applied sciences

MDPI

Article

More Efficient and Verifiable Privacy-Preserving Aggregation Scheme for Internet of Things-Based Federated Learning

Rongquan Shi [1], Lifei Wei [2,*] and Lei Zhang [1,*]

[1] College of Information Technology, Shanghai Ocean University, Shanghai 201306, China; rongquans@163.com
[2] College of Information Engineering, Shanghai Maritime University, Shanghai 201306, China
* Correspondence: lfwei@shmtu.edu.cn (L.W.); lzhang@shou.edu.cn (L.Z.)

Abstract: As Internet of Things (IoT) technology continues to advance at a rapid pace, smart devices have permeated daily life. Service providers are actively collecting copious numbers of user data, with the aim of refining machine learning models to elevate service quality and accuracy. However, this practice has sparked apprehensions amongst users concerning the privacy and safety of their personal data. Federated learning emerges as an evolution of centralized machine learning, enabling a collective training of machine learning models by multiple users on their respective devices. Crucially, this is achieved without the direct submission of data to a central server, thereby significantly mitigating the hazards associated with privacy infringements. Since the machine learning algorithms act locally in federated learning, passing just the local model back to the central server, the users' data remain locally. However, current research work indicates that local models also include user data privacy-related components. Moreover, current privacy-preserving secure aggregation schemes either offer insufficient accuracy or need significantly high computing resources for training. In this work, we propose an efficient and secure aggregation scheme for privacy-preserving federated learning with lower computational costs, which is suitable for those weak IoT devices since the proposed scheme is robust and fault-tolerant, allowing some of the users to dynamically exit or join the system without restarting the federated learning process or triggering abnormal termination. In addition, this scheme with the property of result verification in the situation when the servers return incorrect aggregation results, which can be verified by the users. Extensive experimental evaluations, based on real-world datasets, have substantiated the high accuracy of our proposed scheme. Moreover, in comparison to existing schemes, ours significantly reduces computational and communication costs by at least 85% and 47%, respectively.

Keywords: federated learning; data aggregation; privacy preserving; verifiable computation

Citation: Shi, R.; Wei, L.; Zhang, L. More Efficient and Verifiable Privacy-Preserving Aggregation Scheme for Internet of Things-Based Federated Learning. *Appl. Sci.* **2024**, *14*, 5361. https://doi.org/10.3390/app14135361

Academic Editor: Luis Javier García Villalba

Received: 21 May 2024
Revised: 7 June 2024
Accepted: 17 June 2024
Published: 21 June 2024

1. Introduction

With the rapid development of IoT technology, IoT devices have provided ubiquitous sensing and computing powers and are widely used in many scenarios such as smart cities, healthcare, smart homes, and smart industries. According to a report by IoT Analytics [1], by 2027, there will be 29 billion IoT devices working concurrently around the world, compared to just 14.4 billion in 2022. The vast number of data generated by a multitude of devices provides rich material for analysis by artificial intelligence technologies such as machine learning and deep learning. These technologies have demonstrated powerful analytical and mining capabilities in fields like image classification, speech recognition, and natural language processing, effectively leveraging the data collected by the Internet of Things to uncover hidden patterns and information. Traditional machine learning methods rely on centralized AI frameworks located in data centers or clouds for data analysis and model training, which require centralized collection of data dispersed in various locations and model training in data centers. However, centralized machine learning methods also raise concerns about data privacy, as the data may contain sensitive information such as

users' personal preferences, health status, and more. Owing to concerns over privacy protection, users are often reluctant to share the data they possess, leading to data silos that prevent effective centralization for training and analysis. Therefore, a new machine learning training method is needed to alleviate users' concerns about privacy issues.

In 2016, Google researchers introduced a technique known as federated learning [2], which is a form of distributed machine learning aimed at addressing the issue of machine learning across a range of mobile devices. Federated learning, as opposed to the more traditional centralized machine learning, does away with the requirement to centralize data within a data center. Under the coordination of a server, various IoT devices act as clients to collaboratively train a model. Initially, the server establishes the current global model. Subsequently, each client downloads this model, utilizes its own data to train a local model, and then uploads the outcomes back to the server. The server then amalgamates these local models to form a new global model, which is subsequently downloaded by the clients to train their new local models. This cycle is repeated until a predefined training goal is met. By maintaining data privacy at the local level, this approach to machine learning ensures data privacy and security, potentially alleviating users' privacy concerns. It can enhance user privacy as the server is only aware of the model, not the learning data themselves. As a result, federated learning has been the subject of extensive research and application in areas such as health care [3], financial services [4], and intelligent manufacturing [5], emerging as a highly promising research field within artificial intelligence.

However, existing research indicates that traditional machine learning can be outperformed by federated learning in better protecting users' private data. In the domain of federated learning, there are still multiple attack methods that can threaten users' private data. Through their research, Mothukuri et al. [6] discovered that the local models submitted by users contain some information, and it is possible to infer sensitive information, such as users' training data, by analyzing the output of these local models. A framework was proposed by Wang et al. [7] that integrates a GAN with a multi-task discriminator on a malicious FL server to reconstruct a user's private training data. On the other hand, Zhu et al. [8] suggested a method where by constructing fake data and labels for training, the resulting fake model is maximized to differ as much as possible from the user-submitted local model, and through iterative updates, the user's training data can ultimately be recovered. To address the aforementioned issues, the methods commonly used to enhance privacy preservation in federated learning are primarily divided into homomorphic encryption [9–11], differential privacy [12,13], and secure multi-party computation [14–16].

In addition, it is also important to ensure that the server aggregation results are correct. A malicious federated learning server may return incorrect aggregation results. For example, a malicious federated learning server may intentionally return the aggregation results of the previous round to the user during a certain round of federated learning for the purpose of saving its own computational cost. Therefore, it is of interest to design a federated learning protocol that can both protect user privacy and verify the correctness of the server's aggregation results. Recently, some researchers have proposed the verifiable privacy-preserving aggregation schemes [17–21]. These schemes either require additional trusted hardware support or have high computational power requirements on the user side, rendering them inappropriate for IoT devices characterized by limited processing capability. The properties that the related work satisfies are shown in Table 1.

In this paper, we propose a privacy-preserving federated learning data aggregation scheme that is both efficient and verifiable, specifically designed for IoT devices with lower computational and communication costs to accommodate their weaker performance capabilities. The specific contributions of this paper are as follows.

- We propose a novel, efficiency-focused federated learning scheme that facilitates privacy-preserving data aggregation, while notably reducing computational and communication costs. One of the distinguishing features of our solution is that it allows users to perform low-cost validation of the aggregation results provided by the server to ensure their integrity and reliability. This neither compromises computational

efficiency nor strict privacy measures, making it a secure and effective data aggregation solution.

- We also devised a dynamic user management mechanism. This mechanism is constructed to enable the seamless onboarding and departure of participants within the federated learning process. Through its implementation, it ensures that the federated learning process remains robust against user attrition. When some users dropout, the remaining active participants experience minimal disruption in terms of both computational load and communication bandwidth. Despite these departures, the integrity of the aggregated encrypted gradients is preserved, maintaining the overall efficacy of the federated learning process.
- We conduct comprehensive tests on numerous real-world datasets and thoroughly analyze accuracy and performance. The outcomes of these tests have been strikingly positive. Our scheme demonstrates a significant reduction in computational and communication costs, achieving at least 85% and 47%, respectively, when compared to current schemes in existence. Furthermore, our scheme has proven its mettle in terms of accuracy as well. It has managed to attain a level of precision that is highly comparable to those achieved using plaintext baselines, thereby affirming its reliability and robustness. These findings not only highlight the superiority of our scheme but also underscore its potential for practical application in real-world scenarios, particularly where computational resources are limited.

Table 1. Advantages and disadvantages comparison among the related schemes. "**L**" denotes the lightweight cryptography. "**F**" denotes fault tolerance against users dropping out. "**I**" denotes individual keys **without** trusted third parties. "**H**" denotes high accuracy of global model. "**V**" denotes verifiable computation to prevent malicious server returning incorrect results. "**T**" denotes the trusted execution environments (TEEs) are **not** required.

Schemes	L	F	I	H	V	T
Bonawitz et al. [16]	✓	✓	✓	✓	✗	✓
Fang et al. [22]	✗	✓	✗	✓	✗	✓
Zhu et al. [23]	✗	✓	✗	✓	✓	✓
Ma et al. [24]	✗	✓	✗	✓	✗	✓
Xu et al. [18]	✗	✓	✗	✓	✓	✓
Zhao et al. [25]	✓	✓	✓	✓	✗	✗
Nguyen et al. [26]	✗	✓	✗	✓	✗	✗
Zhang et al. [27]	✓	✓	✓	✓	✗	✓
Our work	✓	✓	✓	✓	✓	✓

2. Related Work

Federated learning, a decentralized machine learning technique, was pioneered by Google in 2016 as a solution to train predictive models on mobile devices without consolidating datas [2]. Its fundamental principle revolves around enabling users to contribute to a collective global model by transmitting their locally trained models to a central server, while keeping their datasets secure on their devices. In the federated learning framework, the server initiates each training cycle by distributing the current global model parameters to participating clients. These clients then utilize their own localized datasets to refine a model tailored to their unique data. Upon completion of local training, the clients relay their updated model parameters back to the server. The server employs an aggregation mechanism to amalgamate these parameters from various clients, resulting in a refined set of global model parameters. Subsequently, these enhanced global parameters are made available for download by the clients, marking the commencement of the subsequent training iteration. This cyclic process persists until the global model reaches a satisfactory level of performance or meets predefined criteria.

