electronics

MDPI

Article

A Privacy Robust Aggregation Method Based on Federated Learning in the IoT

Qingtie Li [1], Xuemei Wang [2] and Shougang Ren [2,*]

[1] Shanghai Lanchang Automation Technology Co., Ltd., Shanghai 201906, China; chairman@lclanchang.com
[2] College of Artificial Intelligence, Nanjing Agricultural University, Nanjing 210095, China; 2021819041@stu.njau.edu.cn
* Correspondence: rensg@njau.edu.cn

Abstract: Federated learning has been widely applied because it enables a large number of IoT devices to conduct collaborative training while maintaining private data localization. However, the security risks and threats faced by federated learning in IoT applications are becoming increasingly prominent. Except for direct data leakage, there is also a need to face threats that attackers interpret gradients and infer private information. This paper proposes a Privacy Robust Aggregation Based on Federated Learning (PBA), which can be applied to multiple server scenarios. PBA filters outliers by using the approximate Euclidean distance calculated from binary sequences and the 3σ criterion. Then, this paper provides correctness analysis and computational complexity analysis on the aggregation process of PBA. Moreover, the performance of PBA is evaluated concerning ensuring privacy and robustness in this paper. The results indicate that PBA can resist Byzantine attacks and a state-of-the-art privacy inference, which means that PBA can ensure privacy and robustness.

Keywords: federated learning; privacy protection; Byzantine-robust

check for updates

Citation: Li, Q.; Wang, X.; Ren, S. A Privacy Robust Aggregation Method Based on Federated Learning in the IoT. *Electronics* **2023**, *12*, 2951. https://doi.org/10.3390/electronics12132951

Academic Editors: Yongjun Ren and Hu Xiong

Received: 27 May 2023
Revised: 26 June 2023
Accepted: 30 June 2023
Published: 5 July 2023

1. Introduction

Internet of Things (IoT) applications can truly realize smart cities, which would require a massive number of IoT devices [1]. Owing to remarkable growth in IoT network size and data volume, these IoT devices can be made possible by centralized machine learning methods [2]. But as centralized machine learning techniques need to transfer the data to a centralized server for training, they are faced with the inherent issues of privacy leakage [3]. Moreover, centralized machine learning may not be practicable with large and dispersed data. Therefore, federated learning (FL) has been introduced to build intelligent and privacy-enhanced IoT systems [4–6].

FL is a distributed collaborative approach that coordinates the data training without sharing datasets [7], while FL itself has privacy concerns, it does not completely guarantee privacy [8–10]. Firstly, FL aims to train a model on multiple dispersed devices collaboratively by transferring local model updates, while only local update needs to be shared during the training process, there is also a high possibility of personal information leakage. This means that if sufficient measures are not taken, it will cause serious data privacy leakage risks. Secondly, the model inversion attack is a type of privacy attack that tries to infer the training data [11,12]. Deep Leakage from Gradients (DLG) [13] and Generative Adversarial Networks (GAN) [14] attacks are both types of model inversion attacks. DLG attack is a gradient-based attack method that can infer the input data of a model by accessing its gradient information. GAN attack is an attack method based on a generative adversarial network. It can generate data similar to the target model by training a generator network and then infer the input data of the model. Furthermore, due to the autonomy of workers in FL environments, any worker that participates in the process may be malicious and disrupt the security of FL [15]. These malicious workers can submit threatening local updates and malicious models, intentionally interfere with

the convergence of the global model, and affect the efficiency of FL. Above all, FL is still affected by privacy inference [13,16,17] and Byzantine attacks [18,19]. And these security risks and threats faced by FL in IoT applications are also increasingly prominent [12,20,21]. It is essential to study privacy protection and Byzantine-robust FL methods in IoT scenarios.

The contributions of this paper are summarized as follows:

1. We design an approximate Euclidean distance calculated by the sign matrix and data matrix, where the sign matrix and data matrix are composed of the sign and absolute value of the model update from each worker. It can decrease the computational complexity.
2. We propose a privacy robust aggregation method based on FL (PBA) in a multi-server scenario. PBA is implemented based on approximate Euclidean distance and adopts scientific 3σ criterion instead of sorting to screen outliers. We also analyze the correctness and computational complexity of PBA.
3. We conduct experiments and evaluate the performance of PBA concerning privacy security, Byzantine robustness, and time costs. The results indicate that PBA can ensure privacy and robustness with less time.

The rest of this paper is organized as follows: Section 2 summarizes the works related to ours while highlighting the differences; Section 3 introduces the threat model and some basic techniques; Section 4 designs a privacy robust aggregation rule based on approximate Euclidean distance and analyzes the correctness and computational complexity; Section 5 conducts experimental evaluation; Section 6 concludes the paper.

2. Related Work

Compared with traditional machine learning, FL already increases the level of privacy. It mitigates the transmission of sensitive data and prevents a third party from performing learning tasks on the unpermitted individual data [22]. However, the local updates/gradients uploaded by individuals, especially parts of personal data that are sensitive to specific features or values, may reveal sensitive information about users' data. Moreover, bad data may be uploaded because of device malfunction or malicious behavior.

Many studies have been conducted to solve the issues mentioned above. Currently, studies on privacy-preserving FL use either secure multi-party computation (MPC) or differential privacy (DP). By combining the locally trained classifiers, Pathak et al. [23] suggested a DP-based global classifier. A framework was proposed that adjusted the objective function during the training process to achieve DP [24]. Other methods based on encryption include homomorphic encryption (HE) and secure aggregation protocol [25]. A privacy-preserving deep learning (PPDL) algorithm was proposed [26], in which a number of distributed participants work together to train a deep learning model using local data. They established a trade-off between efficiency and security for the number of clients taking part in the training process. For Byzantine resilience, Krum [27] computed distance based on Euclidean norm between gradients to filter outliers. Different from Krum [27], coordinate-median and trimmed-mean [28] performed element-wise aggregation rules. After that, Yang et al. [29,30] presented ByRDiE and BRIDGE based on trimmed-mean from centralized to decentralized systems. At every epoch for every regular client, BRIDGE [30] performs the exchange and updates its local model's coordinates, whereas ByRDiE [29] executes multiple scalar exchanges and updates. Essentially combining Krum [27] and a variant of trimmed-mean [28], Bulyan [31] was proposed to reduce the leeway of Byzantine clients. To provide robustness against a poisoning attack that lowers global accuracy, a lightweight federated multi-task learning framework [32] was proposed. Anomaly detection [33], principal component analysis [34], and so on were applied to identify malicious clients. Instead of considering the relationship between privacy and robustness, these efforts concentrated on privacy protection or Byzantine resilience in FL.

