and models with higher relevance, accelerating its convergence speed. This improvement enhances the effectiveness and generalization capability of the global model. Therefore, as the number of users increases, the convergence speed of the aggregated model also increases.

7.3. Computation Overhead

To evaluate the computational overhead of our proposed solution, we tested the total execution time required for model convergence using five models: CVFL, VFChain [27], VerifyNet [24], Blockchain and VFL [29]. The total execution time includes the time cost for data sample partitioning among users, user model training and secure aggregation on the server. The main computational overhead lies in the secure aggregation of models. In this evaluation, we used a CNN neural network and the MNIST dataset, with the number of users set to five and ten, respectively.

It can be observed that our solution has a significantly shorter execution time compared to the other four models. This is because our approach involves fewer complex cryptographic primitives compared to the other models. Additionally, we adopted a verify-first-then-decrypt approach, meaning that if the verification fails, the aggregated gradients are discarded and the aggregation process restarts. On the other hand, VerifyNet, Blockchain and VFChain require decrypting encrypted information before verification. If the verification fails, it results in wasting the time cost of prior decryption. Furthermore, our solution employs a lightweight encryption method, similar to differential privacy [22], where only the initial masks need to be removed. This gives our solution an advantage in terms of execution time compared to VFL and other schemes that use homomorphic encryption [7].

Additionally, we evaluated the relationship between the total execution time of the CVFL model and the number of users. In this evaluation, we used CNN and MLP neural networks with the MNIST dataset, and varied the number of users between 5, 10, 30, 50, 80 and 100. It was easy to observe that, as the number of users increased, the execution time also increased correspondingly, showing nearly linear growth. We are aware that the execution time of other models also increases with the number of users. However, our solution already demonstrates a clear advantage in terms of execution time when dealing with a smaller number of users. This indicates that our solution also has a significant advantage in realistic federated deep learning environments with a larger number of users. By reducing the execution time, we can complete model training and aggregation more efficiently, thereby improving the overall system efficiency and performance.

In conclusion, our CVFL solution significantly reduces its computational overhead and is more suitable for federated deep learning environments with a large number of users.

8. Conclusions

In this paper, we proposed a verifiable protocol for privacy-preserving federated learning, specifically targeting resource-constrained environments with a large number of users. Our solution introduces a novel design for chain encryption of local gradients, incorporating masks in a sequential manner within the user's local gradients, which differs from previous work. This approach resembles encryption with zero noise and enables lightweight encryption and aggregation, reducing the computational cost of encryption and decryption operations significantly. During the gradient transmission process, the server only interacts with the first and last users, thereby alleviating the communication burden on the central server. Additionally, based on the Lagrange interpolation principle, we designed a verifiable protocol that assists the server in achieving correct secure aggregation to ensure the accuracy of deep learning model training. Furthermore, our work provides security analysis and proofs to ensure verifiable secure aggregation against malicious adversaries. We implemented CVFL, and conducted extensive experiments using the popular MLP and CNN architectures and real-world datasets, such as MNIST and CIFAR-10. We evaluated the effectiveness of the CVFL solution using four metrics: accuracy, convergence speed, time overhead and communication count. The experimental results demonstrate that our solution enhances the security and privacy of federated learning while maintaining performance without sacrificing it. Moreover, our solution exhibits advantages in terms of computational overhead, making it scalable in realistic environments. In this work, our main focus was on resisting attacks from malicious aggregation servers while

disregarding the scenario of malicious users. As part of our ongoing research, we will delve into the development of techniques to identify and mitigate potential malicious attacks originating from the user side within the proposed system architecture. This involves exploring robust mechanisms to detect and prevent unauthorized tampering or manipulation of the local model's gradient information by users. Additionally, we will also address concerns related to user revocability, ensuring that appropriate measures are in place to revoke access or privileges for users who may pose a threat to the integrity or security of the federated learning framework. These future research endeavors aim to enhance the overall security and reliability of the system while maintaining the privacy and confidentiality of user data.

Author Contributions: Conceptualization, M.W. and C.C.; methodology, M.W. and X.W.; software, M.W., Z.J. and H.L.; investigation, C.C., X.W. and J.S.; writing—original draft preparation, M.W.; writing—review and editing, C.C., X.W., Q.Z. and J.S.; supervision, C.C. and J.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Key Research and Development Program of National Natural Science Foundation of China (No. 2021YFB2700600) and Joint Funds of the National Natural Science Foundation of China (No. U19B2044).