In recent years, federated learning has been gaining traction in the business world as a solution to privacy and security concerns; academics are pushing its boundaries through research, and innovative algorithms like FedGRU [28] and FedSem [29] are being developed to predict traffic flow for smart vehicles, train medical models without sharing patient data, and utilize unlabeled data in smart cities, respectively. All these developments highlight federated learning's significant promise in cross-institutional data collaboration while maintaining privacy.

However, some research indicates that within the context of federated learning, some attack methods remain capable of threatening users' privacy. Mothukuri et al. [6] have identified several types of attacks, including membership inference assaults, unintentional data leaks, inference-based reconstructions, and GAN-based inference attacks. A membership inference attack, specifically, is when an attacker examines the global model to uncover information about the training data of other users [30]. This type of attack allows the attacker to determine whether a certain type of dataset was used for model training [31–34]. The GAN-based inference attack takes use of GAN's superior data reconstruction performance to learn the distribution of the user's privacy dataset based on the user's local model and then rebuild the user's privacy dataset. It has been proved experimentally that GAN may be used to rebuild user privacy data [35].

To address the aforementioned issues, federated learning commonly employs several privacy-enhancing techniques, including homomorphic encryption [22,24,36] and differential privacy [12,37–39]. These methods aim to achieve model training and information sharing among participants while preserving the privacy of user data.

In the field of verifiable privacy-preserving federated learning, researchers have innovated diverse methods to ensure data confidentiality and collaborative model training. Hahn et al. [17] uses a dual-mask protocol for gradient privacy, requiring servers to prove aggregated results' accuracy. Xu et al. [18] substitutes homomorphic hashing for dual masks in challenge creation. Ghodsi et al. [19] designed a specialized interactive proof protocol using arithmetic circuits to detect malicious servers. This method ensures that the server cannot tamper with users' data or models during the federated learning process. Tramer et al. [20] utilized trusted execution environments such as SGX (Software Guard Extensions) and TrustZone to achieve privacy preserving and execution verification. These trusted execution environments ensure that the computation processes running on servers are secure and users' privacy data remain confidential. Fu et al. [21] applied Lagrange interpolation and blinding techniques for enhanced privacy and gradient verification. Each approach uniquely contributes to the evolving landscape of privacy in federated learning, showcasing varied strategies to tackle privacy concerns in collaborative machine learning settings.

All of these studies and frameworks promote the development of federated learning techniques while protecting privacy, providing powerful methods and guarantees to protect user data privacy. The introduction of homomorphic encryption and differential privacy allows for better protection of individual users' private information during federated learning, realizes cooperative computation and model training while maintaining data privacy, and enables cooperative computation and model training across devices to be performed more reliably.

3. Preliminaries

Federated Learning

Federated learning is a distributed machine learning approach that allows for model training and optimization without moving datasets from everywhere to a centralized server. Users send only local models trained from local data to a centralized server for aggregation, thus jointly training a global model. The aggregation formula is $w_g^{t+1} = \sum_{n=1}^{N} \frac{k_n}{K} w_n^{t+1}$, where K denotes the total number of samples and k_n denotes the number of samples for the n-th client.

During each iteration, each client proceeds as follows: first, the current global model w_g^t is downloaded from the server. Next, the model is trained using its own local dataset using a stochastic gradient descent (SGD) optimizer. During the SGD process, the model parameters of the n-th client in the $(t+1)$-th round of training are updated as follows: $w_n^{t+1} \leftarrow w_g^t - \eta \cdot \delta L(w_g^t, \beta)$, where β denotes the batch randomly sampled from the local dataset, η denotes the learning rate and L denotes the loss function.

After training, each client submits the local model w_n^{t+1} to the server. Finally, the server aggregates the local models w_n^{t+1} submitted by each client to obtain a new global model w_g^{t+2}, which will be used in the next round of iteration.

4. System and Threat Model

Figure 1 depicts the system model of our proposed scheme. we also discuss the threat model, security model and design goals.

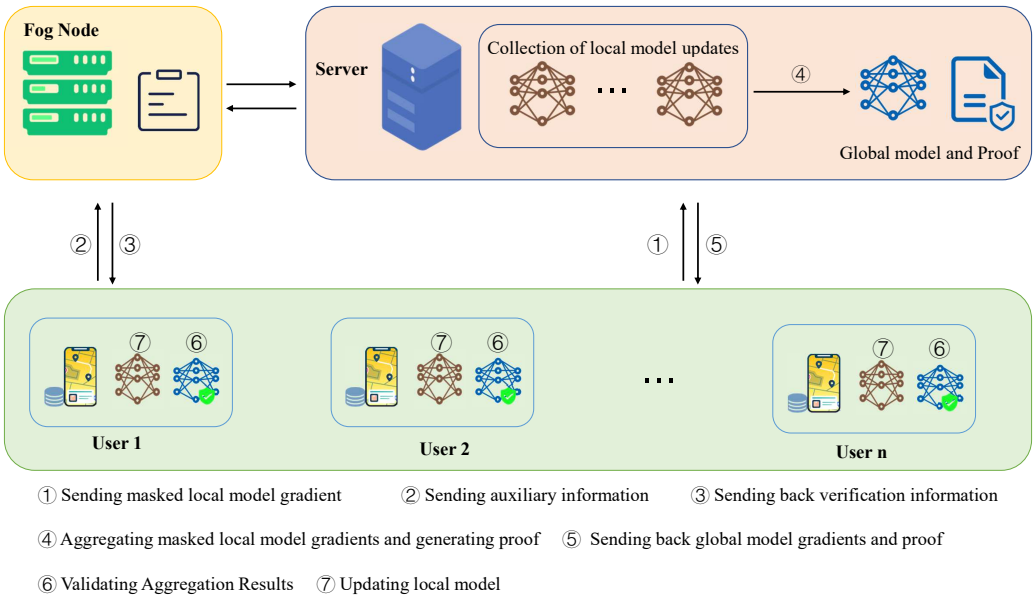


Figure 1. System model.

4.1. System Model

In this paper, we focus on how to aggregate a global models in federated learning while protecting the user's private data, as well as enabling the user to validate the server's aggregation results. Moreover, the global model can still be correctly aggregated when some users drop out. The system model in this paper involves three main parties:

- **Users**. Data ownership and local execution of the training algorithm fall under the purview of the users. Upon finishing a training round, users encrypt the local gradient parameters and transmit them to the fog node. Moreover, users receive the encrypted global gradient parameters that have been aggregated by the aggregation server. They then update their local models and proceed to conduct a next round of model training with their respective datasets. This iterative process continues until the model's training is deemed successful. Users possess the capability to authenticate the correctness of the aggregation outcomes produced by the server.
- **Server**. The fog node dispatches ciphered gradients to the server, which then conducts aggregation calculations without breaching user confidentiality. The server's access is limited to the aggregate of locally encrypted model gradients, precluding any direct exposure to users' unencrypted gradients. As an added measure, the server creates a proof of correctness for its aggregated results, subsequently transmitting this proof alongside the aggregated results to the respective users.

- **Fog Node**. The Fog Node (FN), equipped with robust computing, storage, and networking functionalities, is strategically positioned in proximity to the user. Its primary duty involves gathering data from all online users involved in the current round. Following this collection phase, the aggregated information is seamlessly transmitted to the central aggregation server for further processing.

4.2. Threat Model

In this paper, we posit that every participant, including all users, the fog node, and the server, operates under a semi-honest assumption. They adhere to the protocol's stipulations accurately, yet they harbor curiosity regarding the privacy of their peers and will endeavor to deduce others' confidential data from the local models that are submitted by other users. Beyond this semi-honest threat scenario, we also contemplate a more severe one where, at any juncture, the server might fall into the clutches of malevolent actors. In such a case, the server could stray from the protocol's original intent, deliver erroneous computations, and endeavor to deceive users into accepting these flawed global parameters. It is noteworthy that our proposed scheme permits collusion between any two entities within the aggregation server, a user, and a fog node to uncover a specific user's private information. However, it should be acknowledged that when the server and fog node are in collusion, it becomes quite feasible to fabricate aggregation outcomes and verifications, and our scheme does not allow the simultaneous collusion of all three parties.

4.3. Security Model

We assume that the risk of data interception by malicious parties during user-server communications. To counteract this, we developed a strategy centered around the computational Diffie–Hellman (CDH) problem. Our belief is that the CDH conundrum is computationally arduous, with the chance of any probabilistic polynomial-time attacker (PPTA) deciphering it being practically negligible [40].

4.4. Design Goal

For the federated learning scenario, our goal is to propose a robust, privacy-preserving framework for federated learning networks that enables data aggregation. There are several goals, as follows:

- **Privacy-preserving:** In our proposed scheme, the utmost importance is given to safeguarding users' privacy in federated learning. At no point can an attacker access any genuine information regarding the user's data.
- **Verifiability:** Users have the ability to check if the results that the server sends back, after combining all the data, are correct. They can do this by performing a simple calculation. This process is designed to be easy and not too complicated, ensuring that users can confirm the correctness of the server's aggregated results without much cost.
- **Fault tolerance:** Even if some users drop out or fail to upload their local model gradients in time due to various reasons, the federated learning system is still capable of correctly aggregating the gradients from the remaining normally active users to form a global model gradient.

5. Methodology

In this section, we describe a verifiable privacy-preserving data aggregation scheme for IoT-based federated learning in detail. We give the system process in Figure 2.