To achieve privacy-preserving and Byzantine-resilience, So et al. [35] proposed a FL algorithm based on secure aggregation. Secret sharing (SS) is used, and any two parties need to negotiate a random number to ensure the security of the information. However, this may require a large number of rounds of interaction, leading to higher communication

complexity. Guo et al. [36] proposed an Uniform Byzantine-resilient Aggregation Rule (UBAR) to defeat an arbitrary number of Byzantine nodes. The main idea is to use training samples to test the performance of parameters that are benign clients. This method can converge quickly, but the balance between privacy protection and Byzantine robustness needs to be struck. FoolsGold [37] calculated the cosine similarity between the historical gradients of the clients to mitigate sybils. However, this scheme directly analyzed and calculated the gradients in plain text, which resulted in privacy leakage.

3. Preliminary

In this section, we introduce some techniques involved in this paper. The symbols and notations used in the remainder of this paper are given in Table 1.

Table 1. Notations.

Notation	Description
C	the number of workers
S	the number of servers
f	the number of malicious workers
T	the number of epochs
g	model update (gradient) [1]
x	global model
$g[k]$	the k-th value of the gradient
r	random vector
$\mathcal{S}$	sign matrix
$\mathcal{D}$	data matrix
μ	mean value
σ	standard deviation
η	learning rate

[1] "Global model" and "gradient" are used interchangeably.

3.1. Federated Learning

Data are the cornerstone of model training in the field of artificial intelligence. However, data is frequently present in the form of data islands. Data processing in a centralized way is the direct remedy for data islands, but data leakage may occur during the collection and processing stages [38]. To this end, FL is developed and received widespread attention.

FL seeks to train a global collectively on numerous datasets distributed on individual devices without explicitly transferring data samples [39,40]. In IoT networks, several IoT devices can act as workers who interact with a server to train neural networks. The server first initializes a global model. Each worker downloads the most recent model, computes its own model update, and then sends the computed update to the server. Subsequently, the server aggregates all local updates and updates the global model [5]. Local training, aggregation, update, and other steps mentioned above are repeated until the termination condition is met. The preset training epochs or global model accuracy can be considered as termination conditions.

Although the development of FL solves the problem of data islands and reduces the risk of data leakage, many threats and challenges still need to be addressed urgently. The most core problems include the weakness of communication efficiency [41–43], the defects of privacy security [44–46], the lack of incentive mechanisms [47,48] and so on.

3.2. Privacy Protection Technology

Privacy computing is a set of technologies that can achieve the goal of "usable and invisible" data. Homomorphic encryption (HE), secret sharing (SS), and differential privacy (DP) are presented in this section.

3.2.1. Homomorphic Encryption

HE [49] is a type of encryption that preserves the functionality and format of the encrypted data while allowing a third party to perform some compute operations on the encrypted data. It is usually defined as:

If an encryption scheme satisfies the following equation, then the scheme is called homomorphic with respect to the operation $\star$:

$$E(m_1) \star E(m_2) = E(m_1 \star m_2), \forall m_1, m_2 \in M \tag{1}$$

where E represents the encryption algorithm, and M represents the collection of all information [50].

HE can be used to withstand various security threats, such as membership inference attacks [51], DLG attack [13], chosen plain text attacks [52], etc. Therefore, it has good application prospects in the field of information security.

3.2.2. Secret Sharing

SS is a technique that divides a secret into multiple parts and distributes them to different participants. The secret can only be reconstructed when specific conditions are met [53,54].

Taking the Shamir [55] as an example, it is a classic (t, n) threshold SS method. A secret owner and a group of participants are involved in this method. The owner splits a secret into n shards and transfers them to n participants. The secret can be recovered only by simultaneously obtaining t shards. The specific implementation process of Shamir [55] is as follows:

1. Encryption
   Assuming there is a secret S, take any $t-1$ random numbers and construct a polynomial $f(x) = a_0 + a_1 \cdot x + a_2 \cdot x^2 + ... + a_{(t-1)} \cdot x^{(t-1)}$, where $a_0 = S$. It should be noted that all operations are performed in a finite field. Then take any n numbers, substitute $x_1, x_2, ..., x_n$ into the polynomial to obtain $f(x_1), f(x_2), ..., f(x_n)$4. Send $(x_i, f(x_i))$ to the participants.
2. Decryption
   Polynomial coefficient calculation using secret shards held by any t participants:

$$\begin{cases} a_0 + a_1 \cdot x_1 + a_2 \cdot (x_1)^2 + \cdots + a_{t-1} \cdot (x_1)^{t-1} = y_1, \\ a_0 + a_1 \cdot x_2 + a_2 \cdot (x_2)^2 + \cdots + a_{t-1} \cdot (x_2)^{t-1} = y_2, \\ \vdots \\ a_0 + a_1 \cdot x_3 + a_2 \cdot (x_t)^2 + \cdots + a_{t-1} \cdot (x_t)^{t-1} = y_t. \end{cases} \tag{2}$$

   After obtaining the coefficients, substitute them into the polynomial, take $x = 0$, and you can obtain the secret $S = a_0$.

3.2.3. Differential Privacy

DP [56] protects privacy by merging client data with added noise. This noise makes it difficult for attackers to determine which data belongs to a specific client, thereby increasing the privacy of the data. Local differential privacy (LDP) and global differential privacy (GDP) are variants of DP [57], which are used to protect privacy.

For LDP [58], each worker's data is added with noise. On the contrary, GDP [59] needs to send all workers' data to the central server for processing. This makes GDP more suitable for situations where the entire dataset needs to be analyzed. Obviously, the difference between LDP and GDP is that LDP is a method to achieve DP on a single dataset, while GDP is a method to add noise on multiple datasets.

3.3. Outlier Detection Method

Outliers, which may be caused by human errors, mechanical failures, and changes in system behavior, or natural deviations in the environment, can be identified through outlier detection [60], a method that discovers significant deviations from normal or mean values in given data.

3.3.1. 3σ Criterion

In 3σ criterion [61], outliers are defined as a collection of measurements that deviate from the mean by more than three times the standard deviation. Under normal distribution, the chance of values appearing beyond 3σ from the mean is $p(|x-\mu| > 3\sigma) \leq 0.003$, which is an extremely unusual event. If the data is not normally distributed, $a \cdot \sigma$ can be used to characterize it. Therefore, Curtis et al. [62] proposed the z-score method, which measures the distance of a certain raw score x from the mean in standard deviation units. That is,

$$z = \frac{x-\mu}{\sigma} \tag{3}$$

z-score typically considers data points more than three times the standard deviation from the mean as outliers. In other words, data points with z-score $> a$ are considered outliers, where a can be set according to specific circumstances.

3.3.2. DBSCAN

Density-based Spatial Clustering of Applications with Noise (DBSCAN) [63] is a density-based spatial clustering algorithm. This technique finds clusters of any shape in a noisy spatial database and divides areas with sufficient density into one class. The core idea of DBSCAN is to start from a certain core point and continuously expand to a region where the density can reach so as to obtain a maximized region containing core points and boundary points. Any two points in the region are connected in density. Moreover, some studies use DBSCAN to perform outlier filters. The disadvantage is that the application of DBSCAN results in a longer time required for model training.

