applied sciences

MDPI

Article

FedCrow: Federated-Learning-Based Data Privacy Preservation in Crowd Sensing

Jun Ma [1], Long Chen [1], Jian Xu [2] and Yaoxuan Yuan [3,*]

[1] School of Defence Science and Technology, Xi'an Technological University, Xi'an 710021, China
[2] School of Computer Science and Engineering, Xi'an Technological University, Xi'an 710021, China
[3] Institute of AI and Data Science, Xi'an Technological University, Xi'an 710021, China
* Correspondence: smartyuan66@163.com

Abstract: In the process of completing large-scale and fine-grained sensing tasks for the new generation of crowd-sensing systems, the role of analysis, reasoning, and decision making based on artificial intelligence has become indispensable. Mobile crowd sensing, which is an open system reliant on the broad participation of mobile intelligent terminal devices in data sensing and computation, poses a significant risk of user privacy data leakage. To mitigate the data security threats that arise from malicious users in federated learning and the constraints of end devices in crowd-sensing applications, which are unsuitable for high computational overheads associated with traditional cryptographic security mechanisms, we propose FedCrow, which is a federated-learning-based approach for protecting crowd-sensing data that integrates federated learning with crowd sensing. FedCrow enables the training of artificial intelligence models on multiple user devices without the need to upload user data to a central server, thus mitigating the risk of crowd-sensing user data leakage. To address security vulnerabilities in the model data during the interaction process in federated learning, the system employs encryption methods suitable for crowd-sensing applications to ensure secure data transmission during the training process, thereby establishing a secure federated-learning framework for protecting crowd-sensing data. To combat potential malicious users in federated learning, a legitimate user identification method based on the user contribution level was designed using the gradient similarity principle. By filtering out malicious users, the system reduces the threat of attacks, thereby enhancing the system accuracy and security. Through various attack experiments, the system's ability to defend against malicious user attacks was validated. The experimental results demonstrate the method's effectiveness in countering common attacks in federated learning. Additionally, through comparative experiments, suitable encryption methods based on the size of the data in crowd-sensing applications were identified to effectively protect the data security during transmission.

Keywords: crowd sensing; federated learning; contribution level; gradient similarity

Citation: Ma, J.; Chen, L.; Xu, J.; Yuan, Y. FedCrow: Federated-Learning-Based Data Privacy Preservation in Crowd Sensing. *Appl. Sci.* **2024**, *14*, 4788. https://doi.org/10.3390/app14114788

Academic Editor: José Salvador Sánchez Garreta

Received: 22 April 2024
Revised: 28 May 2024
Accepted: 29 May 2024
Published: 31 May 2024

1. Introduction

In the process of completing large-scale, fine-grained sensing tasks, the multidimensional and massive amounts of sensing data collected by the new generation of crowd-sensing systems can support intelligent applications, such as smart community services; environmental monitoring; intelligent transportation; and navigation through effective processing, monitoring, analysis, inference, and decision making using artificial intelligence [1–3]. Mobile crowd sensing is an open system based on smart mobile devices, where numerous mobile terminal devices collaborate to collect and preprocess a large amount of information. These data are then uploaded to backend servers for further processing through various connectivity methods, such as 3G/4G/5G cellular networks, Wi-Fi, and Bluetooth [4]. Typically, these data contain sensitive information about end users, such as their personal identity, bank account details, and service access records. The wireless communication, collaborative information processing, and centralized storage of user data in

backend servers in this open environment make mobile crowd-sensing systems vulnerable to attacks [5,6], presenting challenges related to user data security and privacy.

Federated learning (FL) [7] is a distributed machine learning technique that involves training a shared artificial intelligence model collaboratively across multiple devices. During this process, training data (or private data) remain stored locally on the devices, with local models being uploaded to a server for the aggregation of all models [8,9]. In federated learning, training data are kept locally, and the central server receives gradient information to update the global model. Data collection, storage, and server training are separated, thereby ensuring the security of private data. Introducing federated learning into crowd-sensing applications becomes particularly crucial in addressing privacy concerns that arise from user data uploads [10,11].

Applying FL to crowd-sensing systems can enhance the data security, but it also introduces the risk of malicious users launching model extraction attacks and model reverse engineering attacks, leading to privacy breaches [12]. This paper proposes a method for securing user data in crowd-sensing applications based on federated learning named FedCrow. First, we constructed an FL framework tailored for crowd-sensing applications. Next, an appropriate encryption method was selected based on the crowd-sensing application scenario to ensure data security during transmission, where the goal was to build a secure federated-learning framework. Subsequently, malicious users were identified and filtered out using a user contribution level to minimize potential harm they may cause. Finally, through experiments, the security of the federated-learning framework based on credibility scores for legitimate user authentication was validated in terms of aspects such as resisting data poisoning, model attacks, inference attacks, and differential privacy attacks. The security of the transmitted model data was also validated through experiments by selecting suitable encryption methods based on the size of the data during the reference model training.