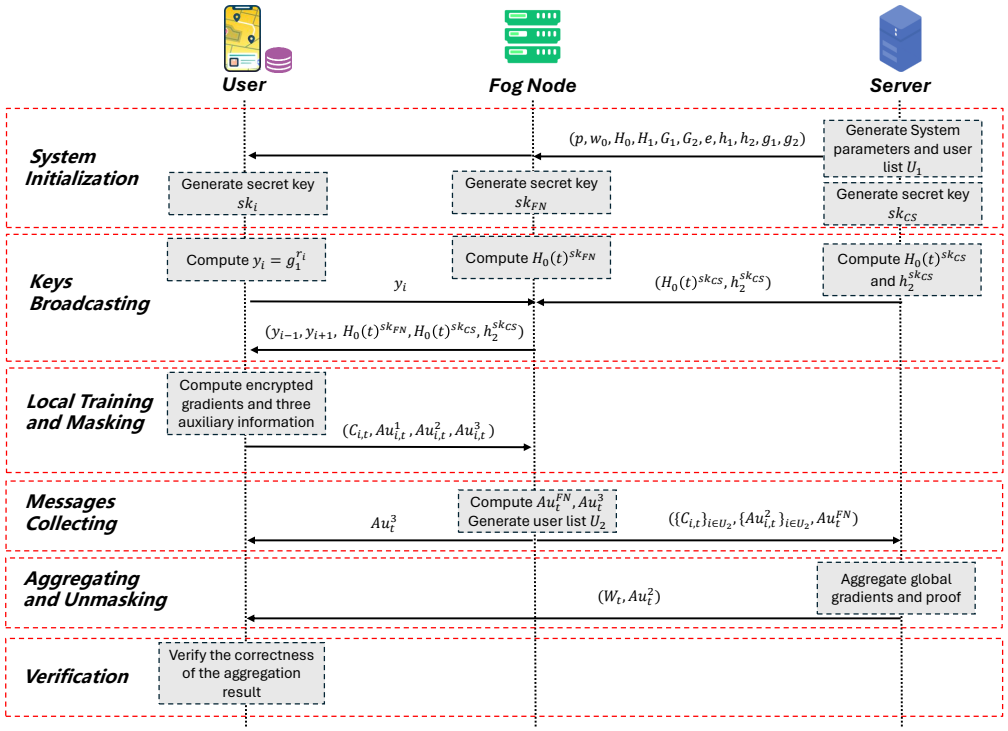


Figure 2. System process.

5.1. System Initialization

In our system, we envision a setup with a total of n users, represented by the set U, where each user i is assigned a unique identifier ranging from 1 to n. The server begins by setting up the initial model parameter w_0 and choosing a large prime number p. We define $\mathbb{G}_1$ and $\mathbb{G}_2$ as two distinct multiplicative cyclic groups that operate under multiplication and have the same prime order p, with g_1 and g_2 serving as their respective generators.

The server also picks a secret number $\kappa \in \mathbb{Z}_p^*$, and calculates h_1 and h_2 as g_1^κ and g_2^κ. Additionally, the server defines a bilinear mapping function $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ and selects two hash functions: $H_0 : \{0,1\} \to \mathbb{Z}_{p^2}^*$ and $H_1 : \{0,1\} \to \mathbb{G}_1$.

After these preparations, the server makes public the system parameters, which include $p, w_0, H_0, H_1, \mathbb{G}_1, \mathbb{G}_2, e, h_1, h_2, g_1$, and g_2. Furthermore, each user $\mathcal{U}_i$ independently selects a private key $sk_i \in \mathbb{Z}_p^*$, the fog node (FN) chooses its own private key $sk_{FN} \in \mathbb{Z}_p^*$, and the server selects a private key $sk_{CS} \in \mathbb{Z}_p^*$.

5.2. Keys Broadcasting

In this phase, user $\mathcal{U}_i$ randomly selects a secret number $r_i \in \mathbb{Z}_p^*$ and computes $y_i = g_1^{r_i}$. The FN computes $H_0(t)^{sk_{FN}}$ for epoch t. The server computes $H_0(t)^{sk_{CS}}$ and $h_2^{sk_{CS}}$.

After that, user $\mathcal{U}_i$ sends y_i to the FN, the server sends $H_0(t)^{sk_{CS}}$ and $h_2^{sk_{CS}}$ to the FN. After the FN receives information from all users, it denotes this group of users by U_1. Then, the FN sends $(\{y_i\}_{i\in U_1}, H_0(t)^{sk_{FN}}, H_0(t)^{sk_{CS}}, h_2^{sk_{CS}})$ to users in U_1.

5.3. Local Training and Masking

In this phase, user $\mathcal{U}_i$ trains a local model $w_{i,t}$ with their own private dataset for epoch t. And then, the user $\mathcal{U}_i$ computes the blinding factor Y_i

$$Y_i = \begin{cases} (y_{i+1}/y_n)^{r_i} \pmod{p^2}, & \text{if } i = 1, \\ (y_{i+1}/y_{i-1})^{r_i} \pmod{p^2}, & \text{if } i = 2, 3, \ldots, n-1, \\ (y_1/y_{i-1})^{r_i} \pmod{p^2}, & \text{if } i = n. \end{cases} \tag{1}$$

The user i generates the gradient ciphertext

$$C_{i,t} = (1 + w_{i,t} \cdot p) \cdot Y_i \cdot H_0(t)^{sk_{FN} sk_i} \pmod{p^2} \quad (2)$$

The user i also computes three pieces of auxiliary information:

$$Au_{i,t}^1 = H_0(t)^{sk_{CS} sk_i} \quad (3)$$

$$Au_{i,t}^2 = H_1(t)^{sk_i} \cdot h_1^{w_{i,t}} \quad (4)$$

$$Au_{i,t}^3 = h_2^{sk_{CS} sk_i} \quad (5)$$

After that, the user i sends $(C_{i,t}, Au_{i,t}^1, Au_{i,t}^2, Au_{i,t}^2)$ to the FN.

5.4. Messages Collecting

When the FN receives $(C_{i,t}, Au_{i,t}^1, Au_{i,t}^2, Au_{i,t}^2)$ from the user $\mathcal{U}_i$, it adds user $\mathcal{U}_i$ into U_2. The FN computes the unmasking key

$$Au_t^{FN} = (\prod_{i=1}^{n} Au_{i,t}^1)^{sk_{FN}} = H_0(t)^{sk_{FN} sk_{CS} \sum_{i=1}^{n} sk_i} \quad (6)$$

and

$$Au_t^3 = \prod_{i=1}^{n} Au_{i,t}^3 = \prod_{i=1}^{n} h_2^{sk_{CS} sk_i} = h_2^{sk_{CS} \sum_{i=1}^{n} sk_i} \quad (7)$$

Afterwards, the FN sends $(\{C_{i,t}\}_{i \in U_2}, \{Au_{i,t}^2\}_{i \in U_2}, Au_t^{FN})$ to the server, and sends Au_t^3 to users in U_2.

5.5. Aggregating and Unmasking

After the server receives the message from the FN, it executes the aggregation computation protocol and generates the corresponding proof of correctness. The server first computes C_t

$$C_t = \left(\prod_{i=1}^{n} C_{i,t} \right)^{sk_{CS}} \pmod{p^2} \quad (8)$$

and obtains the global gradient W_t

$$W_t = \frac{\frac{C_t}{Au_t^{FN}} - 1}{p \cdot sk_{CS}} \pmod{p^2} \quad (9)$$

We give a full description in Section 6.1.

Furthermore, the server computes the proof

$$Au_t^2 = \left(\prod_{i=1}^{n} Au_{i,t}^2 \right)^{sk_{CS}} = \left(\prod_{i=1}^{n} H_1(t)^{sk_i} \cdot h_1^{w_{i,t}} \right)^{sk_{CS}} = H_1(t)^{sk_{CS} \sum_{i=1}^{n} sk_i} \cdot h_1^{sk_{CS} \sum_{i=1}^{n} w_{i,t}} \quad (10)$$

Finally, the server sends the aggregation result and corresponding proof of correctness (W_t, Au_t^2) to users in U_2.

5.6. Verification

After receiving the result and proof from the server, each user verifies the correctness of the aggregation result with Au_t^1 by

$$e(Au_t^2, h_2) \stackrel{?}{=} e(H_1(t), Au_t^3) \cdot e(h_1^{W_t}, h_2^{sk_{CS}}) \quad (11)$$

If the equation mentioned above is valid, then every user will agree to the aggregated result and proceed to update their local model. Should the equation not be valid, they will disregard the aggregated result and advance to the next round.