Data Availability Statement: The datasets used in this paper are public datasets. Datasets are available in the relevant studies of the authors mentioned in Section 6.1. The datasets used and analyzed during the current study are available from the corresponding author on reasonable request.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Chen, C.; Xiao, J.; Liu, J.; Zhang, J.; Jia, J.; Hu, N. Unsupervised Intra-Domain Adaptation for Recommendation via Uncertainty Minimization. In Proceedings of the 2023 IEEE 39th International Conference on Data Engineering Workshops (ICDEW), Anaheim, CA, USA, 3–7 April 2023; pp. 79–86. [CrossRef]
2. Nguyen, D.C.; Ding, M.; Pathirana, P.N.; Seneviratne, A.; Li, J.; Vincent Poor, H. Federated Learning for Internet of Things: A Comprehensive Survey. *IEEE Commun. Surv. Tutor.* **2021**, *23*, 1622–1658. [CrossRef]
3. Chen, M.; Poor, H.V.; Saad, W.; Cui, S. Wireless Communications for Collaborative Federated Learning. *IEEE Commun. Mag.* **2020**, *58*, 48–54. [CrossRef]
4. Khan, L.U.; Saad, W.; Han, Z.; Hossain, E.; Hong, C.S. Federated Learning for Internet of Things: Recent Advances, Taxonomy, and Open Challenges. *IEEE Commun. Surv. Tutor.* **2021**, *23*, 1759–1799. [CrossRef]
5. Madi, A.; Stan, O.; Mayoue, A.; Grivet-Sébert, A.; Gouy-Pailler, C.; Sirdey, R. A Secure Federated Learning framework using Homomorphic Encryption and Verifiable Computing. In Proceedings of the 2021 Reconciling Data Analytics, Automation, Privacy, and Security: A Big Data Challenge (RDAAPS), Hamilton, ON, Canada, 18–19 May 2021; pp. 1–8. [CrossRef]
6. Liu, X.; Li, H.; Xu, G.; Chen, Z.; Huang, X.; Lu, R. Privacy-Enhanced Federated Learning Against Poisoning Adversaries. *IEEE Trans. Inf. Forensics Secur.* **2021**, *16*, 4574–4588. [CrossRef]
7. Luo, X.; Wu, Y.; Xiao, X.; Ooi, B.C. Feature Inference Attack on Model Predictions in Vertical Federated Learning. In Proceedings of the 2021 IEEE 37th International Conference on Data Engineering (ICDE), Chania, Greece, 19–22 April 2021; pp. 181–192. [CrossRef]
8. Alazab, M.; RM, S.P.; M, P.; Maddikunta, P.K.R.; Gadekallu, T.R.; Pham, Q.V. Federated Learning for Cybersecurity: Concepts, Challenges, and Future Directions. *IEEE Trans. Ind. Inform.* **2022**, *18*, 3501–3509. [CrossRef]
9. Wang, Z.; Song, M.; Zhang, Z.; Song, Y.; Wang, Q.; Qi, H. Beyond Inferring Class Representatives: User-Level Privacy Leakage From Federated Learning. In Proceedings of the IEEE INFOCOM 2019—IEEE Conference on Computer Communications, Paris, France, 29 April–2 May 2019; pp. 2512–2520. [CrossRef]
10. Guan, Z.; Zhang, Y.; Zhu, L. Hiding images within EFFECT: An efficient flexible privacy-preserving data aggregation scheme with authentication in smart grid. *IEEE Trans. Inf. Forensics Secur.* **2019**, *62*, 32103 . [CrossRef]
11. Zhang, J.; Zhao, Y.; Wu, J.; Chen, B. LVPDA: A Lightweight and Verifiable Privacy-Preserving Data Aggregation Scheme for Edge-Enabled IoT. *IEEE Internet Things J.* **2020**, *7*, 4016–4027. [CrossRef]
12. Liu, J.; Xu, H.; Wang, L.; Xu, Y.; Qian, C.; Huang, J.; Huang, H. Adaptive Asynchronous Federated Learning in Resource-Constrained Edge Computing. *IEEE Trans. Mob. Comput.* **2023**, *22*, 674–690. [CrossRef]
13. Niknam, S.; Dhillon, H.S.; Reed, J.H. Federated Learning for Wireless Communications: Motivation, Opportunities, and Challenges. *IEEE Commun. Mag.* **2020**, *58*, 46–51. [CrossRef]
14. Zhang, Z.; Wu, L.; He, D.; Wang, Q.; Wu, D.; Shi, X.; Ma, C. G-VCFL: Grouped Verifiable Chained Privacy-Preserving Federated Learning. *IEEE Trans. Netw. Serv. Manag.* **2022**, *19*, 4219–4231. [CrossRef]

15. McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; y Arcas, B.A. Communication-efficient learning of deep networks from decentralized data. In Proceedings of the 20th International Conference on Artificial Intelligence and Statistic, Fort Lauderdale, FL, USA, 20–22 April 2017; pp. 1273–1282.
16. Shi, J.; Wan, W.; Hu, S.; Lu, J.; Yu Zhang, L. Challenges and Approaches for Mitigating Byzantine Attacks in Federated Learning. In Proceedings of the 2022 IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), Wuhan, China, 9–11 December 2022; pp. 139–146. [CrossRef]
17. Zhang, X.; Ji, S.; Wang, H.; Wang, T. Private, Yet Practical, Multiparty Deep Learning. In Proceedings of the 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), Atlanta, GA, USA, 5–8 June 2017; pp. 1442–1452. [CrossRef]
18. Li, P.; Li, J.; Huang, Z.; Li, T.; Gao, C.Z.; Yiu, S.M.; Chen, K. Multi-key privacy-preserving deep learning in cloud computing. *Future Gener. Comput. Syst.* **2017**, *74*, 76–85. [CrossRef]
19. Tueno, A.; Kerschbaum, F.; Katzenbeisser, S.; Boev, Y.; Qureshi, M. Secure Computation of the kth-Ranked Element in a Star Network. *arXiv* **2019**, arXiv:1909.08347.
20. Wenhao, M.; Chunlei, F.; Yan, L.; Chunqiang, H. Secure Two-Party Computation Based on Fast Cut-and-Choose Bilateral Oblivious Transfer. *Secur. Commun. Netw.* **2021**, *2021*, 12938. [CrossRef]
21. Tancik, M.; Mildenhall, B.; Ng, R. Stegastamp: Invisible hyperlinks in physical photographs. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 2117–2126.
22. Wei, K.; Li, J.; Ding, M.; Ma, C.; Yang, H.H.; Farokhi, F.; Jin, S.; Quek, T.Q.S.; Vincent Poor, H. Federated Learning With Differential Privacy: Algorithms and Performance Analysis. *IEEE Trans. Inf. Forensics Secur.* **2020**, *15*, 3454–3469. [CrossRef]
23. Zhang, Y.; Yu, H. Towards verifiable federated learning. *arXiv* **2022**, arXiv:2202.08310.
24. Xu, G.; Li, H.; Liu, S.; Yang, K.; Lin, X. VerifyNet: Secure and Verifiable Federated Learning. *IEEE Trans. Inf. Forensics Secur.* **2020**, *15*, 911–926. [CrossRef]
25. Guo, X.; Liu, Z.; Li, J.; Gao, J.; Hou, B.; Dong, C.; Baker, T. VeriFL: Communication-Efficient and Fast Verifiable Aggregation for Federated Learning. *IEEE Trans. Inf. Forensics Secur.* **2021**, *16*, 1736–1751. [CrossRef]
26. Ren, Y.; Li, Y.; Feng, G.; Zhang, X. Privacy-Enhanced and Verification-Traceable Aggregation for Federated Learning. *IEEE Internet Things J.* **2022**, *9*, 24933–24948. [CrossRef]
27. Peng, Z.; Xu, J.; Chu, X.; Gao, S.; Yao, Y.; Gu, R.; Tang, Y. VFChain: Enabling Verifiable and Auditable Federated Learning via Blockchain Systems. *IEEE Trans. Netw. Sci. Eng.* **2022**, *9*, 173–186. [CrossRef]
28. Huang, C.; Yao, Y.; Zhang, X.; Teng, D.; Wang, Y.; Zhou, L. Robust Secure Aggregation with Lightweight Verification for Federated Learning. In Proceedings of the 2022 IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), Wuhan, China, 9–11 December 2022; pp. 582–589. [CrossRef]
29. Fu, A.; Zhang, X.; Xiong, N.; Gao, Y.; Wang, H.; Zhang, J. VFL: A Verifiable Federated Learning With Privacy-Preserving for Big Data in Industrial IoT. *IEEE Trans. Ind. Inform.* **2022**, *18*, 3316–3326. [CrossRef]
30. Thudi, A.; Deza, G.; Chandrasekaran, V.; Papernot, N. Unrolling SGD: Understanding Factors Influencing Machine Unlearning. In Proceedings of the 2022 IEEE 7th European Symposium on Security and Privacy, Genoa, Italy, 6–10 June 2022; pp. 303–319. [CrossRef]
31. Ye, Y.; Li, S.; Liu, F.; Tang, Y.; Hu, W. EdgeFed: Optimized Federated Learning Based on Edge Computing. *IEEE Access* **2020**, *8*, 209191–209198. [CrossRef]
32. Essanhaji, M.E.A. Lagrange Multivariate Polynomial Interpolation: A Random Algorithmic Approach. *Mathematics* **2022**, *2022*, 8227086. [CrossRef]
33. Shoukralla, E.S.; Ahmed, B.M. Numerical Solutions of Volterra Integral Equations of the Second Kind using Lagrange interpolation via the Vandermonde matrix. *J. Phys. Conf. Ser.* **2020**, *1447*, 012003. [CrossRef]
34. Yu, Q.; Li, S.; Raviv, N.; Kalan, S.M.M.; Soltanolkotabi, M.; Avestimehr, S.A. Lagrange Coded Computing: Optimal Design for Resiliency, Security, and Privacy. In Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics, Naha, Japan, 16–18 April 2019; Volume 89, pp. 1215–1225.
35. Roy, S.S.; Mallik, A.; Gulati, R.; Obaidat, M.S.; Krishna, P.V. A deep learning based artificial neural network approach for intrusion detection. In Proceedings of the Mathematics and Computing: Third International Conference, ICMC 2017, Haldia, India, 17–21 January 2017; Proceedings 3; Springer: Berlin/Heidelberg, Germany, 2017; pp. 44–53.
36. Lecun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.