World Electric Vehicle Journal

Article

Asynchronous Robust Aggregation Method with Privacy Protection for IoV Federated Learning

Antong Zhou [1], Ning Jiang [1] and Tong Tang [2,*]

[1] Mashang Consumer Finance Co., Ltd., Chongqing 401121, China
[2] School of Communication and Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing 400065, China
* Correspondence: tangtong@cqupt.edu.cn

Abstract: Due to the wide connection range and open communication environment of internet of vehicle (IoV) devices, they are susceptible to Byzantine attacks and privacy inference attacks, resulting in security and privacy issues in IoV federated learning. Therefore, there is an urgent need to study IoV federated learning methods with privacy protection. However, the heterogeneity and resource limitations of IoV devices pose significant challenges to the aggregation of federated learning model parameters. Therefore, this paper proposes an asynchronous robust aggregation method with privacy protection for federated learning in IoVs. Firstly, we design an asynchronous grouping robust aggregation algorithm based on delay perception, combines intra-group truth estimation with inter-group delay aggregation, and alleviates the impact of stragglers and Byzantine attackers. Then, we design a communication-efficient and security enhanced aggregation protocol based on homomorphic encryption, to achieve asynchronous group robust aggregation while protecting data privacy and reducing communication overhead. Finally, the simulation results indicate that the proposed scheme could achieve a maximum improvement of 41.6% in model accuracy compared to the baseline, which effectively enhances the training efficiency of the model while providing resistance to Byzantine attacks and privacy inference attacks.

Keywords: federated learning; parameter aggregation; privacy protection; internet of vehicles; Byzantine attacker

check for updates

Citation: Zhou, A.; Jiang, N.; Tang, T. Asynchronous Robust Aggregation Method with Privacy Protection for IoV Federated Learning. *World Electr. Veh. J.* **2024**, *15*, 18. https://doi.org/10.3390/wevj15010018

Academic Editor: Joeri Van Mierlo

Received: 9 November 2023
Revised: 20 December 2023
Accepted: 23 December 2023
Published: 4 January 2024

1. Introduction

In the era of the internet of vehicles, massive devices accessing the network generate large-scale interactive data. The huge demand for data processing and analysis has nurtured intelligent transportation systems (ITS), enhancing the capabilities of analysis, decision making, and collaborative processing. Its application scenarios include autonomous driving assistance and computing resource scheduling [1,2]. However, IoV device data are usually unique to the holder, which hinders the development process of ITS. The characteristic of distributed training of federated learning [3,4] can effectively solve this problem by aggregating and updating multi-party parameters, achieving data sharing and joint model training. Recently, emerging technologies such as wireless charging, e.g., [5,6], have been able to ensure increased device uptime and broader participation in federated learning.

Federated learning is a machine learning approach where multiple clients collaboratively train a model while keeping the training data localized. This concept inherently addresses some privacy and data protection concerns by not requiring data to be shared or stored centrally. Each participating device or client works on its own dataset and only shares model updates or gradients, rather than raw data, with a central server or among themselves.

However, due to the lack of reliable security measures on IoV devices, they are susceptible to malicious control and attacks, resulting in model training failures [7]. A typical attack is the Byzantine attack. Byzantine attacks refer to situations in distributed systems where some nodes may exhibit arbitrary malicious behavior. These nodes can send incorrect or deceptive information to disrupt the normal operation of the system. In the context of federated learning, Byzantine attacks could involve sending corrupted model updates to undermine the accuracy of the shared model. Byzantine attackers can send arbitrary model parameter values to the server, posing a huge challenge to widely used aggregation rules [4,8], as a single Byzantine attacker may compromise overall learning performance [9,10]. The robust aggregation algorithm is a commonly used method to resist Byzantine attacks, which can be divided into synchronous robust aggregation and asynchronous robust aggregation. Asynchronous aggregation is a mechanism in distributed systems for merging information or model updates among multiple processes running on different nodes or devices [11], without requiring these processes to be strictly synchronized. In federated learning, this means that updates from devices participating in the learning can be integrated as soon as they become available, rather than waiting for all devices to submit their updates. This helps to speed up the learning process, especially in environments where there are delays or devices operating at inconsistent speeds. Compared to synchronous robust aggregation, asynchronous robust aggregation is more suitable for IoV scenarios with heterogeneous devices, maintaining good training performance even in the presence of laggards. However, existing asynchronous robust aggregation algorithms are susceptible to noise from communication delays and response probabilities [12].

On the other hand, more and more governments are enacting laws to protect user data as data breaches become a serious concern. Examples of these laws are the CCPA [13] in the US, the PDPA [14] in Singapore, and the GDPR [15] in the European Union. Companies bear a hefty price for violating these policies. Uber was forced to pay USD 148 million to resolve an investigation into the 2016 breach of 600,000 drivers' personal information [16]. As of 18 March 2020, Google has been fined USD 57 million for violating the GDPR [17] the highest penalty possible under the EU's privacy law. Hence, there is an urgent need to research new methods to protect data privacy. As a collaborative learning without exchanging users' original data, federated learning is emerging as a new and hot research topic [18]. However, the traditional federated learning privacy protection mechanism is based on a very strong security assumption that the intermediate parameters exposed in each iteration will not leak sensitive information; research has shown that this security assumption is not valid [19,20]. Therefore, researchers propose using secure aggregation protocols to protect data privacy, but this will incur significant communication or computational costs. In addition, there are already federated learning methods with privacy protection, which usually use encryption technology to hide individual model update parameters, and each model is indiscriminately resistant to privacy inference attacks. Resisting Byzantine attacks require a statistical analysis of model updates and correction of model parameters based on differences between models. Therefore, existing federated learning methods with privacy protection are difficult to resist Byzantine attacks, making it extremely challenging to achieve robust aggregation on ciphertext.

To address the issues, this paper proposes an asynchronous robust aggregation method with privacy protection for IoV federated learning, which achieves a bidirectional defense against semi honest servers and Byzantine attackers, achieving a good balance between model accuracy, training efficiency, and communication overhead. In real-world application scenarios, due to network delays or device malfunctions, traditional federated learning deployed in the internet of vehicles (IoV) faces challenges such as untimely model uploads by users and transmission errors in models. Overall, compared to existing methods, our approach takes into account the aforementioned issues, making our model more aligned with real-world scenarios. The main contributions of this paper are as follows:

(1) To alleviate the impact of stragglers and Byzantine attackers, we propose a delay aware grouping method. We design a cosine anomaly filter to remove abnormal gradients, combine intra-group truth estimation and inter-group delay for parameter aggregation, update the weights to obtain intra-group gradient truth values, and then perform inter-group aggregation based on obsolescence.

(2) To save communication resources and improve security, we propose a communication-efficient and security enhanced aggregation protocol, which eliminates the security assumption of communication channels and adopts additive homomorphic encryption to achieve aggregation. The device only needs to communicate once in the initialization phase and perform random number encryption once to reduce communication overhead.

(3) Finally, we derive theoretical boundaries based on key parameters, analyze the relationship between iteration times, runtime, and errors, and conduct privacy analysis on secure aggregation protocols. And we also analyze computational complexity and communication overhead.

The structure of the entire article is outlined as follows: Section 1 serves as the introduction; Section 2 reviews recent work related to the technologies pertinent to this paper; Section 3 details the problem modeling process; Section 4 discusses the problem-solving approach; Section 5 provides theoretical analysis; Section 6 presents the experimental results; and Section 7 presents a comprehensive summary of this paper.