The main contributions of this work are summarized as follows:

- We propose a federated-learning-based framework for protecting user data in crowd-sensing applications called FedCrow. It utilizes federated learning to locally store sensing data on terminal devices, exchanges only gradient parameters between terminals and servers, and trains models among multiple terminal users in crowd-sensing applications. Additionally, it employs encryption methods suitable for crowd-sensing applications to ensure secure data transmission during the training process, thus constructing a secure federated-learning framework tailored for crowd-sensing applications. We designed a legitimate user identification method based on user contribution using the gradient similarity principle, filtering out malicious users to reduce the threat of attacks on the system.
- We implemented FedCrow on the deployed prototype system. Extensive experimental results demonstrate that the proposed method achieved high security and performance.

The rest of this paper is organized as follows: Section 2 briefly reviews the related works, followed by the main steps of the FedCrow design and implementation in Section 3. In Section 4, we evaluate the performance of the federated-learning-based user data protection system for crowd sensing, and finally, we conclude our paper in Section 5.

2. Related Work

2.1. Crowd-Sensing Data Security

DAEQ-FL [13] proposed an efficient encryption-based data security protocol that does not require third-party assistance to generate keys. It employs model parameter quantization processing and approximate aggregation combined with threshold-secret-sharing technology to reduce communication costs and computational complexity. EPTD [14] presented an efficient privacy protection method that shields vehicle crowd-sensing systems with double masking and allows for offline uploads while maintaining privacy and accuracy. ESPPTD [15] introduced an improved slice-based privacy protection method that ensures data privacy while simplifying the computation and communication load.

However, relying solely on encryption methods to protect data security may increase the system burden and pose security risks. A privacy protection scheme for crowd-sensing security that utilizes ciphertext policy attribute-based encryption and access update policies (CP-ABE-UP) was proposed, as well as ID-based signatures for identity authentication and privacy protection [16].

In response to the impact of malicious users in crowd sensing, some researchers proposed providing services based on the screening of user identity information. A privacy protection scheme utilizing the K-anonymity mechanism was proposed [17]. Rahaman et al. [18] presented a revocable method for protecting user identity information using group signatures. A non-linkable but accountable identity authentication scheme using pseudonymous signatures was constructed [19]. Reference [20] developed a stronger privacy protection method using efficient protocol signatures. PACE [21] proposed a privacy protection incentive method based on data quality, while reference [22] introduced a personalized privacy protection incentive mechanism based on contracts. Although these studies incorporated mechanisms for screening malicious users, the incurred costs were substantial.

In the field of Internet of things (IoT) data privacy and security, a number of studies focused on granular disclosure control techniques that provide fine-grained control over data sharing in different contexts. A. Soltani Panah et al. [23] proposed a framework that enables users to specify how and when their data can be shared with different parties based on the specific context or situation, which enhances the privacy and security of IoT devices and applications. The method was shown to be effective in various contexts, and it provides valuable insights into the development of more secure and privacy-preserving IoT systems.

In existing mobile crowd-sensing applications, data security and user privacy protection methods based on cryptographic techniques generally suffer from high computational complexity and significant system overhead, making them less suitable for resource-constrained mobile terminal devices. Furthermore, the existing methods based on user identity and data filtering also exhibit high computational complexity and do not adequately consider the correlation between users' legitimate identities and data characteristics.

2.2. Federated-Learning Data Security

Although federated learning allows for data to be kept locally, there are still risks to data security. PrivLogit [24] proposed an algorithm that processes the horizontal distribution of data from different participants through logistic regression, integrates encrypted matrices and shared gradients, solves for optimal regression coefficients using Newton's iterative method, and improves the efficiency while addressing user privacy security. Reference [25] introduced a vertical federated-learning algorithm suitable for vertically distributed datasets. The algorithm aligns samples and calculates results by generating CLK identifiers, requires both parties to create random masks, and leverages a third-party entity to ensure training accuracy, reducing the encryption computational costs while protecting privacy information. Wu et al. [26] presented a federated-learning solution that combines adaptive gradient descent and differential privacy mechanisms, introduces differential privacy mechanisms to resist inference attacks, and provides more effective and quantifiable privacy protection for federated learning. A federated-learning-based IoT secure data-sharing mechanism FL2S [27] was proposed that employs a hierarchical asynchronous federated-learning framework. Utilizing deep reinforcement learning technology to select participants, reliable data sharing that protected the privacy of the source data was achieved. PFLM [28] introduced membership proof in federated learning, utilizing an encrypted accumulator to generate proof and publishing it on the blockchain. The PFLM joint-learning scheme was proposed, releasing the threshold assumption and designing a result verification algorithm based on the ElGamal encryption variant.