6. Security Analysis

6.1. Correctness

Theorem 1. *If the server faithfully executes the aggregation protocol, it is easy to aggregate the global gradients.*

Proof of Theorem 1. Given the encrypted gradients $\{C_i\}_{i\in U_2}$, the server can aggregate the global gradients W_t with the auxiliary information Au_t^{FN} sent by the FN. The server first computes

$$\begin{aligned} C_t &= \left(\prod_{i=1}^{n} C_{i,t}\right)^{sk_{CS}} \pmod{p^2} \\ &= \left(\prod_{i=1}^{n}(1+w_{i,t}\cdot p)\cdot Y_i \cdot H_0(t)^{sk_{FN}sk_i}\right)^{sk_{CS}} \pmod{p^2} \\ &= \left(1+p\cdot\sum_{i=1}^{n} w_{i,t}\right)^{sk_{CS}} \cdot \left(g_1^0\right)^{sk_{CS}} \cdot \left(H_0(t)^{sk_{FN}\sum_{i=1}^{n} sk_i}\right)^{sk_{CS}} \pmod{p^2} \\ &= \left(1+sk_{CS}\cdot p\cdot\sum_{i=1}^{n} w_{i,t}\right)\cdot H_0(t)^{sk_{FN}sk_{CS}\sum_{i=1}^{n} sk_i} \pmod{p^2} \end{aligned}$$

and obtains the global gradient W_t of current epoch t by

$$\begin{aligned} W_t &= \frac{\frac{C_t}{Au_t^{FN}}-1}{p\cdot sk_{CS}} \pmod{p^2} \\ &= \frac{\frac{\left(1+sk_{CS}\cdot p\cdot\sum_{i=1}^{n} w_{i,t}\right)\cdot H_0(t)^{sk_{FN}sk_{CS}\sum_{i=1}^{n} sk_i}}{H_0(t)^{sk_{FN}sk_{CS}\sum_{i=1}^{n} sk_i}}-1}{p\cdot sk_{CS}} \pmod{p^2} \\ &= \frac{\left(1+sk_{CS}\cdot p\cdot\sum_{i=1}^{n} w_{i,t}\right)-1}{p\cdot sk_{CS}} \pmod{p^2} \\ &= \sum_{i=1}^{n} w_{i,t} \pmod{p^2} \end{aligned}$$

That is, if the server faithfully executes the aggregation protocol, it is easy to aggregate the global gradients. □

Theorem 2. *If the server faithfully executes the aggregation protocol, the aggregated global gradients are able to pass the verification of users.*

Proof of Theorem 2. Given the encrypted gradients $\{C_i\}_{i\in[1,n]}$, the server can aggregate the global gradients W_t with the auxiliary information Au_t^{FN} sent by the FN. Then, it can compute the proof Au_t^2. After that, it sends the global gradients W_t and the proof Au_t^2 to users. Also, the FN will send auxiliary information Au_t^3 to users.

Based on the global gradients W_t, the proof Au_t^2 and Au_t^3, Each user can use bilinear pairing to verify the correctness of W_t with Equation (11).

The validity of Equation (11) can be derived as follows:

$$\begin{aligned} e(Au_t^2, h_2) &= e(H_1(t)^{sk_{CS}\sum_{i=1}^{n} sk_i} \cdot g_1^{\kappa sk_{CS}\sum_{i=1}^{n} w_{i,t}}, g_2^{\kappa}) \\ &= e(H_1(t)^{sk_{CS}\sum_{i=1}^{n} sk_i}, g_2^{\kappa}) \cdot e(g_1^{\kappa sk_{CS}\sum_{i=1}^{n} w_{i,t}}, g_2^{\kappa}) \\ &= e(H_1(t), g_2^{\kappa sk_{CS}\sum_{i=1}^{n} sk_i}) \cdot e(g_1^{\kappa\sum_{i=1}^{n} w_{i,t}}, g_2^{\kappa sk_{CS}}) \\ &= e(H_1(t), Au_t^3) \cdot e(h_1^{W_t}, h_2^{sk_{CS}}) \end{aligned}$$

If the Equation (11) holds, the aggregated global gradients are able to pass the verification of users. □

6.2. Privacy Protection

Theorem 3. *The server, FN, and curious user can know nothing about the private information of user $\mathcal{U}_i$.*

Proof of Theorem 3. We assume that any probabilistic polynomial time adversary (PPTA), such as the honest but curious server, the FN, and the user, can eavesdrops on any communication between user $\mathcal{U}_i$, the FN, and the server. Moreover, the PPTA successfully obtains all messages transmitted between user $\mathcal{U}_i$ and the FN, including but not limited to $(y_i, y_{i+1}, y_{i-1}, C_{i,t}, H_0(t)^{sk_{FN}})$. To obtain the privacy information $w_{i,t}$, it is impossible for any PPTA without knowing the blinding factor Y_i and $H_0(t)^{sk_{FN}sk_i}$. However, solving Y_i and $H_0(t)^{sk_{FN}sk_i}$ with $(y_i, y_{i+1}, y_{i-1}, H_0(t)^{sk_{FN}})$ is a CDH problem. That is, for any PPTA, inferring the private value $w_{i,t}$ during training of our proposed scheme is at least as difficult as the CDH problem in $\mathbb{G}_1$. Thus, any PPTA can know nothing about the private information of user $\mathcal{U}_i$. □

Theorem 4. *Even though user $\mathcal{U}_{i+1}$, $\mathcal{U}_{i-1}$ and the FN are colluded, it is infeasible to deduce any private information of user $\mathcal{U}_i$.*

Proof of Theorem 4. We assume that user $\mathcal{U}_{i+1}$, $\mathcal{U}_{i-1}$ and the FN are colluded. User $\mathcal{U}_i$ sends $(C_{i,t}, Au_{i,t}^1, Au_{i,t}^2)$ to the FN. The FN randomly selects a number $r_{i'} \in \mathbb{Z}_p^*$, and sends $y_{i'}$ to user $\mathcal{U}_{i-1}$ and $\mathcal{U}_{i+1}$. $\mathcal{U}_{i-1}$ and $\mathcal{U}_{i+1}$ compute $Y'_{i-1} = (y_i/y_{i'})^{r_{i-1}}$ and $Y'_{i+1} = (y_{i'}/y_i)^{r_{i+1}}$, respectively, and send them to the FN. The FN can remove blinding factor Y_i. The FN first computes $Y_{i'} = (y_{i-1}/y_{i+1})^{r_{i'}}$, and then computes $C_{i,t} \cdot Y_{i'} \cdot Y'_{i-1} \cdot Y'_{i+1} = (1 + p \cdot w_{i,t})H_0(t)^{sk_{FN}sk_i}$. To obtain the privacy information $w_{i,t}$, it is impossible without knowing the blinding factor $H_0(t)^{sk_{FN}sk_i}$. However, solving $H_0(t)^{sk_{FN}sk_i}$ with $H_0(t)^{sk_{FN}}$ is a CDH problem. That is, for any PPTA, inferring the private value $w_{i,t}$ during training of our proposed scheme is at least as difficult as the CDH problem in $\mathbb{G}_1$. Thus, even though user $\mathcal{U}_{i+1}$, $\mathcal{U}_{i-1}$ and the FN are colluded, it is infeasible to deduce any private information of user $\mathcal{U}_i$. □

Theorem 5. *Even though the user $\mathcal{U}_{i+1}$, $\mathcal{U}_{i-1}$ and the server are colluded, it is infeasible to deduce any private information of user $\mathcal{U}_i$.*

Proof of Theorem 5. We assume that user $\mathcal{U}_{i+1}$, $\mathcal{U}_{i-1}$ and the server are colluded. The server will obtain $(C_{i,t}, H_0(t)^{sk_{FN}})$. The server randomly selects a number $r_{i'} \in \mathbb{Z}_p^*$, and sends $y_{i'}$ to user $\mathcal{U}_{i-1}$ and $\mathcal{U}_{i+1}$. $\mathcal{U}_{i-1}$ and $\mathcal{U}_{i+1}$ compute $Y'_{i-1} = (y_i/y_{i'})^{r_{i-1}}$ and $Y'_{i+1} = (y_{i'}/y_i)^{r_{i+1}}$, respectively, and send them to the server. The server can remove blinding factor Y_i. The server first computes $Y_{i'} = (y_{i-1}/y_{i+1})^{r_{i'}}$, and then computes $C_{i,t} \cdot Y_{i'} \cdot Y'_{i-1} \cdot Y'_{i+1} = (1 + p \cdot w_{i,t})H_0(t)^{sk_{FN}sk_i}$. To obtain the privacy information $w_{i,t}$, it is impossible without knowing the blinding factor $H_0(t)^{sk_{FN}sk_i}$. However, solving $H_0(t)^{sk_{FN}sk_i}$ with $H_0(t)^{sk_{FN}}$ is a CDH problem. That is, for any PPTA, inferring the private value $w_{i,t}$ during training of our proposed scheme is at least as difficult as the CDH problem in $\mathbb{G}_1$. Thus, even though the user $\mathcal{U}_{i+1}$, $\mathcal{U}_{i-1}$ and the server are colluded, it is infeasible to deduce any private information of user $\mathcal{U}_i$. □

6.3. Verifiability

Theorem 6. *Even though some users and the server are colluded, this scheme realizes aggregate unforgeability and verifiability.*

Proof of Theorem 6. We assume that some users and the server are colluded and that the server obtains $h_2^{sk_{CS}\sum_{i=1}^{n} sk_i}$. If the server wants to forge the proof of correctness of the aggregated results, it must obtain $H_1(t)^{sk_{CS}\sum_{i=1}^{n} sk_i}$. In other words, the server must obtain $\sum_{i=1}^{n} sk_i$. However, to solve $\sum_{i=1}^{n} sk_i$ with $h_2^{sk_{CS}\sum_{i=1}^{n} sk_i}$, which is a discrete logarithm problem. That is, it is at least as difficult for the aggregation server to falsify the proof of correctness of the aggregation result during the training period of our proposed scheme and convince all honest users as it is for the discrete logarithm problem. Thus, even though some users and the server are colluded, this scheme realizes aggregate unforgeability and verifiability. □

6.4. Fault Tolerance

Theorem 7. *Even though some users drop out, the server is still able to aggregate the encryption gradients of other online users.*

Proof of Theorem 7. Suppose that there is a user $\mathcal{U}_i$ that fails to execute the protocol after the *Keys Broadcasting* phase, but $\mathcal{U}_{i+1}$ and $\mathcal{U}_{i-1}$ use the parameter $g_1^{r_i}$ of $\mathcal{U}_i$ to compute blinding factor Y_{i+1} and Y_{i-1}. Therefore, we need to eliminate the effects of the $\mathcal{U}_i$ drop or the server will not be able to aggregate correctly. We need to consider two scenarios.

Scenario I: If user $\mathcal{U}_i$ drops out accidentally and does not send a message to the FN, the FN issues an announcement to recruit a new user; a new user $\mathcal{U}_{i'}$ is willing to join the system.