4. Privacy Robust Aggregation Based on Federated Learning

In this section, we present the system structure and threat model. Moreover, we design an approximate Euclidean distance calculated by the sign matrix and data matrix and propose a Privacy Robust Aggregation Based on Federated Learning (PBA) aggregation rule.

4.1. System Structure and Threat Model

Figure 1 depicts our system model. There are C customers in the overall structure, of which f is Byzantine. The gradients sent by these Byzantine clients may interfere with the convergence and precision of the model. These clients interact with S $(S \geq 2)$ servers. These servers are required to carry out the identification of outliers and global model updates.

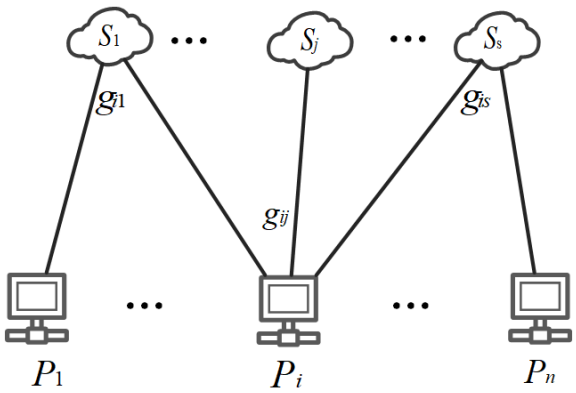

Figure 1. The overall framework of PBA.

On the server side, the threat model is classed as honest-but-curious (semi-honest). This means that servers follow the established protocol without deviating but could try to extract private information from the data transmitted with them. So, if these servers obtain raw gradients, they might employ established methods to extract sensitive client data. Only a fraction of clients may be Byzantine. They may intentionally produce false data in an effort to undermine the training.

4.2. Algorithm Overview

The privacy-robust aggregation rule PBA proposed in this paper is implemented based on S servers, where $S \geq 2$. Suppose that there are n participants in the system, out of which f are Byzantine. The servers in the system are honest and curious (semi-honest) but do not collide with each other. This implies that they follow algorithmic processes, do not disclose each other's information, but may attempt to access user privacy and restore private data. Similarly, n participants are also honest and curious. However, f malicious participants might upload random data, leading to the deviation of the global model from its standard training process. Figure 1 illustrates the overall framework of PBA, where g_ij represents the gradient information sent by participant C_i to the server S_j.

During the model initialization stage, S servers negotiate to create $\frac{S(S-1)}{2}$ random vectors. These vectors need to satisfy the following equation:

$$\sum_{i=1}^{\frac{S(S-1)}{2}} (S-i+1) \cdot r_i = c \tag{4}$$

where r_i represents the i-th vector generated by servers and c is a constant. Then, each client downloads the most recent global model and computes local updates. Before uploading the updates to the servers, the following operations need to be performed:

1. Differential Privacy and Denoise
   To prevent the data leakage in servers, client C_i creates private noise and adds it to the local gradient g_i, i.e., $\tilde{g}_i \leftarrow g_i + \mathcal{N}(0, \triangle^2 \cdot \sigma^2)$. But noises added to the gradients have an impact on model convergence and accuracy. Hence, It is necessary to denoise the gradients after adding noises. Kolmogorov–Smirnov (KS) distance is used as a measure of denoising [64]. C_i applies it to denoise the gradient. That is, $\hat{g}_i \leftarrow KS(\tilde{g}_i, \mathcal{N})$, where $KS()$ represents the KS measure.
   The above operations can not only preserve the characteristics of DP to reduce the risk of data leakage but also eliminate the impact of noise on the model convergence and accuracy [64].
2. Gradient Encoding
   Each client needs to perform some mask operations to provide further protection. The specific masking process is shown below:

$$\hat{g}_{ij} = \hat{g}_i + \sum_{p:p \leq j} r_p + \sum_{p:p \geq j} r_p \tag{5}$$

   where $\hat{g}_{ij}$ represents the encoded gradient that C_i sends to the server S_j.

Above all, C_i generates the gradient $\hat{g}_{ij}$ based on its own local update g_i, which undergoes DP, denoise, and encoding operations and sends to the server S_j. Then, when S_j receives the gradient information, it needs to calculate the approximate Euclidean distance between two gradients, perform outlier detection through 3σ criterion, and combine gradients.

After receiving C encoded gradient information, S_j calculates the approximate Euclidean distance. And the approximate distance between C_i and C_j is temporarily recorded as $AED(i, j)$. Then, S_j detects outliers in the approximate distances based on 3σ criterion. In the application of PBA, outliers can be defined as values with deviations from the mean

exceeding a times, where a is not necessarily three. Thirdly, S_j calculates the sum of all gradients based on the filtration results, i.e.,

$$g_{S_j} = \sum_{g_i \in N_{S_i}} g_i + n \cdot \left(\sum_{p:p \leq j} r_p - \sum_{q:q>j} r_q \right) \tag{6}$$

The computation result g_{S_j} is sent to other servers while S_j receives corresponding results from them. S_j combines gradients as follows:

$$g = \frac{1}{S} \sum g_{S_j} - (n - f) \cdot c \tag{7}$$

Finally, each server completes this epoch by updating the global model. During the initialization phase, S servers negotiate and generate random vectors r_i. Then, The process of PBA is shown in Algorithm 1, where $f_{AED}(\cdot)$ represents the function that calculates the approximate Euclidean distance.

Algorithm 1 PBA.

Input:
The global model in the t-th round, x_t;
Output:
The gradient aggregated in the t-th round, g;
for client $C_i, i = 1, 2, ..., C$: **do**
  randomly sample from the local dataset for local training $g_i = \bigtriangledown f(x_t, \xi_i)$;
  add private noise $\tilde{g}_i \leftarrow g_i + \mathcal{N}(0, \triangle^2 \sigma^2)$;
  denoise $\hat{g}_i \leftarrow KS(\tilde{g}_i, \mathcal{N})$;
  encode $\hat{g}_{ij} = \hat{g}_i + \sum_{p:p \leq j} r_p + \sum_{p:p>j} r_p$;
  send $\hat{g}_{ij}$ to the server S_j;
end for
for server $S_j, j = 1, 2, ..., S$: **do**
  calculate approximate Euclidean distance $AED(i, j) = f_{AED}(\hat{g}_{pj}, \hat{g}_{qj})$;
  filter out outliers based on the 3σ criterion, denoted as N_{S_i};
  calculate the sum of gradients $g_{S_i} = \sum_{g_i \in N_{S_j}} + n \cdot (\sum_{p:p \leq i} r_p + \sum_{p:p>i} r_p)$;
  send the g_{S_j} to other servers and receive gradients from them;
  compute the average $g = \frac{1}{S} \cdot \sum g_{S_j} - (n - f) \cdot c$;
end for
return g