N. M. Hijazi et al. [29] focused on achieving secure federated learning in IoT communications. They proposed a solution based on fully homomorphic encryption to ensure the

privacy and security of data during model training on edge devices. By leveraging fully homomorphic encryption technology, the solution enables computations on encrypted data, thereby avoiding the leakage of sensitive information. This research enhanced the efficiency and security of federated learning while protecting user privacy.

Existing data security protection methods in federated learning based on differential privacy [30–32] and homomorphic encryption [33,34] mainly focus on safeguarding data security during transmission. However, they lack effective identification and screening of malicious users and their data before the encrypted transmission, making them insufficient to resist man-in-the-middle attacks. Moreover, these methods incur a significant system overhead. Additionally, existing approaches lack flexibility in selecting suitable encryption methods according to specific application scenarios.

3. FedCrow Design

In order to effectively protect the data security in crowd-sensing applications, based on the scenarios and characteristics of crowd sensing, this study designed a federated-learning-based data security system for crowd sensing. By introducing federated-learning models into the crowd-sensing system, it ensures that user data are only stored locally to mitigate the risk of data leakage. For this purpose, a federated-learning framework suitable for crowd sensing was designed using the TensorFlow framework based on the principles of federated learning.

Taking into account the unnecessary data leakage that can occur during data interactions in federated-learning systems, we aimed to construct a secure federated-learning framework for safeguarding data in crowd-sensing applications. To achieve this, we utilized appropriate data encryption methods to encrypt the data transmission process by considering the data volume typically encountered in crowd-sensing application scenarios.

Furthermore, the greatest threat in federated-learning systems stems from internal malicious users participating in the training process. Each user in federated learning possesses a dataset, allowing internal malicious actors to easily manipulate datasets, the training process, and the model, consequently compromising the model performance. To mitigate the risk of internal attacks, it is imperative to rigorously scrutinize users participating in the training to ensure their legitimacy. This paper proposes a legitimate user-screening method based on client-side contributions to the global model. It evaluates the contribution level of user models to the server's global model by computing the gradient similarity between different models, thereby identifying trustworthy users.

3.1. Problem Analysis

While federated learning helps to ensure user privacy and data security, there are still potential security risks that could compromise the model and data integrity. Reference [35] delved into these risks and proposed effective preventive measures. Within the threat model of federated learning, insider attacks are the most common challenge, with Byzantine attacks and witch attacks being the most effective methods [36]. Potential adversaries in the system can be classified into two types based on their level of proactiveness: semi-honest and malicious. Once semi-honest adversaries turn malicious, they can skillfully manipulate, replay, or delete information to disrupt federated-learning protocols, and even launch destructive attacks.

Federated learning consists of two stages: training and inference. During the training stage, attackers can potentially target the model through data poisoning or model poisoning. In the inference stage, attackers may interfere with the model, resulting in inaccurate outputs or the collection of false information.

Data-poisoning attacks can be categorized into clean label and dirty label: clean label assumes that attackers will not modify the labels of the training data; dirty label, on the other hand, allows for the introduction of data samples that do not match the expected labels. These two approaches can be employed in local data collection and partial model training. Dirty-label attacks are diverse, covering techniques such as label flipping and

backdooring, which can affect both local and global models, all of which can be utilized to achieve attack objectives.

During the model-training process, the involvement of malicious end users in federated learning can severely impact the integrity of the global model. They may employ tactics such as data contamination or model poisoning to undermine the reliability of the global model in federated learning. To counter such attacks, it is effective to screen participating client devices, selecting those with high reliability while excluding malicious ones. Typically, the number of malicious users in a system is relatively small, and thus, their contribution to the server's global model is either minimal or negative. Utilizing this principle, our framework calculates the gradient similarity between the user models that participate in the training and the server's global model. Users with a low gradient similarity are deemed unreliable, which enables effective screening of the legitimate users, and thereby reducing the harm caused by malicious users.