The FN sends system parameters and $(\{y_i\}_{i\in U_1}, H_0(t)^{sk_{FN}}, H_0(t)^{sk_{CS}}, h_2^{sk_{CS}})$ to user $\mathcal{U}_{i'}$. User $\mathcal{U}_{i'}$ trains a local model $w_{i',t}$ with their own private dataset and randomly selects a secret number $r_{i'} \in \mathbb{Z}_p^*$. Then, user $\mathcal{U}_{i'}$ computes $y_{i'}$, $Y_{i'}$, $C_{i',t}$, $Au_{i',t}^1$, $Au_{i',t}^2$, $Au_{i',t}^3$. Afterwards, user $\mathcal{U}_{i'}$ sends $(y_{i'}, C_{i',t}, Au_{i',t}^1, Au_{i',t}^2, Au_{i',t}^3)$ to the FN. When the FN receives the message from user $\mathcal{U}_{i'}$, it sends $y_{i'}$ to user $\mathcal{U}_{i-1}$ and $\mathcal{U}_{i+1}$. $\mathcal{U}_{i-1}$ and $\mathcal{U}_{i+1}$ compute $Y'_{i-1} = (y_{i'}/y_i)^{r_{i-1}}$ and $Y'_{i+1} = (y_i/y_{i'})^{r_{i+1}}$, respectively, and send them to the FN.

FN computes

$$C'_{i',t} = C_{i',t} \cdot Y'_{i-1} \cdot Y'_{i+1} \pmod{p^2}$$

Scenario II: If user $\mathcal{U}_i$ drops out accidentally and do not send a message to the FN, the FN issues an announcement to recruit a new user, but no new users are willing to join the system.

The FN randomly selects a number $r_{i''} \in \mathbb{Z}_p^*$, sends $y_{i''}$ to to user $\mathcal{U}_{i-1}$ and $\mathcal{U}_{i+1}$. $\mathcal{U}_{i-1}$ and $\mathcal{U}_{i+1}$ compute $Y''_{i-1} = (y_{i''}/y_i)^{r_{i-1}}$ and $Y''_{i+1} = (y_i/y_{i''})^{r_{i+1}}$, respectively, and send them to the FN. The FN first computes $Y_{i''} = (y_{i+1}/y_{i-1})^{r_{i''}}$, and then computes a fake gradient ciphertext

$$C_{i'',t} = (1 + 0 \cdot p) \cdot Y_{i''} \cdot Y''_{i-1} \cdot Y''_{i+1} \pmod{p^2}$$

The FN and server will be able to eliminate the adverse effects of user $\mathcal{U}_i$ dropping out with $C'_{i',t}$ and $C_{i'',t}$, and the server can aggregate correctly. □

7. Results

7.1. Experimental Setup

The proposed scheme in this paper is mainly applied to IoT scenarios characterized by clients with limited resources and servers with abundant resources. Therefore, this paper is mainly concerned with the computational and communication costs on the user side. To implement a prototype, we utilized a library written in Java [41]. Table 2 shows the details of the hardware and software configurations required for the experiments.

The large prime p is set to 512 bits in the experiments in this paper. The length of the elements in the cyclic groups $\mathbb{G}_1$ is 512 bits.

Table 2. Software and hardware configurations.

Software and Hardware	Configurations
OS	Ubuntu 16.04 64 bit LTS
CPU	Intel(R) Xeon(R) Gold 6130 2.10 GHz
Memory	256 GB
Storage	24 TB
GPU	NVIDIA RTX 2080 Ti
Graphics memory	11 GB
Hardware acceleration	4352 CUDA Cores

7.1.1. Train Model

In the course of model training, we utilized three widely recognized public datasets: MNIST, Fashion-MNIST, and CIFAR-10. These datasets each consist of a training and testing set. We split the training set of each dataset into twenty equal segments, allocating each segment to one out of twenty users. In every iteration of training, ten of these users were randomly chosen to engage in the training session.

We employed these three datasets to train three distinct CNN models. For both MNIST and Fashion-MNIST, the architecture of the models we constructed was identical, featuring two convolutional layers followed by two fully connected layers. However, the input size of the first fully connected layer differed between the two models: for MNIST, it was 1024, whereas for Fashion-MNIST, it was 33856. As for CIFAR-10, the model design incorporated two convolutional layers and subsequently three fully connected layers.

7.1.2. Hyperparameter Settings

During local model training, this paper uses the stochastic gradient descent (SGD) optimizer for all three datasets mentioned above. Table 3 demonstrates the experimental settings for different datasets.

Table 3. Experimental settings on different datasets.

Settings	MNIST	Fashion-MNIST	CIFAR-10
Model	CNN	CNN	CNN
Learning rate	0.005	0.005	0.001
Batch size	16	16	128
Local epoch	1	1	1
Optimizer	SGD	SGD	SGD

7.1.3. Accuracy

Typically, model parameters are represented as floating-point numbers. But when it comes to encrypting these parameters, we need to convert them from floating-point to positive integers. As outlined in the method by [42], we effectively transformed floating-point numbers into positive integers using the following approach: In our study, each floating-point number w is scaled to a positive integer $x = \alpha \times (w + L)$, where α is a scaling factor, L is a sufficiently large positive integer, α equals 10^k, and k represents the precision level. The choice of k influences the model's accuracy.

Figures 3–5 highlight the discrepancy in model accuracy between the FedAVG [2] method using floating-point numbers and that quantizing model parameters to positive integers. Experimental findings reveal that when k is set to 4, the FedAVG method employing floating-point numbers outperforms the quantized approach in terms of accuracy. However, as k increases, the gap in accuracy between the two methods markedly narrows. Specifically, when k reaches 5, the difference in accuracy falls within an acceptable range; by the time k hits 7, the accuracy levels of both methods are virtually indistinguishable.

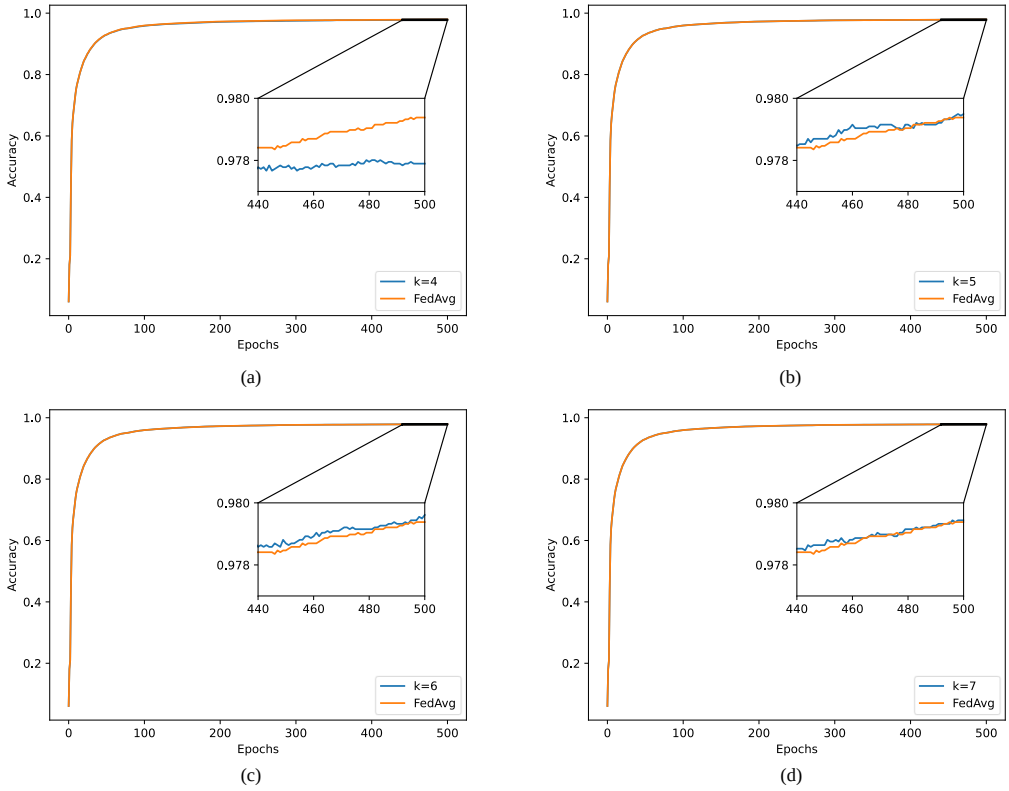


Figure 3. Accuracy comparison between our scheme and FedAvg under different k values on MNIST. (**a**) $k = 4$; (**b**) $k = 5$; (**c**) $k = 6$; (**d**) $k = 7$.