4.3. Approximate Euclidean Distance

Due to the characteristics of data dispersion and multi-device collaborative training, Training Time Attack (TTA), including Model Poisoning Attack and Data Poisoning Attack, is a significant threat to the security of FL system [65]. Studies explored robust aggregation techniques, such as Krum [27] and Median [28], to handle devices that send corrupted updates to the server. Wang et al. [66] proposed that the cosine value between any two gradients is only related to their direction, not magnitude. Therefore, they replace the gradient with its sign rather than specific values, represented as a binary sequence, to highlight the direction differences between any two gradients. However, this method ignores the impact of gradient sizes on model convergence. In distance-based robust aggregation methods, Euclidean distance is the most typical distance measure. It is computed as follows:

$$distance(i, j) = \sqrt{\sum (g_i[k] - g_j[k])^2} \tag{8}$$

where $g_i[k]$ and $g_j[k]$ denote the k-th bit of local training gradients g_i and g_j, respectively. However, distance-based robust aggregation methods could incur significant computational overhead with numerous clients in the system [9,67].

A method which transfers only gradient signs and computes cosine similarity through the binary sequence is proposed, which greatly reduces the communication overhead of FL system [66]. Based on this method, an approximate Euclidean distance based on binary sequences is designed. Let g_i and g_j denote the gradients of C_i and C_j, respectively, and g denote their average gradient. The approximate distance between them is computed as follows:

1. Servers form sign matrices and data matrices by taking the sign and absolute value of g_i and g_j, respectively. The sign matrices are denoted as $\mathcal{S}_i$ and $\mathcal{S}_j$, while the data matrices are denoted as $\mathcal{D}_i$ and $\mathcal{D}_j$.
2. An XOR operation between the two sign matrices is performed, i.e., $sign(i,j) = \mathcal{S}_i \bigoplus \mathcal{S}_i$. At the same time, servers need to subtract the two data matrices element-wise, take the absolute value, and compare it to g. Specifically,

$$data(i,j) = \begin{cases} data(i,j)[k] = 0, & data(i,j)[k] < g[k]; \\ data(i,j)[k] = 1, & data(i,j)[k] \geq g[k]; \end{cases} \tag{9}$$

   where $data(i,j)[k] = |\mathcal{D}_i[k] - \mathcal{D}_j[k]|$.
3. $sign(i,j)$ and $data(i,j)$ are combined, resulting in four different binary sequences: 00, 01, 10, and 11, which are represented as 0, 1, 2, and 3 in decimal form to reflect the degree of maliciousness of the gradients under different scenarios. Specifically,

$$AED(i,j) = \begin{cases} 0, & (sign, data) = 00 \\ 1, & (sign, data) = 01 \\ 2, & (sign, data) = 10 \\ 3, & (sign, data) = 11 \end{cases} \tag{10}$$

The four binary sequences mentioned above represent four scenarios: same-sign with a small value difference, same-sign with a large value difference, opposite-sign with a small value difference, and opposite-sign with a large value difference.

4.4. Theoretical Analysis

4.4.1. Correctness Analysis

PBA first performs DP for local updates and then reduces the noise according to the added noise in DP. In addition, due to the post-processing property of DP, the execution of denoising does not affect the differential property and can still maintain the model accuracy losslessly [64,66]. Therefore, privacy security can still be guaranteed. We provide the correctness analysis and proof for aggregation based on gradient encoding.

Firstly, the correctness of distance computed on encoded gradients is analyzed. For S_i, the gradient received from clients can be represented as $\hat{g}_{xi} = \hat{g}_x + \sum_{p:p\leq i} r_p + \sum_{p:p>i} r_p, x \in [n]$. Therefore, the distance of the gradients between any two clients C_x and C_y can be expressed as

$$\begin{aligned} distance(i,j) &= (\hat{g}_{xi} - \hat{g}_{yi})^2 \\ &= [(\hat{g}_x + \sum_{p:p\leq i} r_p + \sum_{p:p>i} r_p) - (\hat{g}_y + \sum_{p:p\leq i} r_p + \sum_{p:p>i} r_p)]^2 \\ &= (\hat{g}_x - \hat{g}_y)^2 \end{aligned} \tag{11}$$

It can be seen that the result computed on the server S_i is consistent with the result whose gradients are not encoded.

Then, a correctness proof of aggregation is provided. Suppose among n clients, f malicious clients are selected. The gradient sum calculated by server S_i is represented as $g_{S_j} = \sum_{g_i \in N_{S_i}} g_i + n \cdot (\sum_{p:p \le j} r_p - \sum_{q:q>j} r_q)$. Then,

$$\begin{aligned} \sum g_{S_i} &= \sum [\sum_{g_i \in N_{S_i}} g_i + n \cdot (\sum_{p:p \le j} r_p - \sum_{q:q>j} r_q)] \\ &= S \cdot \sum_{g_i \in N} g_i + (n-f) \cdot [S \cdot r_1 + \cdots + (S-i+1) \cdot r_i + \cdots + r_S] \\ &= S \cdot [\sum_{g_i \in N} g_i + (n-f) \cdot c] \end{aligned} \tag{12}$$

Thus, the gradient for the model update can be computed as follows:

$$g = \frac{1}{S} \cdot \sum g_{S_j} - (n-f) \cdot c = \sum_{g_i \in N} g_i \tag{13}$$

where N refers to the set of potential benign gradients.

4.4.2. Complexity Analysis

PBA is based on approximate Euclidean distance, which can reduce computational overheads. Hence, we analyze the computational complexity of approximate Euclidean distance and compare it with Euclidean distance.

Proposition 1. *Let n represent the dimension of a model update, k represent the bits of each dimension. Suppose that the time required to perform a one-bit XOR is unit time, the time required for addition and subtraction is t times the unit time, and the time required for multiplication and division is t times the unit time. Then, the computational complexity of approximate Euclidean distance is*

$$O = (2 \cdot k + 1) \cdot n \tag{14}$$

Proof. For approximate Euclidean distance, it needs to perform an XOR operation between two sign matrices, which requires n times unit time. Moreover, subtraction is performed between data matrices, and the required time is $n \cdot k$ times unit time. Finally, a comparison operation spends $n \cdot k$ times unit time. Hence, the time spent computing an approximate Euclidean distance is $O = (2 \cdot k + 1) \cdot n$. □

For Euclidean distance, the calculation method is shown in Formula (8), including subtraction, multiplication, addition, and square root operations. From this, it can be concluded that the computational complexity of Euclidean distance is $O = 2 \cdot (t+2) \cdot n \cdot k$. Then, after comparison, the computational complexity of approximate Euclidean distance is about $\frac{1}{(t+1) \cdot k}$ times that of Euclidean distance.