2. Related Work

2.1. Robust Federated Learning

Most of the existing robust federated learning methods focus on synchronous training processes. The ByteChain framework proposed by Li et al. [21] detects Byzantine attacks through proof of accuracy of the Byzantine consensus mechanism, and introduces validators to perform heavy validation workflows to improve model validation efficiency. Zhao et al. [22] generated virtual datasets based on local model inversion, and then identified virtual datasets with abnormal Wasserstein distances and excluded them from global updates. Li et al. [23] proposed an automatic weighted geometric median algorithm that automatically removes abnormal gradients based on Euclidean distance reweighting to achieve robustness, and designed a solution algorithm based on an alternating optimization strategy. The above defense algorithms are all based on synchronous optimization algorithms. However, due to the uneven computing resources of IoV devices, the efficiency of federated learning training is low. The application of asynchronous optimization algorithms in IoV scenarios can effectively solve the bottleneck of synchronous optimization algorithms. But asynchronous optimization algorithms can bring additional noise to random gradients, which makes it more difficult to distinguish the information between Byzantine nodes and honest nodes [12].

In recent years, there has been some research on robust federated learning methods for asynchronous training. Damaskinos et al. [24] proposed a method named Kardam, utilized the Lipschitz condition of gradients to filter out abnormal gradient values, and uses frequency filters to limit the number of times each computing node continuously sends gradient information to the server. However, [12] found that Kardam discards most of the correct gradients in order to filter out all incorrect gradients, making it difficult to effectively resist Byzantine attacks; Xie et al. [12] proposed Zeno++ as a highly computationally efficient version of Zeno, which approximates scores using outdated first-order Taylor expansions and proposes an inert update strategy. However, in IoV scenarios, it is not only difficult to obtain additional validation datasets, but storing data on servers also increases the risk of privacy breaches; Yang et al. [25] proposed a buffered asynchronous random gradient descent algorithm (BASGD), which designs B buffers, each of which stores at least one gradient before executing the SGD step. At the same time, in order to avoid the impact of lagging buffers, a mapping table technique for buffer reallocation is introduced. However, the setting of B in it can greatly reduce the update frequency and damage the training efficiency, while being too small can damage the tolerance of Byzantium.

2.2. Federated Learning with Privacy Protection

In order to address the issue of intermediate parameters exposing sensitive information, servers are required to obtain aggregated information without allowing access to individual gradient information. It will reduce the security of aggregation; thus, researchers proposed secure aggregation protocols to protect data privacy. Bonawitz et al. [26] constructed a dual-mask mechanism to ensure the privacy of intermediate parameters, while avoiding the server's use of time difference attacks to obtain parameter plaintext. However, with each round of device dynamic selection, the secret sharing of random variables between devices results in a linear increase in communication overhead. Zhou et al. [27] calculated sensitivity based on batch training data, combined with privacy budget to obtain the variance of adding noise in differential privacy, and added noise during the device upload parameter and server issue parameter stages, ensuring the privacy of each device and server throughout the aggregation process [28]. Others generate lightweight keys based on the CDH problem to encrypt local model parameters, and do not rely on secure communication channel transmission keys. However, the above secure aggregation protocol can ensure the privacy of model updates, but cannot resist Byzantine attacks.

The existing research on robust federated learning that meets privacy protection needs is still in the exploratory stage, and the main challenges faced by this research are secure robust aggregation in ciphertext form and reliability evaluation of data sources. Li et al. [29] rely on symbol metric correlation to filter out gradient vectors with significant differences, and implement reliability estimation based on a threshold homomorphic encryption system. However, in each iteration, the device side needs to calculate and encrypt reliability information, and also needs to perform ciphertext plaintext exponentiation and upload to the server, which inevitably increases the computational and communication costs of IoV devices. Ma et al. [30] proposed a privacy protection defense strategy based on dual trapdoor homomorphic encryption, which can support standardized update judgment and cosine distance weighted aggregation to defend against Byzantine attacks. However, complex homomorphic encryption operations on high-dimensional gradient vectors in each iteration result in huge computational and communication costs that are not applicable in IoV scenarios. So et al. [31] proposed the Byzantine Tolerable Secure Aggregation Framework (BREA), which ensures the validity of shared values through Feldman verifiable secret sharing. At the same time, they use the homomorphic nature of Shamir secret sharing to implement the Muti Krum robust aggregation rule. However, the transmission of secret shared values between each client requires additional communication overhead, not suitable for IoV scenarios with massive devices.

3. Problem Description

3.1. System Model

As illustrated in Figure 1, the proposed system model contains three layers: the cloud layer, the edge layer, and the perception layer. The cloud layer is responsible for the aggregation of the global model, the Edge layer, which includes edge servers and auxiliary servers, is responsible for the secure and robust aggregation of local models, and the IoV devices of the perception layer are responsible for data collection, model training, and gradient perturbation. The specific operation process is divided into the following four steps:

Step 1: The cloud server initializes the global model, and sends the initial global model to the edge server, which then transmits it to the IoV devices in its jurisdiction.

Step 2: Upon receiving the global model, the IoV devices calculate model updates using local data, and send the perturbed model updates back to the auxiliary server, which then encrypts the perturbed data and sends it back to the edge server.

Step 3: Upon receiving returns from the first K devices, the edge server and the auxiliary server collaborate to achieve asynchronous robust aggregation of the perturbed gradients.

Step 4: The edge server uploads the local model to the cloud server for global model aggregation.

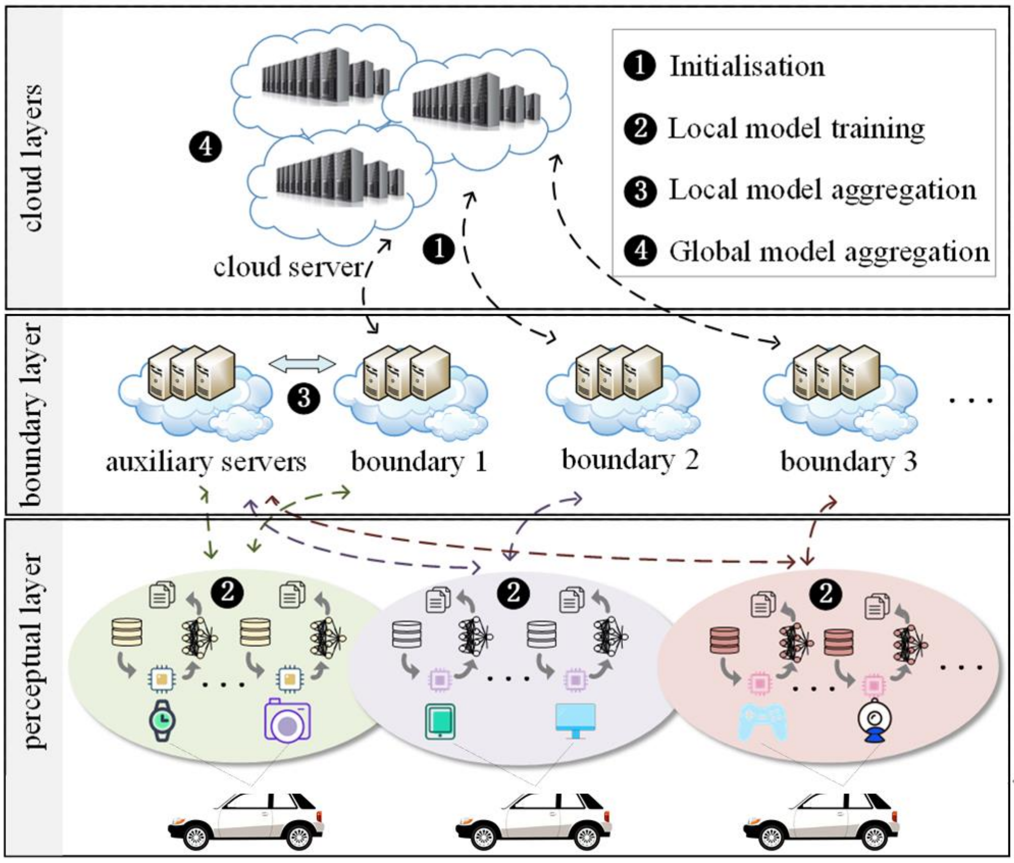


Figure 1. System block diagram.

The above Steps 2–4 are repeated until the model reaches a pre-set threshold or the model converges, at which point the operation is stopped.