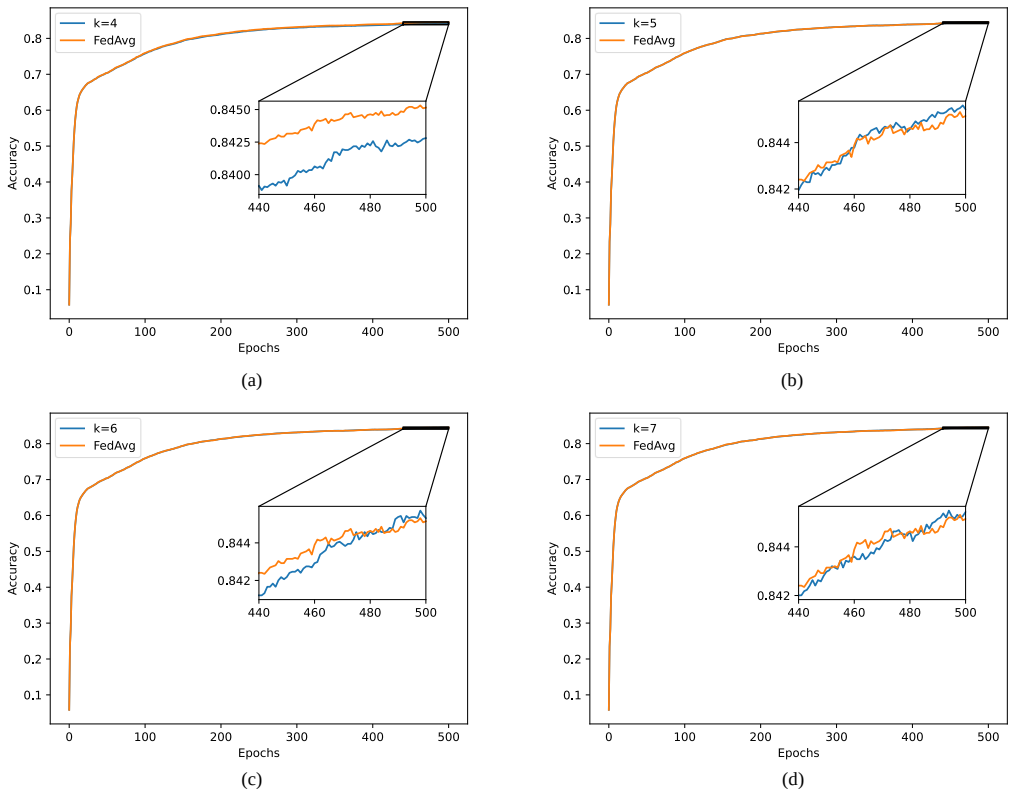


Figure 4. Accuracy comparison between our scheme and FedAvg under different k values on Fashion-MNIST. (**a**) $k = 4$; (**b**) $k = 5$; (**c**) $k = 6$; (**d**) $k = 7$.

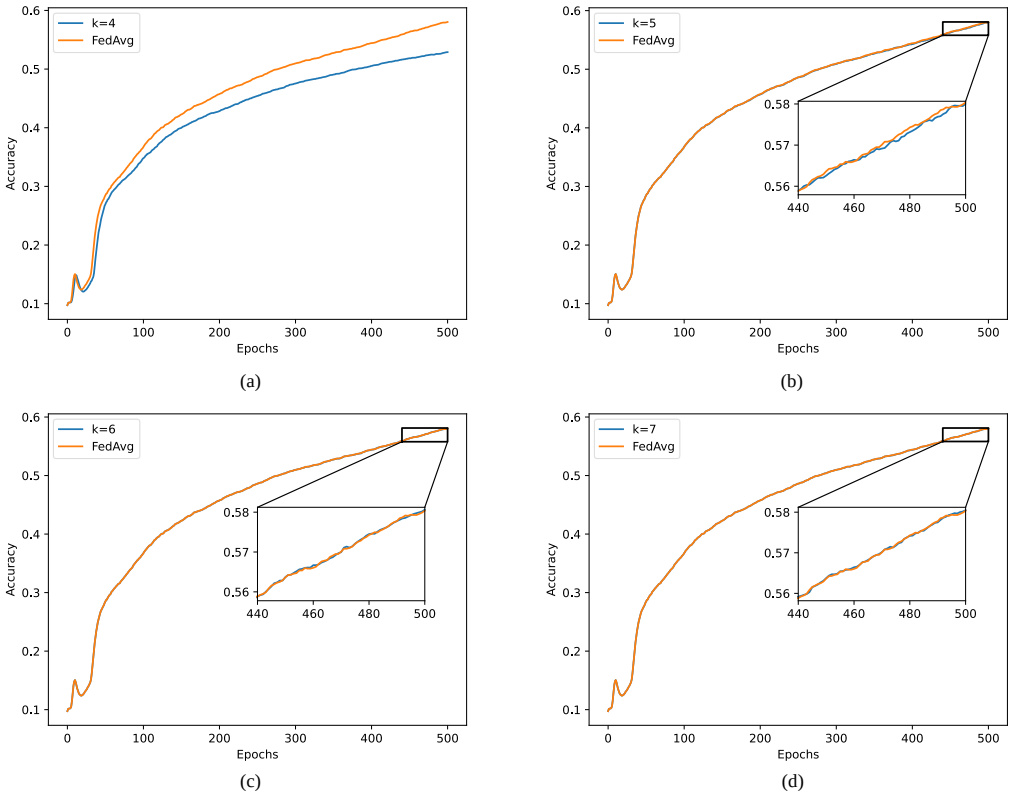


Figure 5. Accuracy comparison between our scheme and FedAvg under different k values on CIFAR-10. (**a**) $k = 4$; (**b**) $k = 5$; (**c**) $k = 6$; (**d**) $k = 7$.

7.2. Computational and Communication Costs

7.2.1. Complexity Analysis

We analyze the computational and communication complexity of the proposed scheme. It is easy to see that each user's computational complexity can be broken up as (1) calculating a secret value in **KeyBroadcasting**, which takes $\mathcal{O}(1)$ time; (2) calculating three parameters $Au_{i,t}^1$, $Au_{i,t}^2$, $Au_{i,t}^2$ and encrypting the local model $w_{i,t}$ in **LocalTraningandMasking**, which takes $\mathcal{O}(n)$ time in total; (3) when some users drop out accidentally, computing a parameter Y' in **MessagesCollecting**, which takes $\mathcal{O}(1)$ time. During each training round, across its four distinct phases, the communication cost incurred by each participating user amounts to $\mathcal{O}(1)$ in **SystemInitialization**, $\mathcal{O}(1)$ in **KeyBroadcasting**, $\mathcal{O}(n)$ in **LocalTraningandMasking** and $\mathcal{O}(n)$ in **AggregatingandUnmasking**. To summarize, the computational and communication complexity of each user is $\mathcal{O}(n)$ and $\mathcal{O}(n)$.

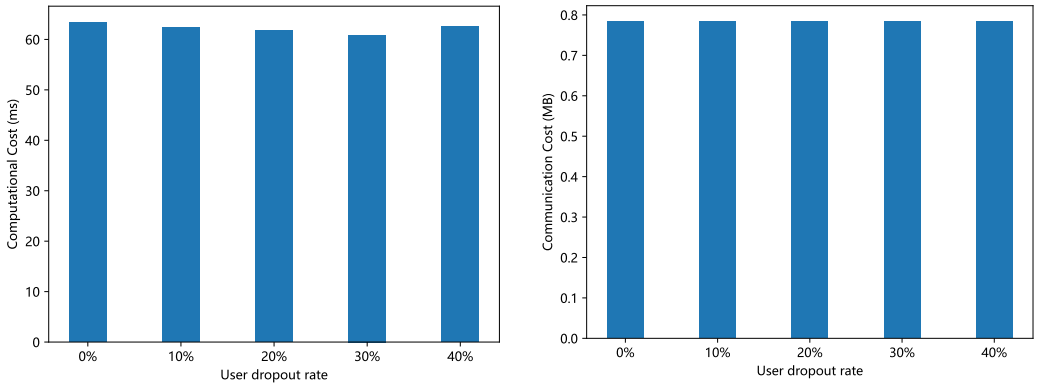
7.2.2. Computational Cost

The computational cost of each user in each epoch is shown in Table 4. From the results, it is not difficult to find that the main computational overhead for each user is in the *Local Training and Masking* phase, because in this phase, each user needs to mask the trained local model gradients and compute three pieces of auxiliary information, and it can also be found that the computational cost of in the *Local Training and Masking* phase increases with the number of gradients, while the other phases do not increase with it. In addition, we evaluated the computational cost incurred by each user in scenarios involving user dropout. As depicted in Figure 6a, the computational cost per user remained relatively stable, showing no considerable variation as the ratio of users dropping out escalated. The rationale for this conclusion arises from the fact that regardless of the user dropout rate, each user is only required to calculate two auxiliary pieces of information in the worst-case scenario, a computation that takes less than one millisecond. Consequently, despite an

increase in the proportion of users exiting the system, the computational cost on a single user remains relatively constant.

Table 4. Computation cost of users at each epoch (ms). "Phase I" denotes system initialization. "Phase II" denotes keys broadcasting. "Phase III" denotes local training and masking. "Phase IV" denotes message collecting. "Phase V" denotes verification.

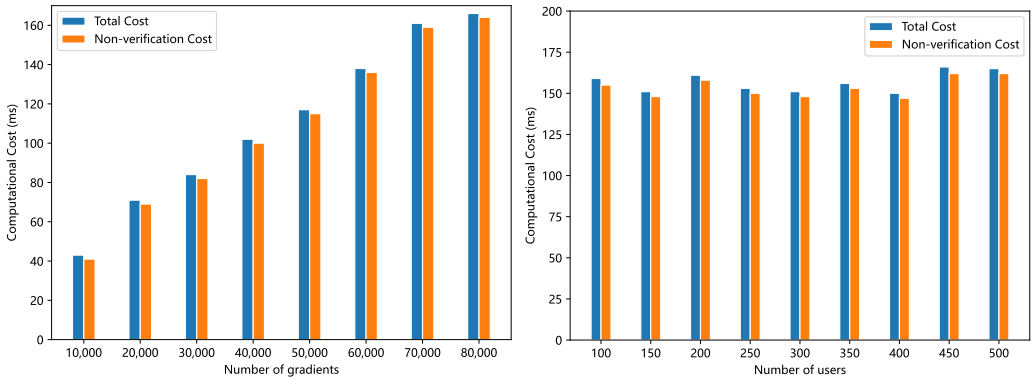
Gradient Numbers	Phase I	Phase II	Phase III	Phase IV	Phase V	Total
1×10^5	0.25	0.2	149.31	0.28	2.67	152.71
2×10^5	0.22	0.22	224.09	0.2	2.59	227.32
3×10^5	0.25	0.24	426.31	0.25	3.57	430.62
4×10^5	0.45	0.2	553.29	0.18	3.47	557.59
5×10^5	0.38	0.2	692.21	0.21	2.46	695.49



(**a**) (**b**)

Figure 6. Computational and communication costs of each user with different dropout rates, $|\mathcal{U}| = 100$, $|\mathcal{G}| = 1 \times 10^4$. (**a**) Computational cost with different dropout rates. (**b**) Communication cost with different dropout rates.