5. Performance Evaluation

5.1. Experiment Setup

All trials are conducted on a computer equipped with an AMD Ryzen 7 5800H, Radeon Graphics 3.20 GHz, and NVIDIA GeForce GTX 1650. And PyTorch [68] is used for training in Python. In addition, a task for object recognition is developed that involves n clients working together to train a convolutional neural network (CNN) with two convolutional layers and two fully connected layers.

Assuming there are S samples and L categories in the dataset, two methods are studied for dividing the dataset among the clients:

1. Independent Identically Distribution (IID)
   In this method, data is shuffled and split into $\frac{S}{C}$ samples per client.

2. Non-Independent Identically Distribution (Non-IID)
   In this method, the data set is sorted and divided into C parts of size $\frac{S}{C}$, assigning one part to each client. Non-IID data will be used to study the privacy security of PBA in order to reflect the leakage of specific sensitive information from a client.
   In the experiment, the MNIST [69] image dataset is used to train a CNN model, which is a handwritten digit (0–9) dataset consisting of 60,000 training samples and 10,000 test samples, with each sample being a 28×28 grayscale image

Moreover, in order to evaluate the Byzantine robustness, several Byzantine attacks are considered in the experiment:

1. Gaussian Attack (GA)
   This is an aimless poisoning attack aimed at reducing the accuracy of the model. Specifically, malicious clients can randomly sample from the Gaussian distribution and upload it as local update parameters, namely:

$$\hat{g}[d] = Gaussian(\mu, \sigma^2) \tag{15}$$

   Among them, μ represents the mathematical expectation of a Gaussian distribution, and σ^2 represents its variance.
2. Label Flipping Attack (LFA)
   This is a targeted poisoning attack. Malicious clients flip the labels of local data to generate error gradients [66], especially flipping the labels of each sample from x to $L-1-x, x \in (0, \ldots, L-1)$.

Furthermore, some baselines are compared with PBA, such as Krum [27], trimmed-mean [28] and UBAR [36]. And following performance metrics will be considered:

1. Global Accuracy: Test the accuracy of the trained model on the validation set;
2. Loss Value: Test the cross-entropy of the training model on the validation machine;
3. Reconstructed Images: Image reconstruction under DLG attack.

A total of $C = 20$ clients are set up, with f Byzantine clients. We compare the global accuracy and loss of the model under different numbers of malicious clients. When evaluating the security of PBA, the DLG attack will be conducted to see if user-sensitive information in images can be reconstructed.

5.2. Performance Analysis

Through a series of experiments, a comparison and analysis of the security and robustness of PBA are conducted. The model performance of PBA is evaluated concerning ensuring security and robustness.

5.2.1. Security Evaluation

To evaluate the security of PBA, the DLG attack is implemented, and LeNet5 is used as the generator and model training network. Moreover, we take the MNIST dataset sample "7" as an example to observe whether the reconstructed image can be recognized. In this part, we simulate malicious attackers intercepting the model updates of the parties participating in the training and attempting to recover sensitive information. We take the FL method without any protection as the baseline and compare the reconstructed images of these two methods. Each attack result displays the reconstruction results of every ten rounds within 500 rounds.

Firstly, the DLG attack is conducted on the baseline, which is not perturbed. Although there is still some noise in the data sample in the 10th round, as shown in Figure 2a, it could already be recognized by the human eye. Hence, it can be indicated that information leakage occurs in this method. Subsequently, we conduct the DLG attack on PBA. When the gradient is perturbed in PBA, the DLG attack cannot reconstruct the user's sensitive image, as shown in Figure 2b.

Moreover, the DLG attack is conducted on different degrees of perturbation, and the attack results are all unable to reconstruct the image. That is, the attack failed. The experimental results show that PBA is effective in resisting DLG attacks and can achieve the goal of protecting user privacy.

Figure 2. Reconstructed images under the DLG. (**a**) Baseline. (**b**) PBA.

5.2.2. Robustness Evaluation

In the evaluation of robustness, we take the FL method without any protection as a baseline method to compare with PBA, Krum [27], trimmed-mean [28], and UBAR [36]. We, respectively, set Byzantine clients accounting for 0%, 10%, and 30% of the total, comparing, and analyzing global accuracy changes under Gaussian attacks and label flipping attacks.

To begin with, the performance without malicious clients and attacks is tested and compared with the presence of Byzantine clients. The results as shown in Figure 3. Compared with the baseline, the accuracy of PBA is close to other methods, about 88%. In the same way, the loss of PBA, about 0.4, is lower or similar to that of other methods. Moreover, the convergence speed of PBA is almost the same as the baseline.

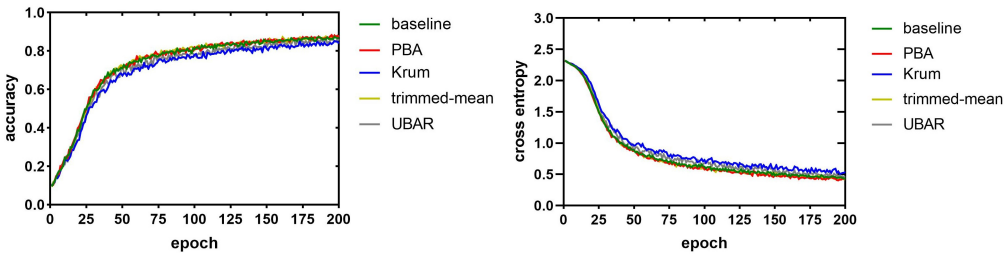

Figure 3. Convergence performance without malicious clients and attacks. (**a**) Global accuracy. (**b**) Cross entropy.

After that, the performance of PBA is compared with that of Krum [27], trimmed-mean [28], and UBAR [36] with the different number of Byzantine clients. Faced with the same attack, the degree to which model performance is affected by Byzantine clients is related to the number of malicious clients in the system.

We take GA to simulate a fraction of clients uploading bad data. From Figure 4, the model accuracy of all methods with $f = 6$ Byzantine clients in the system is lower than that with $f = 2$. When there are few malicious clients, the global accuracy of PBA is better than Krum [27] and UBAR [36], almost equal to the accuracy without Byzantine clients. But when malicious clients account for 30%, the performance of PBA and trimmed-mean [28] decreases significantly. On the contrary, the performance of UBAR [36] and Krum [27] can maintain more stability. Moreover, we conduct LFA to evaluate the model performance. As shown in Figure 5, although LFA is more severe than GA, the model performance of all methods is almost not affected with few Byzantine clients in the system, the accuracy of about 85%. However, if there are 60% malicious clients in the system, the performance

of methods, except UBAR [36], is reduced remarkably. Krum [27] and trimmed-mean [28] can not achieve convergence. This means Byzantine clients reach the goal of disrupting the model. Although PBA and UBAR [36] can achieve convergence, their accuracy is influenced significantly. Hence, PBA is unsuitable for the system with many Byzantine clients.