3.2. Threat Model

Privacy threat model: We assume that the edge servers and auxiliary servers are honest-but-curious, i.e., they always follow the protocol but try to steal the original information from the devices through the information they receive, leading to a potential privacy leak from the devices.

Security threat model: It is possible that less than half of the participants in the system are Byzantine attackers. They can eavesdrop on the communication channels to obtain transmitted information, or fabricate intercepted information, deliberately uploading false data into the federated learning system.

3.3. Cryptographic Primitive

Next, we will discuss the cryptographic primitives necessary for constructing the scheme.

Homomorphic encryption is a special encryption algorithm where the result obtained by performing operations on multiple ciphertexts is equivalent to the effect produced by directly performing operations on the corresponding plaintexts. In other words, computations can be performed directly on encrypted data using homomorphic encryption system. Let M and C respectively represent the plaintext space and the ciphertext space, the homomorphic encryption algorithm $Enc_{pk}(\cdot)$ can be represented as

$$\forall m_1, m_2 \in M, Enc_{pk}[m_1 \odot_M m_2] = Enc_{pk}[m_1] \odot_C Enc_{pk}[m_2] \tag{1}$$

Here, $\odot_M$ and $\odot_C$ respectively represent the operations in the plaintext space and the ciphertext space. If $\odot_M$ is "+", it is referred to as an additive homomorphic encryption algorithm. This paper adopts a widely used additive homomorphic encryption system, namely the Paillier encryption system [32]. For plaintext $m \in \mathbb{Z}_n$, it can be encrypted into $Enc_{pk}[m] = g^m r^n \text{mod} n^2$ with public key $pk = (n, g)$, where $r \in \mathbb{Z}_n^*$, is secretly randomly selected by the user who calculates the ciphertext. Then, $\forall m_1, m_2, m_3 \in \mathbb{Z}_n$, the following equations represent the additive homomorphic properties and homomorphic multiplication properties of the Paillier encryption system:

$$\begin{gathered} Dec_{sk}(Enc_{pk}[m_1 + m_2]) = Dec_{sk}(Enc_{pk}[m_1] \cdot Enc_{pk}[m_2]) = m_1 + m_2 \\ Dec_{sk}(Enc_{pk}[a \cdot m_3]) = Dec_{sk}(Enc_{pk}[m_3]^a) = a \cdot m_3 \end{gathered} \tag{2}$$

where $Dec_{sk}(\cdot)$ represents the decryption function.

4. Scheme Design

4.1. Asynchronous Grouping Robust Aggregation Algorithm

Low training efficiency and vulnerability to Byzantine attacks can significantly affect the performance of federated learning. To address these issues, we propose an asynchronous robust aggregation algorithm with delay-aware grouping to pursue high efficiency and security, as shown in Figure 2. Specifically, we design a delay-aware grouping method, which, through cosine anomaly filtering, combined with group-wise truth estimation and inter-group delay aggregation, effectively mitigates the impact of stragglers and Byzantine attackers.

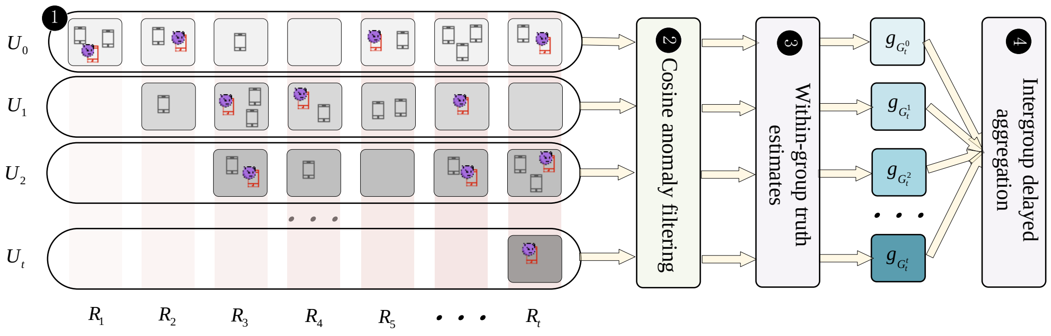


Figure 2. Asynchronous grouping robust aggregation algorithm.

A. Latency-aware Grouping:

The study is based on the K asynchronous SGD algorithm proposed by [33], in which the server waits for the first K devices to finish, rather than waiting for all devices to complete the training. Let represent a group of devices that were selected for training in the jth round and returned gradients in the tth round, then $U_j = \cup_{t=j}^{\infty} G_t^j, t \geq j$ implies a group of devices with the same latency. Here, G_∞^j denotes the devices that were selected in the jth round but failed to return the results to the server successfully, and the groups satisfy $G_a^j \neq G_b^j, a \neq b$.

B. Cosine Anomaly Filtering

Extensive research [34] indicates that once the global model has been attacked, the impact on the global model persists even if there are no subsequent attacks. Therefore, it is necessary to filter out the attackers from the federated learning training system to eliminate the influence of the attackers before the final aggregation.

We choose cosine distance as the measurement criterion. The angle between two vectors in space can determine the similarity of the two vectors, which is called cosine distance.

The closer the cosine value is to 1, the more similar the two vectors. In addition, cosine distance is suitable for the high-dimensional data of model updates, and it is more efficient than calculating the Pearson correlation coefficient and other indicators. For the device $G_t^j(j < t)$, the the same initial round $g_{G_{t-1}^j}$ represents the intra-group update value of the current iteration. However, for the device $G_t^j(j = t)$, the intra-group update value cannot be predicted before all intra-group updates are aggregated in the current iteration.

To solve this problem, we assume that model training is usually a stable convergence, so the difference between two consecutive model updates will be as small as possible [35]. Therefore, we use the global update from the previous iteration to estimate the intra-group update value in the current iteration, calculate the cosine distance between the last iteration's global update and each group's update, and avoid attackers by excluding the cosine distances of outliers through a threshold mechanism. The cosine distance calculation for device k is as follows:

$$cs_k = \frac{< g, g_k >}{||g||\cdot||g_k||}, g = \begin{cases} g_{G_{t-1}^j}, k \in G_t^j(j < t) \\ g_{t-1}, k \in G_t^j(j = t) \end{cases} \tag{3}$$

Here, in the case where most nodes are honest, we sort the calculated cosine distances and use the median as a threshold, which is reliable.

C. Intra-group Truth Estimation:

After filtering out the abnormal cosine distances based on the threshold, within a group of devices with the same latency, we can consider measuring the distance between each dimension to adjust the weight of each device's uploaded gradient and update the gradient truth value of the group accordingly based on the weights. The main content includes the following two steps:

(1) Weight update: First, we need to initialize the value of the gradient truth. Here, we select the median g_{med} of each dimension in the filtered normal values as the initial gradient truth. Then, each device will update its weight based on the difference between its gradient and the initial gradient truth of the group.

$$w_k = \log\left(\frac{\sum_{k=1}^{M}\sum_{i=1}^{N} dist(g_k^i, g_{med}^i)}{\sum_{i=1}^{N} dist(g_k^i, g_{med}^i)}\right), k \in G_t^j, j = 0, 1, \ldots, t \tag{4}$$

where $dist()$ is the function to measure the distance between g_k^i and g_{med}^i, calculated by $dist(g_k^i, g_{med}^i) = (g_k^i, g_{med}^i)^2$, where M represents the number of devices contained in G_t^j, i.e., $M = \left|G_t^j\right|$, and N represents the dimension of the gradient.

(2) Truth estimation: Given each device's weight update, the gradients in each group are aggregated to obtain the group's gradient truth value.

$$g_{G_t^j} = \frac{\sum_{k=1}^{M} w_k g_k}{\sum_{k=1}^{M} w_k}, j = 0, 1, \ldots, t \tag{5}$$

D. Inter-group Delay Aggregation:

For $i = 0, 1, \ldots, t$, starting from the gradient truth value of all groups $g_{G_t^0}, g_{G_t^1}, g_{G_t^2}, \cdots, g_{G_t^t}$, aggregate according to different staleness via delay aggregation. The aggregation formula is as follows:

$$w_{t+1} = w_t - \eta \sum_{j=0}^{t} \Lambda(\tau) g_{G_t^j}, j = 0, 1, \ldots, t \tag{6}$$

where $\Lambda(\tau)$ is any decay function. Here, we set it as $\Lambda(\tau) = {}^1/_{1+\tau}, \tau = t - j$.