Furthermore, we evaluate the differences in the computational cost between the settings that with verification and non-verification; the result is shown in Figure 7. This result demonstrates that the verification cost in our scheme is very low and stable, with no increases with the number of gradients or users. In addition, it is not difficult to find that the computational cost of each user is only affected by the number of gradients and not the number of users.



(**a**) (**b**)

Figure 7. Computational cost comparison between verification and non-verification for each user. (**a**) $|\mathcal{U}| = 100$, with different numbers of gradients. (**b**) $|\mathcal{G}| = 1 \times 10^5$, with different numbers of users.

To comprehensively evaluate the performance of our proposed scheme, we benchmarked it against existing approaches, namely SA [16], Versa [17], and ESVFL [43]. Comparative results are illustrated in Figure 8. It becomes evident from these results that our proposed scheme significantly reduces computational overhead at the user side compared to alternative methods. Precisely, our scheme decreases computational costs by at least 85% and 89% relative to SA and Versa, respectively. This disparity in performance is largely due to the fact that both SA and Versa rely on an elliptic curve Diffie–Hellman key exchange to construct $n-1$ encryption keys for secure communication and compute $n-1$ pseudorandom number seeds for generating cryptographic parameters used to mask local model gradients. Moreover, Versa requires two masking procedures to enable verification. Notably, in our scheme, each user's computational load does not scale with the number of other participants. Hence, as the number of users grows, the superiority of our proposed scheme over others becomes increasingly pronounced.

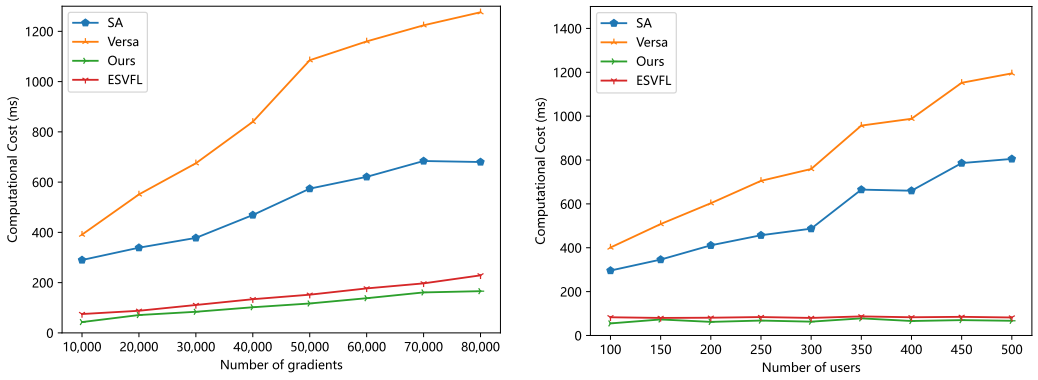
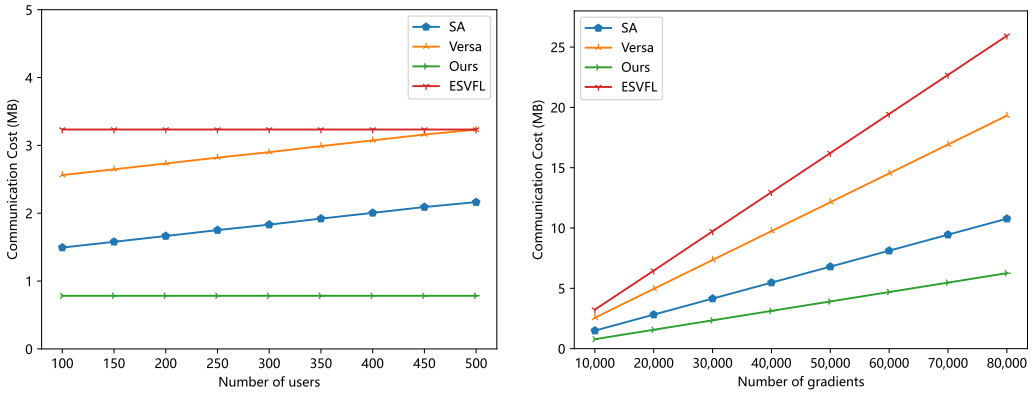


Figure 8. Computational cost comparison of each user. (**a**) $|\mathcal{U}| = 100$, with different numbers of gradients. (**b**) $|\mathcal{G}| = 1 \times 10^4$, with different numbers of users.

7.2.3. Communication Cost

We have recorded and analyzed the communication performance under varying conditions and compared our proposed scheme against existing methods, including SA [16], Versa [17], and ESVFL [43]. The comparative results are depicted in Figure 9. These results clearly demonstrate that our proposed scheme exhibits a significant advantage in terms of communication cost. Specifically, our scheme achieves reductions in communication cost of at least 47%, 69%, and 75% compared to SA, Versa, and ESVFL, respectively. This superior performance stems from the fact that in SA and Versa, each user must transmit a large volume of parameters to either the cloud server or other users, thereby increasing communication overhead. Similarly, in ESVFL, each user is required to send four encrypted gradients, which also contributes to higher communication burdens. Notably, our scheme demonstrates remarkable stability in communication cost, unaffected by an increase in the number of users, with variations primarily influenced by changes in gradient quantities.

Similarly, we evaluated the communication cost per user when users drop out. As shown in 1. Please confirm whether there are repeated formulas in paper. 2. The format of the formulas needs to be consistent. Please check/confirm Figure 6b, there was no significant fluctuation in the communication overhead per user as the proportion of dropout users increased. This stability is attributed to the fact that, irrespective of fluctuations in the user dropout rate, each active user is required to transfer merely two auxiliary pieces of information in the worst-case scenario, a process consuming less than 0.05MB of communication resources. Consequently, despite a rise in the dropout user ratio, the communication cost on a single user did not experience a significant escalation.



(**a**) (**b**)

Figure 9. Communication cost comparison of each user. (**a**) $|\mathcal{U}| = 100$, communication cost comparison with different numbers of gradients. (**b**) $|\mathcal{G}| = 1 \times 10^4$, communication cost comparison with different numbers of users.

8. Conclusions

In this paper, we propose an efficient, verifiable, and privacy-preserving data aggregation scheme for federated learning. The scheme features lower computational and communication costs, making it highly resistant to inference attacks and suitable for scenarios involving IoT devices with relatively weak computational and communication capabilities. To effectively achieve privacy protection and ensure the verifiability of aggregated results, we devised a lightweight encryption protocol and leveraged fog nodes to mitigate the computational and communication costs for users. In addition, we designed a user management mechanism to ensure that the solution proposed in this paper is fault-tolerant. Even if some users drop out due to various issues, the solution proposed in this paper can still be successfully aggregated. Also, users can dynamically join the federated learning process. In the event of worst-case conditions, the incremental computational cost incurred by users remains capped at less than 1 millisecond, and the additional communication cost is maintained at under 0.05 MB. The experiments are conducted on three commonly used datasets in academia and compared with other existing schemes. The experimental results show that the federated learning scheme has high accuracy while protecting the privacy of users' data. In the future, we will consider how to support malicious user detection and prevention.

Author Contributions: Conceptualization, R.S. and L.W.; methodology, R.S. and L.W.; software, R.S.; validation, L.W. and L.Z.; formal analysis, R.S.; writing—original draft preparation, R.S.; writing—review and editing, L.W. and L.Z.; supervision, L.W.; funding acquisition, L.W. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported in part by National Natural Science Foundation of China under Grant 61972241, Natural Science Foundation of Shanghai under Grant 22ZR1427100 and Soft Science Project of Shanghai under Grant 23692106700.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All datasets used in this experiment are publicly available datasets at the following URLs: MNIST. http://yann.lecun.com/exdb/mnist/(accessed on 29 April 2024); Fashion-MNIST. https://github.com/zalandoresearch/fashion-mnist (accessed on 29 April 2024); Cifar-10. https://www.cs.toronto.edu/~kriz/cifar.html (accessed on 29 April 2024).

Acknowledgments: The authors would like to express their gratitude for the support of Fishery Engineering and Equipment Innovation Team of Shanghai High-level Local University.

Conflicts of Interest: The authors declare that this study was conducted without any commercial or financial relationships that could be considered potential conflicts of interest.