Based on the above experiments, the performance of PBA is close to the baseline without Byzantine clients and attacks. However, if there are many malicious clients in the system and the attacks are strong, the performance of PBA is greatly compromised. Although PBA did not always perform well in experiments, it could reduce the time cost compared to other methods.

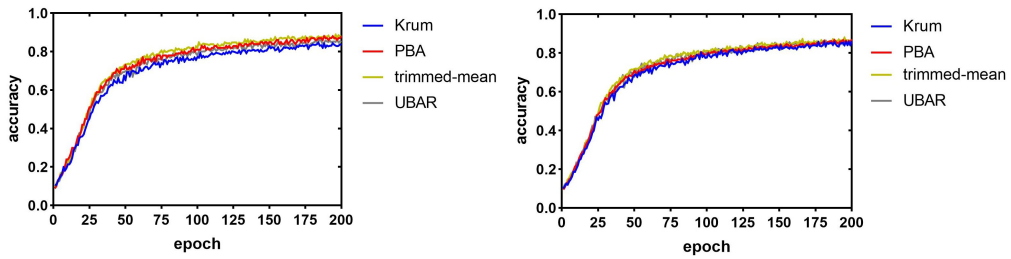

Figure 4. Global accuracy under GA with different numbers of malicious clients. (**a**) $f = 2$. (**b**) $f = 6$.

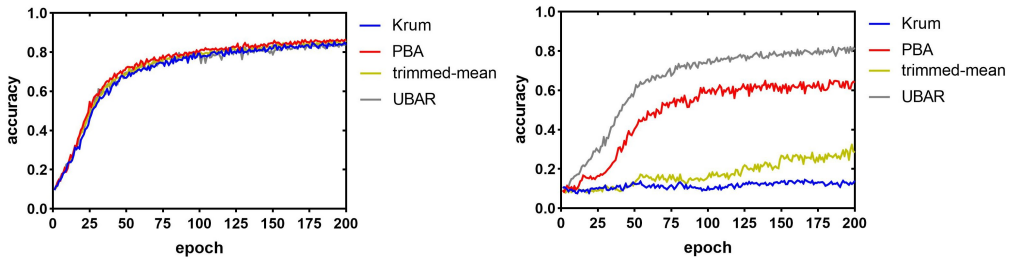

Figure 5. Global accuracy under LFA with different numbers of malicious clients. (**a**) $f = 2$. (**b**) $f = 6$.

5.2.3. Cost Evaluation

Wang et al. [66] proposed a method called Brief that calculates cosine similarity using binary operations and then implements B2A to calculate the Euclidean distance further. Similarly, PBA uses binary sequences to calculate the Euclidean distance, somewhat reducing the time cost. Hence, we measure the time spent by different methods in an epoch. And the presentation is the average time required for five epochs. The results are shown in Table 2.

Because of B2A and clustering used in Brief [66], it needs a long time to finish a training epoch, about 57 s. Moreover, to achieve high accuracy, UBAR [36] reuses training samples to test the performance, which takes some time. The time spent by other distance-based methods is not significantly different. For PBA, it is based on approximate Euclidean distance and 3σ criterion. PBA, computing approximate Euclidean distance by binary sequences, has lower complexity than other distance-based methods. Compared with the clustering method in Brief [66], the 3σ criterion with lower complexity is scientific.

Table 2. Running time of different methods.

Method	Time
Krum [27]	10.640 s
trimmed-mean [28]	17.588 s
UBAR [36]	23.030 s
Brief [66]	56.659 s
PBA	9.391 s

6. Conclusions

In order to solve the issue that previous federated learning methods cannot quickly execute outlier detection, this paper proposes a method to estimate the Euclidean distance using binary sequences that reflects the differences between gradients. Based on this approach, PBA, a 3σ criterion-based federated learning security robust method, is proposed. Moreover, this paper analyzes the correctness and complexity of PBA and evaluates the performance of PBA on open datasets. The results show that its privacy security can be guaranteed and Byzantine robustness can be achieved in some conditions. PBA offers good convergence performance and somewhat lowers the computational cost as compared to other approaches. However, faced with severe attacks or many Byzantine clients in the system, PBA can maintain stable performance.

We may try to deploy this method to multi-server scenarios that require a shorter time to train the FL model. In addition, in the future, we hope to further study how to better integrate FL with the IoT, which can protect privacy, achieve Byzantine resilience, and be more lightweight.

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

Funding: This research was funded by Research on Key Technologies for Greenhouse Gas Emission Reduction and Low Carbon Farming in Poultry Farming (BE2022310).

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable for this article.

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

References

1. Khan, L.U.; Yaqoob, I.; Imran, M.; Han, Z.; Hong, C.S. 6G wireless systems: A vision, architectural elements, and future directions. *IEEE Access* **2020**, *8*, 147029–147044. [CrossRef]
2. Qi, K.; Yang, C. Popularity prediction with federated learning for proactive caching at wireless edge. In Proceedings of the 2020 IEEE Wireless Communications and Networking Conference (WCNC), Seoul, Republic of Korea, 25–28 May 2020; IEEE: Toulouse, France , 2020; pp. 1–6.
3. Khan, L.U.; Saad, W.; Han, Z.; Hossain, E.; Hong, C.S. Federated learning for internet of things: Recent advances, taxonomy, and open challenges. *IEEE Commun. Surv. Tutor.* **2021**, *23*, 1759–1799. [CrossRef]
4. 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 Artificial Intelligence and Statistics, PMLR, Lauderdale, FL, USA, 20–22 April 2017; pp. 1273–1282.
5. Nguyen, D.C.; Ding, M.; Pathirana, P.N.; Seneviratne, A.; Li, J.; Poor, H.V. Federated learning for internet of things: A comprehensive survey. *IEEE Commun. Surv. Tutor.* **2021**, *23*, 1622–1658. [CrossRef]
6. Liu, Z.; Guo, J.; Yang, W.; Fan, J.; Lam, K.Y.; Zhao, J. Privacy-preserving aggregation in federated learning: A survey. In *IEEE Trans. Big Data*; IEEE: Toulouse, France, 2022.
7. Konečnỳ, J.; McMahan, H.B.; Yu, F.X.; Richtárik, P.; Suresh, A.T.; Bacon, D. Federated learning: Strategies for improving communication efficiency. *arXiv* **2016**, arXiv:1610.05492.
8. Kairouz, P.; McMahan, H.B.; Avent, B.; Bellet, A.; Bennis, M.; Bhagoji, A.N.; Bonawitz, K.; Charles, Z.; Cormode, G.; Cummings, R.; et al. Advances and open problems in federated learning. *Found. Trends Mach. Learn.* **2021**, *14*, 1–210. [CrossRef]
9. Hao, M.; Li, H.; Xu, G.; Chen, H.; Zhang, T. Efficient, Private and Robust Federated Learning. In *Annual Computer Security Applications Conference*; ACM: New York, NY, USA, 2021. [CrossRef]
10. Ma, X.; Gu, L. Research and Application of Generative-Adversarial-Network Attacks Defense Method Based on Federated Learning. *Electronics* **2023**, *12*, 975. [CrossRef]
11. Wang, K.C.; Fu, Y.; Li, K.; Khisti, A.; Zemel, R.; Makhzani, A. Variational Model Inversion Attacks. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 9706–9719. [CrossRef]
12. Zhao, Y.; Zhao, J.; Jiang, L.; Tan, R.; Niyato, D.; Li, Z.; Lyu, L.; Liu, Y. Privacy-Preserving Blockchain-Based Federated Learning for IoT Devices. *IEEE Internet Things J.* **2021**, *8*, 1817–1829. [CrossRef]