3.2. Secure Federated-Learning Framework Construction

The construction of the federated-learning framework was designed using TensorFlow, which included the construction of artificial intelligence models, model fusion, data synchronization, data transmission, and multi-threaded network communication. In federated learning, the artificial intelligence model can be replaced according to requirements, and model fusion is achieved using a weighted average method. Federated-learning model training is a cyclical process, and thus, model fusion needs to ensure that the data fused each time are from the same batch of clients; otherwise, it may cause missing client models and inaccuracies in the overall model. Data synchronization and transmission use a multi-threaded network communication mode. Model fusion utilizes the method of merging all the participating client-side models using the mean value of the crowd's intelligence perception.

During model fusion, it is necessary to identify which client the model belongs to and merge them based on the weights assigned to different client IDs. Before transmitting the models, all clients establish a connection with the server and send their respective ID numbers to ensure the effective filtering of legitimate clients. Model fusion is then achieved by taking the mean of the models from different clients, considering their respective weights. Data synchronization is determined based on the number of iterations of the client models. (During the first iteration, it is necessary to verify whether all client models have completed one iteration. During the second iteration, the check should ensure that all client models have completed two iterations, and so on).

The data transmission between the server and clients in federated learning is implemented using socket technology and the TCP/IP protocol. In federated learning, the server needs to collaborate with multiple clients. However, standard sockets can only support one-to-one communication, and cannot handle concurrent connections. To address this limitation, the SocketServer technology internally incorporates multi-threading and multi-processing techniques, enabling the server to simultaneously handle multiple client socket requests. Therefore, the server utilizes SocketServer to facilitate network transmission. Once the SocketServer is running, it continuously waits for client connections until the participating users connect to the server to receive tasks and models. Each client then performs local training using the tasks and initial models received from the server. Finally, all clients send their trained results back to the server.

The data exchange process in federated-learning systems may lead to unnecessary data leakage. To address this issue, encryption technology can be utilized to perform computations on encrypted data during the model-training process. This approach helps to construct a secure federated-learning framework for protecting the security of crowdsourced data. By studying the data encryption methods used in federated learning and analyzing the principles and characteristics of additive homomorphic encryption algorithms, multiplicative homomorphic encryption algorithms, and differential privacy, we can choose suitable methods to protect data security based on the application scenarios of

crowd sensing. When dealing with large amounts of data, differential privacy methods should be preferred to ensure data security. When dealing with small amounts of data, homomorphic encryption methods can be considered.

3.3. The Contribution-Level-Based Legitimate User Selection Approach

Regardless of the method employed, data security threats stem from malicious users who attempt to disrupt the system's model accuracy by engaging in abnormal training or uploading incorrect data. Furthermore, they may exploit vulnerabilities to steal other users' data through gradient analysis, posing serious security risks. Typically, the number of malicious users is small, and their contribution to the overall model on the server is minimal or negative. Leveraging this principle, the similarity between the gradients of the models contributed by participating users and the total model on the server is computed. A low similarity indicates unreliability, thereby effectively filtering out legitimate users and reducing the attack threats posed by malicious users.

The approach proposed in this paper measures the contribution level of different clients to the aggregated model based on gradients. It uses cosine similarity to calculate the similarity between the client model and the aggregated model, and uses this similarity to quantify the client's contribution level.

In gradient-based learning, high-quality client gradients typically exhibit a high similarity to the total model gradient, indicating minimal model loss. Conversely, low-quality client gradients demonstrate lower similarity to the total model gradient, suggesting greater model loss.

In order to locate the minimum point of the loss function, first-order optimization can be achieved using the gradient descent method. This method involves searching in the opposite direction until reaching a local minimum, enabling the identification of the lowest point and the values of model parameters. Through iterative processes of gradient ascent and gradient descent, local optimal solutions can be effectively discovered to assess the similarity of models or data. The gradient describes the direction of the derivative of the function at a specific location, which reflects the speed and rate of change of the function. In [37], the gradient value was leveraged as structural information in images, leading to the proposal of two structural similarity algorithms: gradient structural similarity (GSSIM) and gradient structural dissimilarity (GSSD), both of which have demonstrated significant achievements in evaluating images with certain types of distortions.

The neural network structure is illustrated in Figure 1. To solve for the minimum loss function, the gradient descent method is employed to compute and update the weight matrix W. The weight matrix is a parameter matrix obtained through continuous updates. By calculating the similarity of the weight matrix, inferences can be made about the similarity of gradients. Directly utilizing the weight parameters can reduce the computational energy required for recalculating gradients, thereby lowering the system burden.

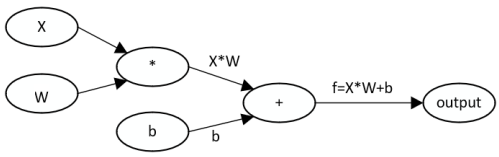


Figure 1. Neural network architecture.