References

1. Hasan, M. State of IOT 2022: Number of Connected IOT Devices Growing 18% to 14.4 Billion Globally. 2022. Available online: https://iot-analytics.com/number-connected-iot-devices/ (accessed on 28 April 2024)
2. 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 Statistics, Fort Lauderdale, FL, USA, 20–22 April 2017.
3. Xu, J.; Glicksberg, B.S.; Su, C.; Walker, P.; Bian, J.; Wang, F. Federated learning for healthcare informatics. *J. Healthc. Inform. Res.* **2021**, *5*, 1–19. [CrossRef] [PubMed]
4. Long, G.; Tan, Y.; Jiang, J.; Zhang, C. Federated learning for open banking. In *Federated Learning: Privacy and Incentive*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 240–254.
5. Kevin, I.; Wang, K.; Zhou, X.; Liang, W.; Yan, Z.; She, J. Federated transfer learning based cross-domain prediction for smart manufacturing. *IEEE Trans. Ind. Inform.* **2021**, *18*, 4088–4096.
6. Mothukuri, V.; Parizi, R.M.; Pouriyeh, S.; Huang, Y.; Dehghantanha, A.; Srivastava, G. A survey on security and privacy of federated learning. *Future Gener. Comput. Syst.* **2021**, *115*, 619–640. [CrossRef]
7. 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; IEEE: Piscataway, NJ, USA, 2019; pp. 2512–2520.
8. Zhu, L.; Liu, Z.; Han, S. Deep leakage from gradients. In *Advances in Neural Information Processing Systems 32*; Neural Information Processing Systems Foundation: La Jolla, CA, USA, 2019.
9. Phong, L.T.; Aono, Y.; Hayashi, T.; Wang, L.; Moriai, S. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Trans. Inf. Forensics Secur.* **2017**, *13*, 1333–1345. [CrossRef]
10. Cheon, J.H.; Kim, A.; Kim, M.; Song, Y. Homomorphic encryption for arithmetic of approximate numbers. In *Advances in Cryptology—ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, 3–7 December 2017, Proceedings, Part I 23*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 409–437.
11. Wibawa, F.; Catak, F.O.; Kuzlu, M.; Sarp, S.; Cali, U. Homomorphic encryption and federated learning based privacy-preserving cnn training: Covid-19 detection use-case. In Proceedings of the 2022 European Interdisciplinary Cybersecurity Conference, Barcelona, Spain, 15–16 June 2022; pp. 85–90.
12. Wei, K.; Li, J.; Ding, M.; Ma, C.; Yang, H.H.; Farokhi, F.; Jin, S.; Quek, T.Q.; Poor, H.V. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Trans. Inf. Forensics Secur.* **2020**, *15*, 3454–3469. [CrossRef]
13. Fu, Y.; Zhou, Y.; Wu, D.; Yu, S.; Wen, Y.; Li, C. On the practicality of differential privacy in federated learning by tuning iteration times. *arXiv* **2021**, arXiv:2101.04163.
14. Zhao, C.; Zhao, S.; Zhao, M.; Chen, Z.; Gao, C.Z.; Li, H.; Tan, Y.A. Secure multi-party computation: Theory, practice and applications. *Inf. Sci.* **2019**, *476*, 357–372. [CrossRef]
15. Kalapaaking, A.P.; Stephanie, V.; Khalil, I.; Atiquzzaman, M.; Yi, X.; Almashor, M. SMPC-Based Federated Learning for 6G-Enabled Internet of Medical Things. *IEEE Netw.* **2022**, *36*, 182–189. [CrossRef]
16. Bonawitz, K.; Ivanov, V.; Kreuter, B.; Marcedone, A.; McMahan, H.B.; Patel, S.; Ramage, D.; Segal, A.; Seth, K. Practical secure aggregation for privacy-preserving machine learning. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, Dallas, TX, USA, 30 October–3 November 2017; pp. 1175–1191.
17. Hahn, C.; Kim, H.; Kim, M.; Hur, J. VerSA: Verifiable Secure Aggregation for Cross-Device Federated Learning. *IEEE Trans. Dependable Secur. Comput.* **2023**, *20*, 36–52. [CrossRef]
18. Xu, G.; Li, H.; Liu, S.; Yang, K.; Lin, X. Verifynet: Secure and verifiable federated learning. *IEEE Trans. Inf. Forensics Secur.* **2019**, *15*, 911–926. [CrossRef]
19. Ghodsi, Z.; Gu, T.; Garg, S. Safetynets: Verifiable execution of deep neural networks on an untrusted cloud. In *Advances in Neural Information Processing Systems 30*; Neural Information Processing Systems Foundation: La Jolla, CA, USA, 2017.
20. Tramer, F.; Boneh, D. Slalom: Fast, verifiable and private execution of neural networks in trusted hardware. *arXiv* **2018**, arXiv:1806.03287.
21. 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.* **2020**, *18*, 3316–3326. [CrossRef]
22. Fang, H.; Qian, Q. Privacy Preserving Machine Learning with Homomorphic Encryption and Federated Learning. *Future Internet* **2021**, *13*, 94. [CrossRef]
23. Zhu, Y.; Gong, J.; Zhang, K.; Qian, H. Malicious-Resistant Non-Interactive Verifiable Aggregation for Federated Learning. *IEEE Trans. Dependable Secur. Comput.* **2024**, 1–17. [CrossRef]
24. Ma, J.; Naas, S.A.; Sigg, S.; Lyu, X. Privacy-preserving federated learning based on multi-key homomorphic encryption. *Int. J. Intell. Syst.* **2022**, *37*, 5880–5901. [CrossRef]
25. Zhao, L.; Jiang, J.; Feng, B.; Wang, Q.; Shen, C.; Li, Q. SEAR: Secure and Efficient Aggregation for Byzantine-Robust Federated Learning. *IEEE Trans. Dependable Secur. Comput.* **2022**, *19*, 3329–3342. [CrossRef]
26. Nguyen, T.; Thai, M.T. Preserving Privacy and Security in Federated Learning. *IEEE/ACM Trans. Netw.* **2024**, *32*, 833–843. [CrossRef]

27. Zhang, Z.; Wu, L.; Ma, C.; Li, J.; Wang, J.; Wang, Q.; Yu, S. LSFL: A Lightweight and Secure Federated Learning Scheme for Edge Computing. *IEEE Trans. Inf. Forensics Secur.* **2023**, *18*, 365–379. [CrossRef]
28. Liu, Y.; James, J.; Kang, J.; Niyato, D.; Zhang, S. Privacy-preserving traffic flow prediction: A federated learning approach. *IEEE Internet Things J.* **2020**, *7*, 7751–7763. [CrossRef]
29. Albaseer, A.; Ciftler, B.S.; Abdallah, M.; Al-Fuqaha, A. Exploiting Unlabeled Data in Smart Cities using Federated Edge Learning. In Proceedings of the 2020 International Wireless Communications and Mobile Computing (IWCMC), Limassol, Cyprus, 15–19 June 2020; pp. 1666–1671. [CrossRef]
30. Shokri, R.; Stronati, M.; Song, C.; Shmatikov, V. Membership inference attacks against machine learning models. In Proceedings of the 2017 IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, 22–26 May 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 3–18.
31. Bos, J.W.; Lauter, K.; Naehrig, M. Private predictive analysis on encrypted medical data. *J. Biomed. Inform.* **2014**, *50*, 234–243. [CrossRef] [PubMed]
32. Melis, L.; Song, C.; De Cristofaro, E.; Shmatikov, V. Exploiting unintended feature leakage in collaborative learning. In Proceedings of the 2019 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 20–22 May 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 691–706.
33. Fredrikson, M.; Jha, S.; Ristenpart, T. Model inversion attacks that exploit confidence information and basic countermeasures. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, 12–16 October 2015; pp. 1322–1333.
34. Bhowmick, A.; Duchi, J.; Freudiger, J.; Kapoor, G.; Rogers, R. Protection against reconstruction and its applications in private federated learning. *arXiv* **2018**, arXiv:1812.00984.
35. Hitaj, B.; Ateniese, G.; Perez-Cruz, F. Deep models under the GAN: Information leakage from collaborative deep learning. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, Dallas, TX, USA, 30 October–3 November 2017; pp. 603–618.
36. Park, J.; Yu, N.Y.; Lim, H. Privacy-Preserving Federated Learning Using Homomorphic Encryption With Different Encryption Keys. In Proceedings of the 2022 13th International Conference on Information and Communication Technology Convergence (ICTC), Jeju Island, Republic of Korea, 19–21 October 2022; pp. 1869–1871. [CrossRef]
37. Erlingsson, Ú.; Pihur, V.; Korolova, A. Rappor: Randomized aggregatable privacy-preserving ordinal response. In Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, 3–7 November 2014; pp. 1054–1067.
38. Wang, N.; Xiao, X.; Yang, Y.; Zhao, J.; Hui, S.C.; Shin, H.; Shin, J.; Yu, G. Collecting and analyzing multidimensional data with local differential privacy. In Proceedings of the 2019 IEEE 35th International Conference on Data Engineering (ICDE), Macao, China, 8–11 April 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 638–649.
39. Wang, S.; Huang, L.; Nie, Y.; Zhang, X.; Wang, P.; Xu, H.; Yang, W. Local differential private data aggregation for discrete distribution estimation. *IEEE Trans. Parallel Distrib. Syst.* **2019**, *30*, 2046–2059. [CrossRef]
40. Jung, T.; Li, X.Y.; Wan, M. Collusion-tolerable privacy-preserving sum and product calculation without secure channel. *IEEE Trans. Dependable Secur. Comput.* **2014**, *12*, 45–57. [CrossRef]
41. De Caro, A.; Iovino, V. jPBC: Java pairing based cryptography. In Proceedings of the 16th IEEE Symposium on Computers and Communications, ISCC 2011, Kerkyra, Greece, 28 June–1 July 2011; pp. 850–855.
42. Zhao, P.; Cao, Z.; Jiang, J.; Gao, F. Practical Private Aggregation in Federated Learning Against Inference Attack. *IEEE Internet Things J.* **2022**, *10*, 318–329. [CrossRef]
43. Cai, J.; Shen, W.; Qin, J. ESVFL: Efficient and secure verifiable federated learning with privacy-preserving. *Inf. Fusion* **2024**, *109*, 102420. [CrossRef]