13. Zhu, L.; Liu, Z.; Han, S. Deep leakage from gradients. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 14774–14784 . [CrossRef]
14. Yang, Z.; Zhang, J.; Chang, E.C.; Liang, Z. Neural network inversion in adversarial setting via background knowledge alignment. In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, London, UK, 11–15 November 2019; ACM: New York, NY, USA, 2019; pp. 225–240.
15. Lin, L.; Zhang, X. PPVerifier: A Privacy-Preserving and Verifiable Federated Learning Method in Cloud-Edge Collaborative Computing Environment. *IEEE Internet Things J.* **2022**, *10*, 8878–8892.
16. Gao, W.; Guo, S.; Zhang, T.; Qiu, H.; Wen, Y.; Liu, Y. Privacy-preserving collaborative learning with automatic transformation search. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 19–25 June 2021; IEEE: Toulouse, France, 2021; pp. 114–123. [CrossRef]
17. Geiping, J.; Bauermeister, H.; Dröge, H.; Moeller, M. Inverting gradients-how easy is it to break privacy in federated learning? *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 16937–16947.
18. Baruch, G.; Baruch, M.; Goldberg, Y. A little is enough: Circumventing defenses for distributed learning. *Adv. Neural Inf. Process. Syst.* **2019**, *32, 8635–8645* .
19. Bagdasaryan, E.; Veit, A.; Hua, Y.; Estrin, D.; Shmatikov, V. How to backdoor federated learning. In Proceedings of the International Conference on Artificial Intelligence and Statistics, PMLR, Online, 26–28 August 2020; pp. 2938–2948.
20. Islam, A.; Al Amin, A.; Shin, S.Y. FBI: A federated learning-based blockchain-embedded data accumulation scheme using drones for Internet of Things. *IEEE Wirel. Commun. Lett.* **2022**, *11*, 972–976.
21. Liu, J.; Huang, J.; Zhou, Y.; Li, X.; Ji, S.; Xiong, H.; Dou, D. From distributed machine learning to federated learning: A survey. *Knowl. Inf. Syst.* **2022**, *64*, 885–917. [CrossRef]
22. Briggs, C.; Fan, Z.; Andras, P. A review of privacy-preserving federated learning for the Internet-of-Things. In *Federated Learning Systems: Towards Next-Generation AI*; Springer: Cham, Switzerland, 2021; pp. 21–50. [CrossRef]
23. Pathak, M.; Rane, S.; Raj, B. Multiparty differential privacy via aggregation of locally trained classifiers. *Adv. Neural Inf. Process. Syst.* **2010**, *23, 1876–1884* .
24. Zhao, L.; Wang, Q.; Zou, Q.; Zhang, Y.; Chen, Y. Privacy-preserving collaborative deep learning with unreliable participants. *IEEE Trans. Inf. Forensics Secur.* **2019**, *15*, 1486–1500.
25. 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; ACM: New York, NY, USA, 2017; pp. 1175–1191. [CrossRef]
26. Shokri, R.; Shmatikov, V. Privacy-preserving deep learning. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, 12–16 October 2015; ACM: New York, NY, USA, 2015; pp. 1310–1321.
27. Blanchard, P.; El Mhamdi, E.M.; Guerraoui, R.; Stainer, J. Machine learning with adversaries: Byzantine tolerant gradient descent. *Adv. Neural Inf. Process. Syst.* **2017**, *30, 118–128* .
28. Yin, D.; Chen, Y.; Kannan, R.; Bartlett, P. Byzantine-robust distributed learning: Towards optimal statistical rates. In Proceedings of the International Conference on Machine Learning, PMLR, Stockholm, Sweden, 10–15 July 2018; pp. 5650–5659.
29. Yang, Z.; Bajwa, W.U. ByRDiE: Byzantine-resilient distributed coordinate descent for decentralized learning. *IEEE Trans. Signal Inf. Process. Over Netw.* **2019**, *5*, 611–627.
30. Fang, C.; Yang, Z.; Bajwa, W.U. BRIDGE: Byzantine-resilient decentralized gradient descent. *IEEE Trans. Signal Inf. Process. Over Netw.* **2022**, *8*, 610–626. [CrossRef]
31. Guerraoui, R.; Rouault, S. The hidden vulnerability of distributed learning in byzantium. In Proceedings of the International Conference on Machine Learning, PMLR, Stockholm, Sweden, 10–15 July 2018; pp. 3521–3530. [CrossRef]
32. Li, T.; Hu, S.; Beirami, A.; Smith, V. Ditto: Fair and robust federated learning through personalization. In Proceedings of the International Conference on Machine Learning, PMLR, Virtual, 18–24 July 2021; pp. 6357–6368.
33. Lin, J.; Du, M.; Liu, J. Free-riders in federated learning: Attacks and defenses. *arXiv* **2019**, arXiv:1911.12560.
34. Tolpegin, V.; Truex, S.; Gursoy, M.E.; Liu, L. Data poisoning attacks against federated learning systems. In Proceedings of the Computer Security—ESORICS 2020: 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, 14–18 September 2020; Proceedings, Part I 25; Springer: Berlin/Heidelberg, Germany, 2020; pp. 480–501.
35. So, J.; Güler, B.; Avestimehr, A.S. Byzantine-resilient secure federated learning. *IEEE J. Sel. Areas Commun.* **2020**, *39*, 2168–2181.
36. Guo, S.; Zhang, T.; Yu, H.; Xie, X.; Ma, L.; Xiang, T.; Liu, Y. Byzantine-resilient decentralized stochastic gradient descent. *IEEE Trans. Circuits Syst. Video Technol.* **2021**, *32*, 4096–4106. [CrossRef]
37. Miao, Y.; Liu, Z.; Li, H.; Choo, K.K.R.; Deng, R.H. Privacy-preserving Byzantine-robust federated learning via blockchain systems. *IEEE Trans. Inf. Forensics Secur.* **2022**, *17*, 2848–2861. [CrossRef]
38. Zhang, C.; Xie, Y.; Bai, H.; Yu, B.; Li, W.; Gao, Y. A survey on federated learning. *Knowl.-Based Syst.* **2021**, *216*, 106775. [CrossRef]
39. Yin, X.; Zhu, Y.; Hu, J. A comprehensive survey of privacy-preserving federated learning: A taxonomy, review, and future directions. *ACM Comput. Surv.* **2021**, *54*, 1–36. [CrossRef]