This study investigated the contribution of different clients to the overall model after model aggregation from the perspective of gradients. The cosine similarity was utilized to compute the similarity between client models and the overall model, and the similarity was used to calculate the reputation scores of clients. The calculation of the loss function for a simple linear regression model is represented by Formula (1). $L(w, b)$ is the loss function, which measures the discrepancy between the model's predicted values and the actual values. $\frac{1}{N}\sum_{i=1}^{N}$ represents the average across all samples, where N is the number of

samples. y_i represents the true label of the ith sample. $f(w_{xi} + b)$ is the model's prediction for the ith sample, where w is the weight of the feature and b is the bias. $(y_i - f(w_{xi} + b))^2$ represents the square of the difference between the actual value y_i and the model's predicted value $f(w_{xi} + b)$. This is commonly known as the mean squared error (MSE) loss function.

$$L(w, b) = \frac{1}{N} \sum_{i=1}^{N} (y_i - f(w_{xi} + b))^2 \tag{1}$$

The gradient is a vector whose components are the partial derivatives of the function in each variable direction. For a two-dimensional vector function $u(x, y)$, its gradient is denoted by ∇u, which is expressed as Formula (2). g is a vector field represented as the sum of its components: the component in the x-direction is $\frac{\partial u}{\partial x}\, i$, and the component in the y-direction is $\frac{\partial u}{\partial y}\, j$. $\frac{\partial u}{\partial x}$ and $\frac{\partial u}{\partial y}$ are, respectively, the partial derivatives of the scalar function u with respect to the variables x and y. i and j are the unit vectors in the two-dimensional Cartesian coordinate system that point in the directions of x and y, respectively. This gradient vector represents the rate and direction of change of the function $u(x, y)$ in the two-dimensional plane.

$$g = \frac{\partial u}{\partial x}\, i + \frac{\partial u}{\partial y}\, j \tag{2}$$

Assuming a learning rate of r (typically a value like 0.001, 0.0001, 0.00001, or 0.000001), updating the gradient once yields

$$x_1 = x_0 - r \cdot g(x_0)$$

This formula is the update rule in the gradient descent algorithm. It means moving the current parameters x_0 along the opposite direction of the gradient (i.e., the negative direction of the gradient) by a distance r. The purpose of doing this is to gradually decrease the objective function in the parameter space in order to find a local minimum or global minimum. x_0 is the current parameter value, $g(x_0)$ is the gradient of the objective function at x_0, and r is the learning rate (or step size) that controls the magnitude of each update.

Suppose the gradient of the loss function L at w_0 is $g(w_0) = \frac{\partial L}{\partial w_0}$. After updating, the weight becomes $w_1 = w_0 - 1 \cdot g(w_0)$, where $w_n = w_0 - 1 \cdot \sum_{i=0}^{n-1} g(w_i)$. The values of w_n and the sum of the gradients from 0 to $n - 1$ are determined. When the gradient stabilizes, it indicates that the loss function is approaching the optimal value. The closer it gets to the optimal value, the lower the loss function becomes. From the above, it can be inferred that the gradient increases with the increase in the loss function. As the parameter w gets updated, the loss function decreases, leading to a further reduction in the gradient until the loss function reaches its lowest point and the gradient descends to 0. At this point, the optimal solution can be achieved.

Based on the above inference, to achieve the minimum loss function in a neural network, it is necessary to calculate the gradient of the loss function with respect to the weights W using gradient descent, update the weights W accordingly, and then derive the weight matrix. Consequently, the weight matrix is obtained through the iterative updates during the gradient descent process. Therefore, computing the similarity of the gradients is equivalent to computing the similarity of the weight matrices. The calculation of the parameter matrices can utilize the cosine similarity Formula (3).

$$\cos\theta = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}} \tag{3}$$

The formula for cosine similarity is used to measure the similarity between two vectors. This formula calculates the cosine of the angle between vectors A and B, representing their similarity. A cosine similarity close to 1 indicates a high similarity between the two vectors, while a value close to -1 indicates they are dissimilar. A cosine similarity near 0 means

there is no correlation between the vectors. θ is the angle between the two vectors; $A \cdot B$ is the dot product of vectors A and B; $\|A\|$ and $\|B\|$ are the norms (or lengths) of vectors A and B, respectively; and A_i and B_i are the ith elements of vectors A and B, respectively.

The client contribution level is used to identify and eliminate potential malicious users, as their contributions are typically low. By comparing the similarity of each client to the total model, participants who contribute less or negatively to the overall model are identified as unreliable or malicious users and are removed. In each round of model updating, participants whose similarity to the total model falls below a certain threshold are excluded from subsequent rounds. Removed participants will no longer participate in the model training, thus reducing the threat of attacks from malicious users during the model training process.