4.2. Communication Efficient and Security Enhanced Aggregation Protocol

The flowchart of the communication efficient and security enhanced aggregation protocol is shown in Figure 3, and the four modules are introduced as follows.

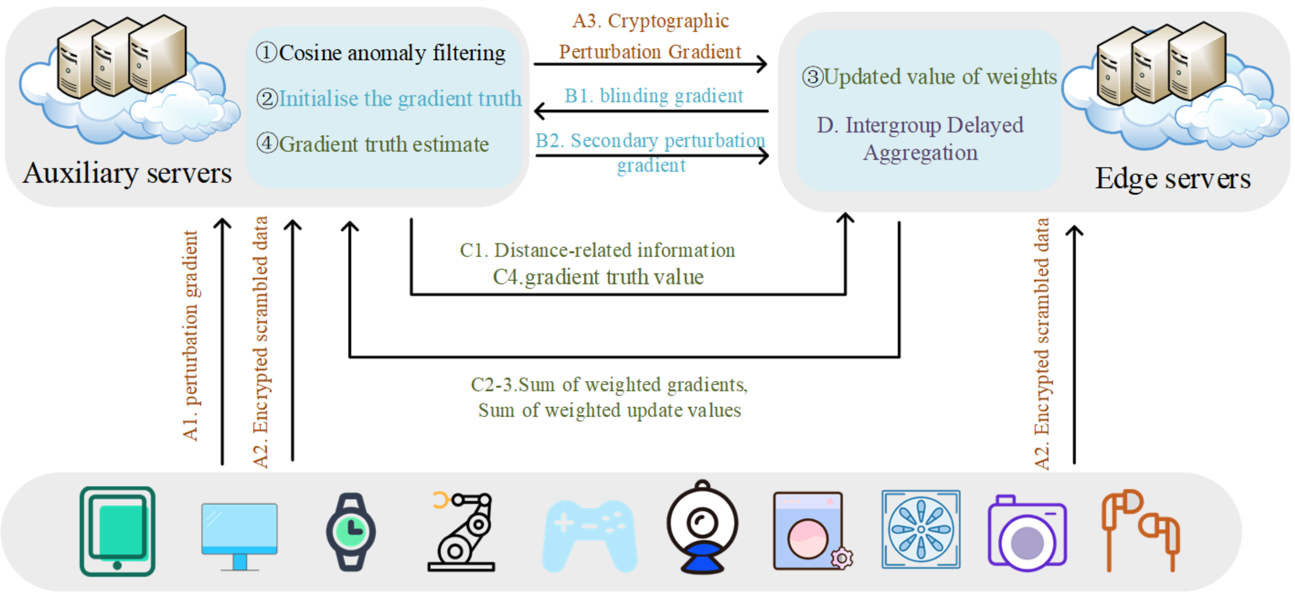

Auxiliary servers
①Cosine anomaly filtering
②Initialise the gradient truth
④Gradient truth estimate
A3. Cryptographic Perturbation Gradient
B1. blinding gradient
B2. Secondary perturbation gradient
③Updated value of weights
D. Intergroup Delayed Aggregation
Edge servers
A1. perturbation gradient
A2. Encrypted scrambled data
C1. Distance-related information
C4.gradient truth value
C2-3.Sum of weighted gradients, Sum of weighted update values
A2. Encrypted scrambled data

Figure 3. Communication efficient and security enhanced aggregation protocol.

A. Initialization Stage:

Edge server S_e and auxiliary server S_a each generate their own public and private keys, (pk_e, sk_e) and (pk_a, sk_a), represented by and as the key pair for S_e and S_a, respectively.

Step 1: The device k generates a random number α_k^i for the local gradient g_k^i, then perturbs g_k^i into $\tilde{g}_k^i = g_k^i - \alpha_k^i$, and returns the perturbed result $\{\tilde{g}_k^i\}_{i=1}^{N}$ to the auxiliary server S_a.

Step 2: The device k encrypts the random number α_k^i into $[\![\alpha_k^i]\!]_{pk_e}$, where $[\![\cdot]\!]_{pk_e}$ indicates encryption with the edge server's public key pk_e, and sends $\left\{ [\![\alpha_k^i]\!]_{pk_e} \right\}_{i=1}^{N}$ to S_e and S_a.

Step 3: The auxiliary server S_a encrypts all received perturbed gradients $\{\tilde{g}_k^i\}_{i,k=1}^{N,M}$ to obtain $\left\{ [\![\tilde{g}_k^i]\!]_{pk_a} \right\}_{i,k=1}^{N,M}$, and sends it to the edge server S_e.

B. Cosine Anomaly Filtering:

This stage is entirely carried out by the cooperation between the edge server S_e and the auxiliary server S_a, and it will not increase the computation and communication volume of the device side.

Step 1: After the edge server S_e receives $\left\{ [\![\tilde{g}_k^i]\!]_{pk_a} \right\}_{i,k=1}^{N,M}$, it calculates $[\![\tilde{g}_k^i]\!]_{pk_a} \cdot [\![\alpha_k^i]\!]_{pk_a}{}_{i,k=1}^{N,M}$ to obtain $\left\{ [\![g_k^i]\!]_{pk_a} \right\}_{i,k=1}^{N,M}$. To prevent the auxiliary server from directly viewing the device's gradient information, it randomly selects r for blinding treatment, calculates $\left\{ [\![g_k^i]\!]_{pk_a}^{r} \right\}_{i,k=1}^{N,M}$, and returns it to the auxiliary server. The auxiliary server decrypts it to obtain $\{r \cdot g_k^i\}_{i,k=1}^{N,M}$. The auxiliary server, having gradient information $g_{G_{t-1}^j}$ and g_{t-1}, can obtain the cosine distance to obtain cs_k.

Step 2: Having obtained all the cosine distances, sort them and select the median to screen the device gradient of the normal cosine distance. At the same time, the screened device gradient is subjected to secondary perturbation, select a random number s, calculate to obtain $\{\widetilde{g}_k^i + s\}_{i,k}^{N,M}$, and send it to the edge server.

C. Intra-group Truth Estimation Stage:

Step 1: After the edge server S_e receives $\{\widetilde{g}_k^i + s\}_{i,k}^{N,M}$, and by adding $\sum_{k=1}^{M}\sum_{i=1}^{N}\alpha_k^i$, it obtains $\{g_k^i + s\}_{i,k}^{N,M}$. The edge server compares to obtain the median value $\{g_{med}^i + s\}_{i=1}^{N}$ of each dimension and returns the initial gradient truth to the auxiliary server S_a.

Step 2: After the auxiliary server receives $\{g_{med}^i + s\}_{i=1}^{N}$, it removes the secondary perturbation value s and performs the following calculations:

$$\begin{aligned} T_{total} &= \prod_{k=1}^{M}\prod_{i=1}^{N} [\![(\widetilde{g}_k^i - g_{med}^i)^2]\!]_{pk_e} \cdot [\![\alpha_k^i]\!]_{pk_e}^{2(\widetilde{g}_k^i - g_{med}^i)} \\ &= \prod_{k=1}^{M}\coprod_{i=1}^{N} [\![(g_k^i - \alpha_k^i - g_{med}^i)^2 + 2\alpha_k^i(g_k^i - \alpha_k^i - g_{med}^i)]\!]_{pk_e} \\ &= \coprod_{k=1}^{M}\coprod_{i=1}^{N} [\![(g_k^i - g_{med}^i)^2 - (\alpha_k^i)^2]\!]_{pk_e} \end{aligned} \tag{7}$$

$$\begin{aligned} T_{\sin\mathrm{gle}} &= \prod_{i=1}^{N} [\![(\widetilde{g}_k^i - g_{med}^i)^2]\!]_{pk_e} \cdot [\![\alpha_k^i]\!]_{pk_e}^{2(\widetilde{g}_k^i - g_{med})} \\ &= \coprod_{i=1}^{N} [\![(g_k^i - \alpha_k^i - g_{med}^i)^2 + 2\alpha_k^i(g_k^i - \alpha_k^i - g_{med}^i)]\!]_{pk_e} \\ &= \coprod_{i=1}^{N} [\![(g_k^i - g_{med}^i)^2 - (\alpha_k^i)^2]\!]_{pk_e}, k = 1,2,3\ldots,M \end{aligned} \tag{8}$$

and returns the above calculation results T_{total} and $T_{\sin\mathrm{gle}}$ to the edge server S_e.