40. Xie, C.; Koyejo, S.; Gupta, I. Zeno: Distributed stochastic gradient descent with suspicion-based fault-tolerance. In Proceedings of the International Conference on Machine Learning, PMLR, Long Beach, CA, USA, 9–15 June 2019; pp. 6893–6901. [CrossRef]
41. Chen, M.; Shlezinger, N.; Poor, H.V.; Eldar, Y.C.; Cui, S. Communication-efficient federated learning. *Proc. Natl. Acad. Sci. USA* **2021**, *118*, e2024789118.
42. 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: Toulouse, France, 2019; pp. 954–964. [CrossRef]
43. Hamer, J.; Mohri, M.; Suresh, A.T. Fedboost: A communication-efficient algorithm for federated learning. In Proceedings of the International Conference on Machine Learning, PMLR, Virtual, 13–18 July 2020; pp. 3973–3983.
44. Hao, M.; Li, H.; Luo, X.; Xu, G.; Yang, H.; Liu, S. Efficient and privacy-enhanced federated learning for industrial artificial intelligence. *IEEE Trans. Ind. Inform.* **2019**, *16*, 6532–6542.
45. Xu, R.; Baracaldo, N.; Zhou, Y.; Anwar, A.; Ludwig, H. Hybridalpha: An efficient approach for privacy-preserving federated learning. In Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security, London, UK, 15 November 2019; ACM: New York, NY, USA, 2019; pp. 13–23. [CrossRef]
46. Truex, S.; Liu, L.; Chow, K.H.; Gursoy, M.E.; Wei, W. LDP-Fed: Federated learning with local differential privacy. In Proceedings of the Third ACM International Workshop on Edge Systems, Analytics and Networking, Heraklion, Greece, 27 April 2020; ACM: New York, NY, USA, 2020; pp. 61–66.
47. Yu, H.; Liu, Z.; Liu, Y.; Chen, T.; Cong, M.; Weng, X.; Niyato, D.T.; Yang, Q. A Fairness-aware Incentive Scheme for Federated Learning. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*; ACM: New York, NY, USA, 2020.
48. Zhan, Y.; Li, P.; Qu, Z.; Zeng, D.; Guo, S. A learning-based incentive mechanism for federated learning. *IEEE Internet Things J.* **2020**, *7*, 6360–6368.
49. Rivest, R.L.; Adleman, L.; Dertouzos, M.L. On data banks and privacy homomorphisms. *Found. Secur. Comput.* **1978**, *4*, 169–180. [CrossRef]
50. Acar, A.; Aksu, H.; Uluagac, A.S.; Conti, M. A survey on homomorphic encryption schemes: Theory and implementation. *ACM Comput. Surv.* **2018**, *51*, 1–35.
51. 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: Toulouse, France, 2017; pp. 3–18. [CrossRef]
52. Liu, X.; Li, H.; Xu, G.; Chen, Z.; Huang, X.; Lu, R. Privacy-enhanced federated learning against poisoning adversaries. *IEEE Trans. Inf. Forensics Secur.* **2021**, *16*, 4574–4588.
53. Xu, P.; Hu, M.; Chen, T.; Wang, W.; Jin, H. LaF: Lattice-based and communication-efficient federated learning. *IEEE Trans. Inf. Forensics Secur.* **2022**, *17*, 2483–2496. [CrossRef]
54. Beimel, A. Secret-sharing schemes: A survey. In Proceedings of the Coding and Cryptology: Third International Workshop, IWCC 2011, Qingdao, China, 30 May–3 June 2011; Proceedings 3; Springer: Berlin/Heidelberg, Germany, 2011; pp. 11–46. [CrossRef]
55. Shamir, A. How to share a secret. *Commun. ACM* **1979**, *22*, 612–613.
56. Dwork, C. Differential privacy. In Proceedings of the Automata, Languages and Programming: 33rd International Colloquium, ICALP 2006, Venice, Italy, 10–14 July 2006; Proceedings, Part II 33; Springer: Berlin/Heidelberg, Germany, 2006; pp. 1–12. [CrossRef]
57. El Ouadrhiri, A.; Abdelhadi, A. Differential privacy for deep and federated learning: A survey. *IEEE Access* **2022**, *10*, 22359–22380.
58. Zhao, Y.; Zhao, J.; Yang, M.; Wang, T.; Wang, N.; Lyu, L.; Niyato, D.; Lam, K.Y. Local differential privacy-based federated learning for internet of things. *IEEE Internet Things J.* **2020**, *8*, 8836–8853. [CrossRef]
59. 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]
60. Ayadi, H.; Zouinkhi, A.; Boussaid, B.; Abdelkrim, M.N. A machine learning methods: Outlier detection in wsn. In Proceedings of the 2015 16th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA), Monastir, Tunisia, 21–23 December 2015; IEEE: Toulouse, France, 2015; pp. 722–727. [CrossRef]
61. Pukelsheim, F. The three sigma rule. *AM STAT* **1994**, *48*, 88–91.
62. Curtis, A.E.; Smith, T.A.; Ziganshin, B.A.; Elefteriades, J.A. The mystery of the Z-score. *Aorta* **2016**, *4*, 124–130.
63. Ester, M.; Kriegel, H.P.; Sander, J.; Xu, X. Density-based spatial clustering of applications with noise. *Int. Conf. Knowl. Discov. Data Min.* **1996**, *96*, 226–231. [CrossRef]
64. Nasr, M.; Shokri, R. Improving deep learning with differential privacy using gradient encoding and denoising. *arXiv* **2020**, arXiv:2007.11524.
65. Li, S.; Ngai, E.; Voigt, T. Byzantine-robust aggregation in federated learning empowered industrial iot. *IEEE Trans. Ind. Inform.* **2021**, *19*, 1165–1175.
66. Wang, R.; Wang, X.; Chen, H.; Picek, S.; Liu, Z.; Liang, K. BRIEF but Powerful: Byzantine-Robust and Privacy-Preserving Federated Learning via Model Segmentation and Secure clustering. *arXiv* **2022**, arXiv:2208.10161. [CrossRef]
67. Zhai, K.; Ren, Q.; Wang, J.; Yan, C. Byzantine-robust federated learning via credibility assessment on non-IID data. *arXiv* **2021**, arXiv:2109.02396.

68. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. Pytorch: An imperative style, high-performance deep learning library. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 8024–8035 .
69. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324.