3.4. System Design

The model training process for identifying legitimate users based on the user contribution level is illustrated in Figure 2.

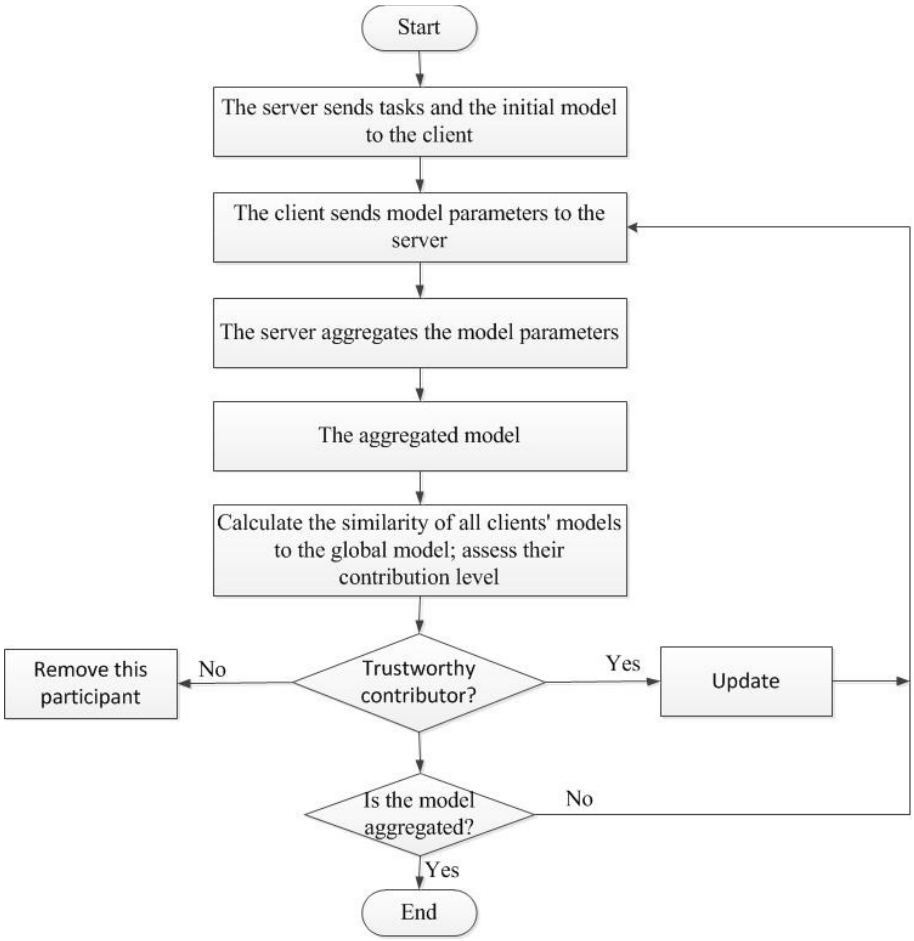


Figure 2. The model-training process for identifying legitimate users.

The specific implementation steps are as follows:

(1) The server sends tasks and the initial model to the clients.

(2) The clients send model parameters to the server.

(3) After receiving the parameters sent by all clients, the server performs a fusion based on the mean of the user models (with each client having equal weight after excluding the malicious clients).

(4) After obtaining the server's global model, the similarity of all clients to the global model is calculated using the cosine similarity algorithm, and this is used as the contribution

of each client (clients with a high contribution are considered reliable users, while those with a low contribution are considered unreliable users), and the unreliable participants with a low contribution are removed.

(5) The server sends the aggregated model to all clients for further training.

(6) Steps (2) to (5) are repeated until the server model converges.

3.5. Analysis and Selection of Encryption Algorithm in Federated Learning

Based on the analysis of commonly used encryption methods in federated learning, suitable encryption algorithms for crowd-sensing applications were selected to enhance the data security during information exchange. By analyzing the encryption effectiveness and computational overhead, encryption algorithms were flexibly chosen based on the data volume in crowd-sensing applications. Generally, federated learning employs homomorphic encryption and a differential privacy encryption algorithm. The following section mainly introduces these two encryption methods.

Homomorphic encryption algorithms are widely used in federated learning. According to the characteristics of homomorphic encryption, it is most effective to use this algorithm when dealing with small data volumes. It not only effectively enhances the data security but also does not compromise the data availability. However, when dealing with exceptionally large data volumes, using homomorphic encryption significantly increases the computational overhead, leading to longer processing times. Currently, federated-learning applications for crowd sensing are mainly focused on large-scale industrial scenarios, where most clients are mobile smart devices with limited resources. Therefore, in federated-learning-based crowd-sensing systems, using homomorphic encryption is not suitable when dealing with large data volumes.