Step 3: After the edge server S_e receives T_{total} and $T_{\sin\mathrm{gle}}$, it decrypts using sk_e to obtain $\left\{(g_k^i - g_{med}^i)^2 - (\alpha_k^i)^2\right\}_{i,k=1}^{N,M}$ and $\left\{(g_k^i - g_{med}^i)^2 - (\alpha_k^i)^2\right\}_{i=1}^{N}$. Adding $\sum_{k=1}^{M}\sum_{i=1}^{N}(\alpha_k^i)^2$ and $\sum_{i=1}^{N}(\alpha_k^i)^2$ gives the sum of the distances $\sum_{k=1}^{M}\sum_{i=1}^{N} dist(g_k^i, g_{med}^i)$ between all device gradients and the initial gradient truth, and the distance between each device gradient and the initial gradient truth $\sum_{i=1}^{N} dist(g_k^i, g_{med}^i)$, and further calculates each device's weight update value $w_0, w_1, w_2, \ldots, w_k$.

Step 4: The edge server S_e calculates $\sum_{k=1}^{M} w_k\alpha_k$ based on the weight update value, encrypts with the auxiliary server's public key pk_a, and then S_e calculates:

$$\begin{aligned} [\![\textstyle\sum_{k=1}^{M} w_k g_k]\!]_{pk_a} &= \prod_{k=1}^{M} [\![\widetilde{g}_k]\!]_{pk_a}^{w_k} \cdot [\![\textstyle\sum_{k=1}^{M} w_k\alpha_k]\!]_{pk_a} \\ &= [\![\textstyle\sum_{k=1}^{M} w_k\widetilde{g}_k]\!]_{pk_a} \cdot [\![\textstyle\sum_{k=1}^{M} w_k\alpha_k]\!]_{pk_a} \\ &= [\![\textstyle\sum_{k=1}^{M} w_k(g_k - \alpha_k) + \sum_{k=1}^{M} w_k\alpha_k]\!]_{pk_a} \end{aligned} \tag{9}$$

and returns the sum of the weighted gradients $[\![\sum_{k=1}^{M} w_k g_k]\!]_{pk_a}$ of all devices and the sum of the weight $\sum_{k=1}^{M} w_k \sum_{k=1}^{M} w_k$ update values of all devices to the auxiliary server S_a.

Step 5: After the auxiliary server S_a obtains $[\![\sum_{k=1}^{M} w_k g_k]\!]_{pk_a}$ and $\sum_{k=1}^{M} w_k$, it gets each group's gradient truth value $g_{G_t^0}, g_{G_t^1}, g_{G_t^2}, \ldots, g_{G_t^t}$ according to the truth estimation formula and sends it to the edge server S_e.

D. Inter-group Delay Aggregation Stage:

This stage is completed entirely by the edge server S_e, independently implementing inter-group delay aggregation. Aggregation is based on the delay of each device's uploaded gradient by S_e.

Throughout the entire process of the above protocol, the device only needs to participate in the initialization stage, and the protocol does not require the device to encrypt an N dimensional gradient vector in each round. It only needs to encrypt an N dimensional random number in the first round. While ensuring the security of the communication link, it greatly reduces the computational burden on the device.

5. Theoretical Analysis

5.1. Privacy Analysis

Theorem 1. *In the secure aggregation protocol, if the edge server and the auxiliary server are honest but curious, and there is no collusion between them, the gradient information of each device will not be obtained by anyone.*

Proof. In the secure aggregation protocol, since the information transmitted in the link is protected by homomorphic encryption and cannot be cracked, and each participating device does not receive any information about other devices, we only need to prove that the gradient information of each device will not be stolen by the edge server and the auxiliary server. □

For the auxiliary server S_a, the information obtained includes ciphertext $\left\{ [\![\alpha_k^i]\!]_{pk_e} \right\}_{i,k=1}^{N,M}$, T_{total}, $T_{\sin\text{gle}}$, and plaintext $\left\{\tilde{g}_k^i\right\}_{i,k=1}^{N,M}$, $\left\{r \cdot g_k^i\right\}_{i,k=1}^{N,M}$, $\{cs_k\}_{k=1}^{M}$, $\left\{g_{med}^i\right\}_{i=1}^{N}$, $\sum_{k=1}^{M} w_k g_k$, $\sum_{k=1}^{M} w_k$, $\left\{g_{G_t^j}\right\}, j \leq t$. Without the private key sk_e of the edge server S_e, the above ciphertext information cannot be decrypted by S_a. Since the weight update value $w_0, w_1, w_2, \ldots, w_k$ of each device estimated on the edge server is not sent to the auxiliary server S_a, S_a cannot infer the weight update value of each device just based on the sum of the weighted gradients of all devices $\sum_{k=1}^{M} w_k g_k$ and the sum of the weight update values of all devices $\sum_{k=1}^{M} w_k$. At the same time, it is difficult to restore the original gradient when only obtaining the cosine distance $\{cs_k\}_{k=1}^{M}$ and the group update values $g_{G_{t-1}^j}$ and g_{t-1}. Finally, since the edge server and the auxiliary server do not collude, the edge server will not infer the gradient information of each device from $\left\{\tilde{g}_k^i\right\}_{i,k=1}^{N,M}$ and $\left\{r \cdot g_k^i\right\}_{i,k=1}^{N,M}$.

For the edge server S_e, the information obtained includes ciphertext $\left\{ [\![\tilde{g}_k^i]\!]_{pk_a} \right\}_{i,k=1}^{N,M}$, $\left\{ [\![g_k^i]\!]_{pk_a} \right\}_{i,k=1}^{N,M}$, $\left\{ [\![g_k^i]\!]_{pk_a}^{r} \right\}_{i,k=1}^{N,M}$, $[\![\sum_{k=1}^{M} w_k g_k]\!]_{pk_a}$, and plaintext $\left\{\alpha_k^i\right\}_{i,k=1}^{N,M}$, $\left\{g_k^i + s\right\}_{i,k}^{N,M}$, $\left\{g_{med}^i + s\right\}_{i=1}^{N}$, $\sum_{k=1}^{M}\sum_{i=1}^{N} dist(g_k^i, g_{med}^i)$, $\sum_{i=1}^{N} dist(g_k^i, g_{med}^i)$, $\{w_k\}_{k=1}^{M}$, $\left\{g_{G_t^j}\right\}, j \leq t$. Without the private key sk_a of the auxiliary server S_a, the above ciphertext information cannot be decrypted by S_e. Therefore, S_e cannot obtain the gradient information of each device based on the encrypted perturbed gradient $\left\{ [\![\tilde{g}_k^i]\!]_{pk_a} \right\}_{i,k=1}^{N,M}$ and the perturbed data $\left\{\alpha_k^i\right\}_{i,k=1}^{N,M}$. In addition, since the edge server and the auxiliary server do not collude, the second perturbation value s on the auxiliary server S_a is not sent to the edge server S_e, so S_e cannot obtain the initial gradient truth $\left\{g_{med}^i\right\}_{i=1}^{N}$, and therefore cannot learn the gradient information of a single device from $\sum_{i=1}^{N} dist(g_k^i, g_{med})$.