Differential privacy adds noise to data to mitigate the risk of data leakage. However, adding noise inevitably reduces the data availability, which is an unavoidable factor of differential privacy. In cases of small data volumes, adding noise can significantly impact the data availability, leading to a substantial decrease in the model accuracy.

When the data volume in a crowd-sensing system is large, adding noise not only reduces the security risks associated with data leakage but also, due to the larger data volume, the impact of the added noise on the overall data is relatively minor. Tables 1 and 2 present the encryption results for matrices of sizes 4×8 and $4 \times 8 \times 30$, respectively. According to these tables, it can be observed that when the data volume was large, both the addition and multiplication homomorphic encryptions required significant encryption and decryption times. Conversely, in scenarios with smaller data volumes, the differential privacy method resulted in a substantial loss of model accuracy.

Table 1. Comparison of encryption methods for matrices of size 4×8.

Encryption Method	Encryption/Decryption Time	Model Accuracy
Additive homomorphic encryption	8.69 s	97.48%
Multiplicative homomorphic encryption	28.57 s	97.32%
Differential privacy	None	69.56%

Table 2. Comparison of encryption methods for a $4 \times 8 \times 30$ matrix.

Encryption Method	Encryption/Decryption Time	Model Accuracy
Additive homomorphic encryption	320.78 s	97.44%
Multiplicative homomorphic encryption	1125.82 s	97.19%
Differential privacy	None	95.45%

After studying the data encryption methods used in federated learning and conducting a thorough analysis of the principles and characteristics of additive homomorphic

encryption, multiplicative homomorphic encryption, and differential privacy, we concluded, in light of the application scenarios of crowd sensing, that when dealing with large data volumes, it is advisable to opt for the differential privacy method to safeguard the data security. Conversely, when the data volume is relatively small, consideration should be given to employing homomorphic encryption methods.

4. Evaluation

In this section, we discuss the validation of the security of the federated-learning system trained for the proposed crowd-sensing application through experiments. For this purpose, we constructed a federated-learning system and simulated a crowd-sensing environment for validation experiments. We selected open-source large-scale datasets for actual neural network training tasks in the experiments. The experiments consisted of three parts: system accuracy experiments; comparative experiments on data transmission encryption in federated learning; and experiments on the system's resistance to common attack threats, including data poisoning, model attacks, and inference attacks. In the data transmission security experiments, we conducted comparative experiments using additive homomorphic encryption algorithms, multiplicative homomorphic encryption algorithms, and differential privacy. The experimental results and analysis are provided in the aforementioned section. The results of the experiments that assessed the system's resistance to attacks verified the attack resistance of the secure federated-learning framework based on the user-contribution-level-based malicious-user-screening mechanisms from the perspectives of the impact on model accuracy when malicious nodes were present in the user base and when poisoning, model attacks, and inference attacks were launched.

4.1. Experimental Environment and Parameter Settings

The crowd-sensing environment: adopted a constructed crowd-sensing federated-learning framework and simulated the distribution of crowd-sensing devices with 100 randomly generated coordinate nodes. On average, each client had 200 sample data points.

Federated-learning environment: An artificial intelligence model, which was built using Keras, consisted of four fully connected layers to form a neural network model. The data features consisted of 30 attributes, hence the input x was of size 30. The four-layer neural network was composed of 32, 16, 8, and 1 neurons. The data used in this experiment were from the Credit Card Fraud Detection dataset (from Kaggle), which contains records of credit card transactions made by European cardholders over two days in September 2013. The dataset consists of 285,299 entries, with 284,807 transactions and 492 suspected cases of fraud. The machine learning model in this experiment was built using Keras, and employed four fully connected layers to construct the neural network model structure illustrated in Figure 3.

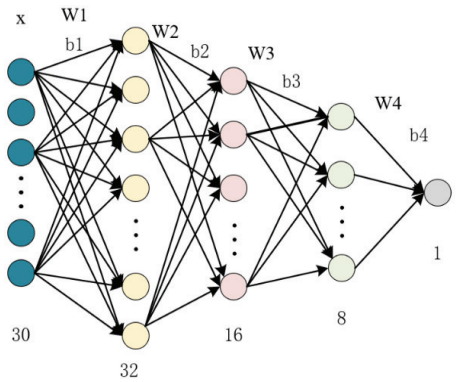


Figure 3. The neural network model composed of four fully connected layers.