In summary, in the secure aggregation protocol, the gradient information of each device will not be obtained by anyone.

5.2. Efficiency Analysis

Computational complexity: According to the privacy computation process, in lightweight secure aggregation, devices only participate in the initialization phase, and the computation cost for each device is $O(N)$ encryption operations. For edge servers and auxiliary servers, both involve cosine anomaly filtering, the initialization phase, and the within-group truth estimation phase. The auxiliary server needs to perform $O(MN)$ encryption operations, $O(MN)$ ciphertext multiplications, and power operations, and $O(M)$ decryption operations. In addition, the edge server needs to perform $O(MN)$ ciphertext power operations, $O(MN)$ and

$O(M)$ decryption operations, $O(N)$ encryption operations, $O(MN)$ ciphertext multiplications, and $O(M)$ ciphertext power operations.

Communication overhead: In the secure aggregation protocol, since the devices participating in the training only participate in the initialization stage, there are only 2 rounds of communication between each device and the server throughout the protocol process. In the initialization stage, the auxiliary server needs to send data once, and in the cosine anomaly filtering and within-group truth estimate stage, the edge server and the auxiliary server send data to each other three times, so the communication overhead between the edge server and the auxiliary server is 3 rounds, and the communication overhead between the auxiliary server and the edge server is 4 rounds.

6. Experimental Result

6.1. Experimental Design

(1) Attack mode

Symbol flipping attack: Each Byzantine attacker calculates true gradient information g, then sends $g = cg$ to the server, in our experiments we set $c = -10$ as in [25].

Gaussian attack: Each Byzantine attacker sends information g to the server, each component in g is sampled from $N(0, 200)$ as in [9].

Noise attack: Each Byzantine attacker adds Gaussian noise ε to the real gradient and then sends $g = g + \varepsilon$ to the server, as in [25], ε is randomly sampled from the normal distribution $N(0, \|\sigma_{atk} g\|^2 \cdot I)$ and added to each component, where σ_{atk} is a parameter, for image recognition task $\sigma_{atk} = 0.2$, for natural language processing tasks, I is an identity matrix.

(2) Baseline

We selected asynchronous SGD that has not been attacked as the gold standard, select Kardam [24], Zeno++ [12], BASGD [25], BREA [31] as the baseline algorithms, where Kardam, Zeno++, and BASGD are asynchronous robust aggregation algorithms, BREA is a robust aggregation method with privacy protection.

(3) Datasets and models

The experiment was conducted on the image classification dataset CIFAR-10 [36], which consists of 50,000 training samples and 10,000 test samples, and Resnet-20 [37] was used as the training model.

(4) Client settings and dataset partitioning

We set up 30 clients, where the proportion of Byzantine attackers in all clients is set to $q = 0 - 20\%$. Asynchronous settings reference [25], k_{del} is a parameter that characterizes delay, randomly extracted from the truncated standard normal distribution of $[0, +\infty)$. We adopted an IID partitioning method for the dataset, and the sample partitioning between clients does not intersect.

(5) Evaluation indicators

We evaluated the performance of the algorithm using three indicators: model accuracy, convergence speed, and communication overhead. Model accuracy is the proportion of correctly classified samples for all samples in the dataset, and the global model is tested on the server side every round. The convergence speed is the accuracy that the model can achieve under a fixed number of rounds. The communication overhead is the total amount of data that needs to be transmitted in a single protocol execution.

6.2. Performance Comparison

We conducted extensive experiments on the defense performance of asynchronous robust aggregation algorithms under various Byzantine attacks under heterogeneous device settings, and compared our proposed method with the existing defense methods, including Kardam, Zeno++, BASGD. Numerical results demonstrated the effectiveness of our method. Next, we compared the accuracy of the model under different ratios of stragglers to observe the convergence speed. Moreover, we tested the communication overhead of the proposed method and compared it with the existing privacy protection robust aggregation scheme BREA.

(1) Model accuracy comparison

To verify the good performance of our method in heterogeneous device settings, we conducted two sets of experiments including $q = 10\%$ and $q = 20\%$ for each type of attack. In these two sets of experiments, we set 10 and 15 buffers for BASGD, and combined them with the media algorithm. Moreover, we set the decay function of Kardam to a to ensure the fairness of the comparison, where the proportion of stragglers is fixed at 20%, and AGRA is compared with existing asynchronous robust aggregation algorithms under the same experimental settings.

It can be seen from Figures 4 and 5 that firstly, under three types of attacks, model accuracy of the proposed method is obviously better than that of ASGD, Zeno++, BASGD, Kardam. Then, under less harmful attacks of symbol flipping attack and Gaussian attack, although ASGD and Kardam still could converge, both their accuracies significantly decrease. And under noise attack, neither ASGD nor Kardam can converge. There are two main reasons for this. On the one hand, Kardam not only filters out 100% of Byzantine gradients, but also filters out nearly 100% of correct gradients. Zeno++ has shown that Kardam has a false positive rate of up to 99%, which makes its convergence very slow and performance poor. On the other hand, the threat model used in this chapter does not guarantee Kardam's important assumption that any continuously received gradient sequence of length $2q + 1$ must contain at least gradients from honest participant $q + 1$. This is difficult to guarantee in asynchronous environments, as Byzantine attackers can easily send continuous Byzantine gradients, and the methods in this chapter do not rely on such strong assumptions. Finally, as the proportion of Byzantine attackers increases, the model accuracy of our method decreases the least under the three types of attacks. This is attributed to the use of delayed grouping strategy to avoid the difficulty of comparing different delay gradients, and the use of media as the benchmark algorithm to accurately evaluate the aggregation of multiple gradient information and obtain the true value gradient. Therefore, our proposed method has the best performance under these three types of attacks.

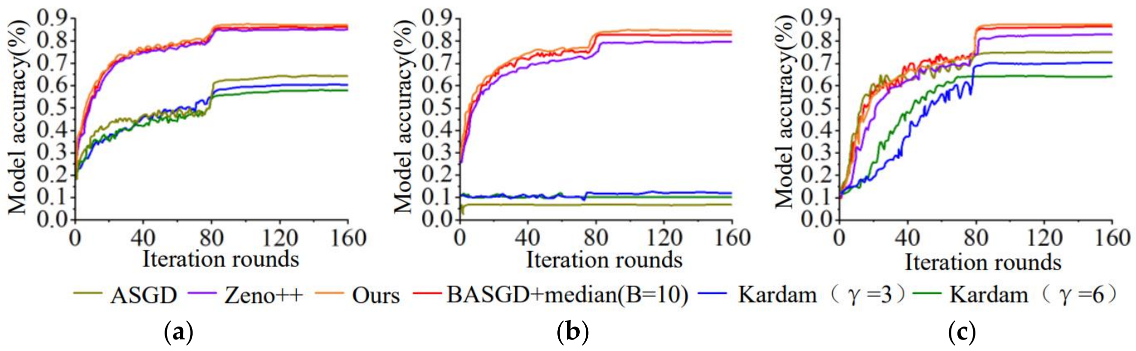


Figure 4. Model accuracy comparison on CIFAR-10 and $q = 10\%$. (**a**) Symbol flipping attack. (**b**) Noise attack. (**c**) Gaussian attack.

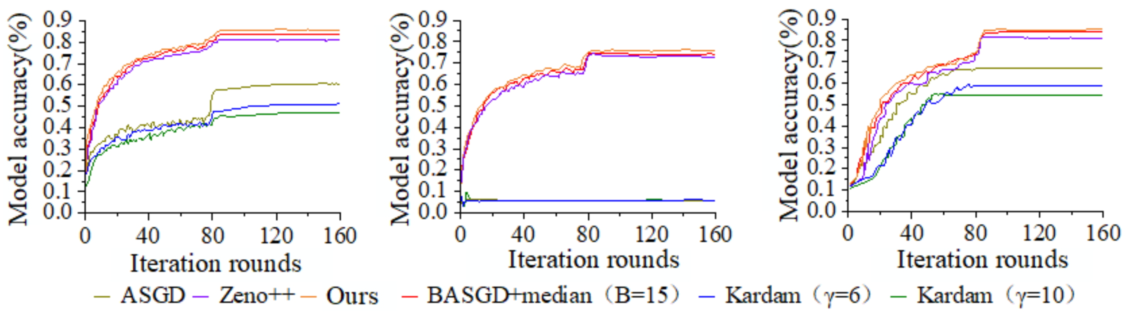


Figure 5. Model accuracy comparison on CIFAR-10 and $q = 20\%$. (**a**) Symbol flipping attack. (**b**) Noise attack. (**c**) Gaussian attack.

(2) Convergence speed comparison

In this section, we compare the convergence speed of different methods in the absence of Byzantine attackers. We fixed the training round to 100 rounds and observed the accuracy of each asynchronous robust aggregation algorithm by changing the proportion of stragglers to 20%, 60%, 80%, and 90%. We set the BASGD buffer to 10 and implemented it in combination with the Kardam and media algorithms.

It can be seen from Figure 6 that our method achieves the best performance for all proportions of stragglers in fixed training rounds. AGRA has a significant improvement in accuracy compared to Zeno++, BASGD, and Kardam, and its convergence speed is closest to that of the proposed method. But as the proportion of stragglers increases, the convergence speed slows down, because a higher proportion of stragglers increases the error and noise of random gradients, thereby slowing down the convergence speed.

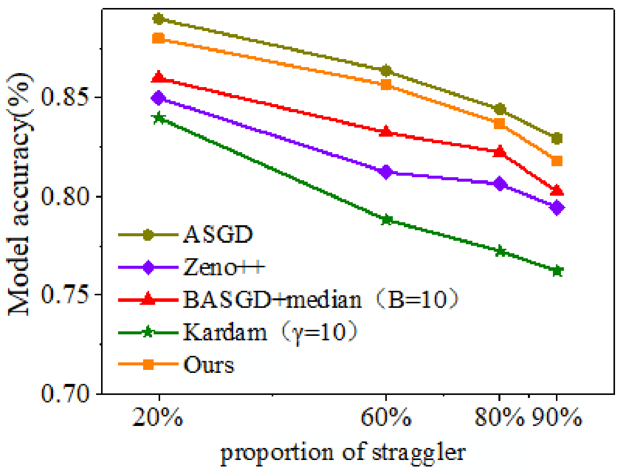


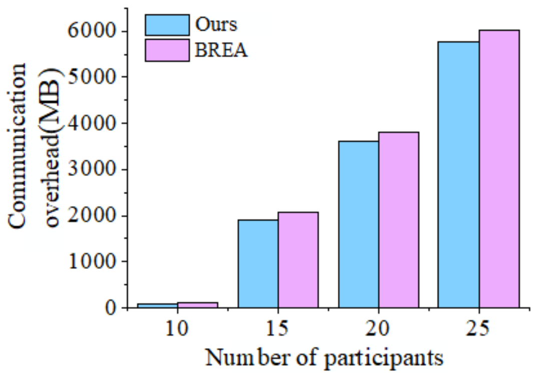
Figure 6. Model accuracy under different proportions of straggler on CIFAR-10.

In all the experiments above, when there are Byzantine attackers, our method converges faster than the gold standard ASGD. However, in the absence of Byzantine attackers, our method is slightly inferior to ASGD, but the algorithm in this chapter achieves a good balance between security and convergence speed.

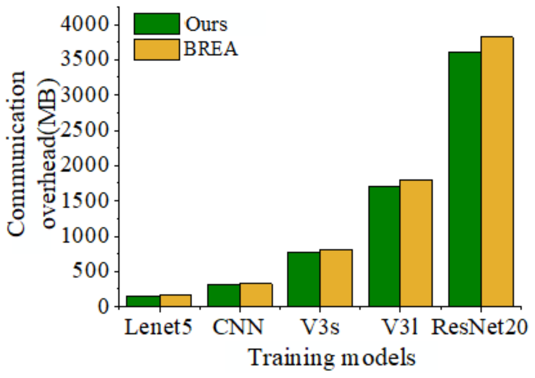
(3) Communication overhead comparison

In this section, we compare the communication overhead of our method with BREA from two aspects. On the one hand, the fixed training model is ResNet20, and the number of devices is set to 10, 15, 20, and 25. We observe the changes in communication overhead as the number of devices increases. On the other hand, we fix 20 devices and set the training models as ResNet20, MobileNet V3 large, MobileNet V3 small, CNN, and LeNet5.

It can be seen from Figure 7a that as the number of devices increases, the communication overhead of the method proposed in this paper is always smaller than that of BREA. This is because the verification phase of BREA requires transmission between devices several times, while the execution process of the method proposed in this paper only requires the transmission of encrypted random numbers once in the initial stage, with most of the communication overhead borne by the server, which is extremely friendly for IoV devices with limited communication capabilities and unstable connection states. It can be seen from Figure 7a that as the dimensions of the training model increase, the same results are also presented. The incremental speed of our method is slightly slower than that of BREA, but as the number of training rounds increases, the superiority of our method will become more significant because only encrypted random numbers are transmitted in the initial stage of training. The above simulations can all demonstrate that the method proposed in this paper is suitable for IoV scenarios with massive devices and complex task requirements.



(**a**)



(**b**)

Figure 7. Communication overheads of the proposed method and BREA on CIFAR-10. (**a**) Communication overhead of participants. (**b**) Communication overhead of training model.

7. Conclusions

This paper presents an asynchronous federated learning approach for the heterogeneous internet of vehicles (IoVs), enhancing privacy and mitigating delays and attacks. It introduces a unique grouping strategy, anomaly filtering, and truth estimation to counteract adversarial behavior, and employs additive homomorphic encryption to minimize communication rounds for better efficiency and security. An aggregation algorithm reduces the impact of stragglers and Byzantine attackers, while a new protocol upholds privacy and security against such attacks. Experimental results show our method outperforms the baseline by 41.6% and 13.3% against symbol flipping and Gaussian attacks, respectively, and is superior to Kardam and ASGD under noise attacks. The algorithm suits IoV scenarios for collaborative model learning, ensuring data privacy and system robustness against cyber threats. However, scalability may be challenged by the computational demands of encryption, and real-world IoV conditions could test the robustness further.

In future research, we aim to delve deeper into the complexities introduced by the data heterogeneity inherent in the internet of vehicle (IoV) ecosystems. As for the non-IID (independently and identically distributed) data distributions in IoV devices, we plan to tackle the challenges of heterogeneity in federated learning systems. Our goal is to design robust aggregation algorithms specifically tailored to manage this diversity in data while simultaneously enhancing privacy protection.

We will explore novel aggregation techniques that are resilient to the statistical discrepancies across the data collected from various IoV devices, ensuring that the federated model remains robust and accurate even when trained on highly heterogeneous data. This will likely involve the development of advanced machine learning models that can identify and adapt to the unique statistical properties of the data generated by different vehicles.

Author Contributions: Conceptualization, T.T.; methodology, Ning Jiang; validation, A.Z. and N.J.; formal analysis, A.Z.; investigation, A.Z.; writing—original draft preparation, A.Z.; writing—review and editing, N.J. and T.T. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by Chongqing Key projects for Technological Innovation and Application Development under grant CSTB2022TIAD-KPX0067 and National Natural Science Foundation of China under grant U20A20157, 61771082, 62271096, and 61871062.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest. Antong Zhou, Ning Jiang are employees of Mashang Consumer Finance Co., Ltd., Chongqing. The paper reflects the views of the scientists, and not the company.

References

1. Li, Y.; Tao, X.; Zhang, X.; Liu, J.; Xu, J. Privacy-preserved federated learning for autonomous driving. *IEEE Trans. Intell. Transp. Syst.* **2021**, *23*, 8423–8434. [CrossRef]
2. Mothukuri, V.; Khare, P.; Parizi, R.M.; Pouriyeh, S.; Dehghantanha, A.; Srivastava, G. Federated learning-based anomaly detection for IoV security attacks. *IEEE Internet Things J.* **2021**, *9*, 2545–2554. [CrossRef]
3. Tang, T.; Yin, Z.; Li, J.; Wang, H.; Wu, D.; Wang, R. End-to-End Distortion Modeling for Error-Resilient Screen Content Video Coding. *IEEE Trans. Multimedia.* **2023**. [CrossRef]
4. McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; Arcas, B.A. Communication-efficient learning of deep networks from decentralized data. In Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS), Fort Lauderdale, FL, USA, 20–22 April 2017; pp. 1273–1282.
5. Luo, Z.; Zhao, Y.; Xiong, M.; Wei, X.; Dai, H. A Self-Tuning LCC/LCC System Based on Switch-Controlled Capacitors for Constant-Power Wireless Electric Vehicle Charging. *IEEE Trans. Ind. Electron.* **2023**, *70*, 709–720. [CrossRef]
6. Luo, Z.; Wei, X. Analysis of Square and Circular Planar Spiral Coils in Wireless Power Transfer System for Electric Vehicles. *IEEE Trans. Ind. Electron.* **2018**, *65*, 331–341. [CrossRef]
7. Li, Z.; Gao, X.; Li, Q.; Guo, J.; Yang, B. Edge caching enhancement for industrial internet: A recommendation-aided approach. *IEEE Internet Things J.* **2022**, *9*, 16941–16952. [CrossRef]
8. Zhou, M.; Wei, X.; Jia, W.; Kwong, S. Joint Decision Tree and Visual Feature Rate Control Optimization for VVC UHD Coding. *IEEE Trans. Image Process.* **2022**, *32*, 219–234. [CrossRef] [PubMed]
9. Blanchard, P.; Mhamdi, E.M.E.; Guerraoui, R.; Stainer, J. Machine learning with adversaries: Byzantine tolerant gradient descent. In Proceedings of the 31st International Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017.
10. Yin, D.; Chen, Y.; Ramchandran, K.; Bartlett, P. Byzantine-robust distributed learning: Towards optimal statistical rates. In Proceedings of the International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; Volume 80, pp. 5650–5659.
11. Xu, C.; Qu, Y.; Xiang, Y.; Gao, L. Asynchronous Federated Learning on Heterogeneous Devices: A Survey. *Comput. Sci. Rev.* **2023**, *50*, 100595. [CrossRef]
12. Xie, C.; Koyejo, S.; Gupta, I. Zeno++: Robust fully asynchronous SGD. In Proceedings of the International Conference on Machine Learning, Virtual Event, 13–18 July 2020; Volume 119, pp. 10495–10503.
13. Californians for Consumer Privacy. California Privacy Rights Act. 2020. Available online: https://www.caprivacy.org/ (accessed on 28 December 2023).
14. Chik, W.B. The Singapore Personal Data Protection Act and an assessment of future trends in data privacy reform. *Comput. Law Secur. Rev.* **2013**, *29*, 554–575. [CrossRef]
15. G.D.P. Regulation. General Data Protection Regulation (GDPR). Intersoft Consulting. 2018, 1. Available online: https://gdpr-info.eu/ (accessed on 28 December 2023).
16. Conger, K.; Uber Settles Data Breach Investigation for 148 Million. The New York Times. Available online: https://www.nytimes.com/2018/09/26/technology/uber-data-breach.html (accessed on 26 September 2018).
17. Satariano, A.; Google Is Fined 57 Million under Europe's Data Privacy Law. The New York Times. Available online: https://www.nytimes.com/2019/01/21/technology/google-europe-gdpr-fine.html (accessed on 21 January 2019).
18. Kostrzewski, M.; Marczewska, M.; Uden, L. The Internet of Vehicles and Sustainability—Reflections on Environ-mental, Social, and Corporate Governance. *Energies* **2023**, *16*, 3208. [CrossRef]

19. Melis, L.; Song, C.; De Cristofaro, E.; Shmatikov, V. Exploiting unintended feature leakage in collaborative learning. In Proceedings of the IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 19–23 May 2019; pp. 691–706.
20. Zhu, L.; Han, S. Deep leakage from gradients. In *Federated Learning*; Springer: Cham, Switzerland, 2020; pp. 17–31.
21. Li, Z.; Yu, H.; Zhou, T.; Luo, L.; Sun, G. Byzantine resistant secure blockchained federated learning at the edge. *IEEE Network.* **2021**, *35*, 295–301. [CrossRef]
22. Zhao, B.; Sun, P.; Wang, T.; Jiang, K. Fedinv: Byzantine-robust federated learning by inversing local model updates. *Proc. AAAI Conf. Artif. Intell.* **2022**, *36*, 9171–9179. [CrossRef]
23. Li, S.; Ngai, E.; Voigt, T. Byzantine-robust aggregation in federated learning empowered industrial IoV. *IEEE Trans. Ind. Inform.* **2021**, *19*, 1165–1175. [CrossRef]
24. Damaskinos, G.; Mhamdi, E.M.E.I.; Guerraoui, R.; Patra, R.; Taziki, M. Asynchronous Byzantine machine learning (the case of SGD). In Proceedings of the International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; Volume 80, pp. 1145–1154.
25. Yang, Y.R.; Li, W.J. Basgd: Buffered asynchronous sgd for byzantine learning. In Proceedings of the International Conference on Machine Learning, Virtual, 18–24 July 2021; Volume 139, pp. 11751–11761.
26. 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.
27. Zhou, H.; Yang, G.; Dai, H.; Liu, G. PFLF: Privacy-preserving federated learning framework for edge computing. *IEEE Trans. Inf. Forensics Secur.* **2022**, *17*, 1905–1918. [CrossRef]
28. 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]
29. Li, Y.; Li, H.; Xu, G.; Huang, X.; Lu, R. Efficient Privacy-Preserving Federated Learning with Unreliable Users. *IEEE Internet Things J.* **2021**, *9*, 11590–11603. [CrossRef]
30. Ma, Z.; Ma, J.; Miao, Y.; Li, Y.; Zhao, B.; Yang, Y.; Liu, X. ShieldFL: Mitigating model poisoning attacks in privacy-preserving federated learning. *IEEE Trans. Inf. Forensics Secur.* **2022**, *17*, 1639–1654. [CrossRef]
31. So, J.; Güler, B.; Avestimehr, A.S. Byzantine-resilient secure federated learning. *IEEE J. Sel. Areas Commun.* **2020**, *39*, 2168–2181. [CrossRef]
32. Paillier, P. Public-key cryptosystems based on composite degree residuosity classes. In Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques, Prague, Czech Republic, 2–6 May 1999; Springer: Berlin, Heidelberg, 1999; pp. 223–238.
33. Gupta, S.; Zhang, W.; Wang, F. Model accuracy and runtime tradeoff in distributed deep learning: A systematic study. In Proceedings of the 2016 IEEE 16th International Conference on Data Mining (ICDM), Barcelona, Spain, 12–15 December 2016; IEEE: New York, NY, USA, 2016; pp. 171–180.
34. Xu, C.; Jia, Y.; Zhu, L.; Zhang, C.; Jin, G.; Sharif, K. TDFL: Truth Discovery Based Byzantine Robust Federated Learning. *IEEE Trans. Parallel Distrib. Syst.* **2022**, *33*, 4835–4848. [CrossRef]
35. Luping, W.; Wei, W.; Bo, L. CMFL: Mitigating communication overhead for federated learning. In Proceedings of the 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS), Dallas, TX, USA, 7–10 July 2019; IEEE: New York, NY, USA, 2019; pp. 954–964.
36. Krizhevsky, A.; Hinton, G. *Learning Multiple Layers of Features from Tiny Images*; Technical Report TR-2009; University of Toronto: Toronto, ON, Canada, 2009.
37. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.