Server environment: processor, AMD 5800H (Asus, Kowloon, Hong Kong, China); graphics card, NVIDIA RTX 3060 (NVIDIA, Santa Clara, CA, USA); 16 GB RAM; and 1 TB hard drive.

Client environment: The client utilized both a Lenovo Legion laptop (Lenovo, Beijing, China) and a mobile emulator. The mobile emulator was operated using a Nox App Player (Nox Limited, Hong Kong, China), with an Android version of 7.1.2, 4 GB of RAM, and 128 GB of storage capacity. Within the Android environment, Python runtime was implemented using qpython3. The QPython 3.9.1 software had a size of 23.1 MB. Communication was established using IP addresses and port numbers.

Encryption parameters and algorithm: with a dataset size of 285,299 entries, which is relatively large, this experiment utilized differential privacy algorithms to encrypt the data during transmission.

Simulated attack types: data poisoning [38], model attacks [39], and inference attacks [40,41].

User types: this included normal users, unreliable users, and malicious users.

4.2. Experimental Results And Analysis

The experiment selected three clients, with a learning rate of 0.001 and 10 training epochs, and utilized the Adam optimizer for conducting security validation experiments on model data. The parameter settings are shown in Table 3.

Table 3. Parameter settings.

Parameter	Value
Number of clients	3
Learning rate	0.001
Training epochs	10
Optimizer	Adam

4.2.1. System Accuracy

The convergence curve of the model is shown in Figure 4. From the figure, it can be seen that the model converged after the seventh iteration, achieving a precise model accuracy. Therefore, in this experiment, the number of training epochs for federated learning was greater than seven.

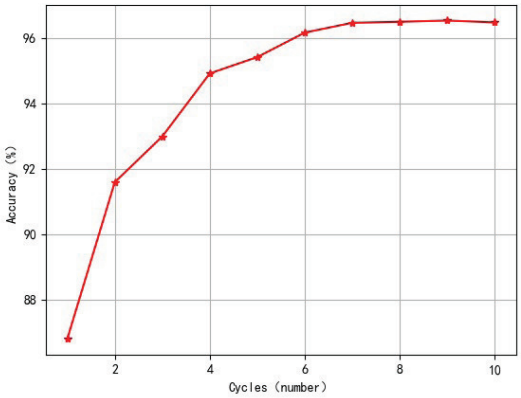


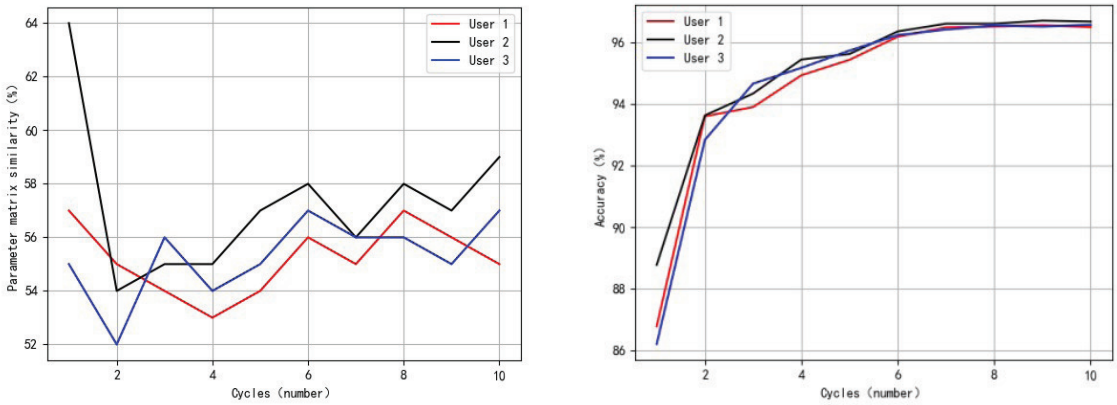
Figure 4. The convergence curve of the model.

4.2.2. Attack Resistance Experiment

In federated learning, clients only need to upload model parameters, while user data are kept locally. This approach largely addresses data privacy concerns compared with traditional distributed machine learning. However, due to the openness of federated learning, there may be malicious users that aim to obtain access to the server model through

training or compromise the model's accuracy. To tackle this issue, this paper proposes a legitimate user authentication method based on the user contribution level. It evaluates the credibility of different clients by calculating their contributions to the server model. Currently, the main threats to the federated-learning model and data security can be categorized into four types: data poisoning, model attacks, inference attacks, and differential privacy attacks. The following experiments targeted these attack methods to validate the security of the federated-learning framework using the user-contribution-level-based legitimate-user-screening method.

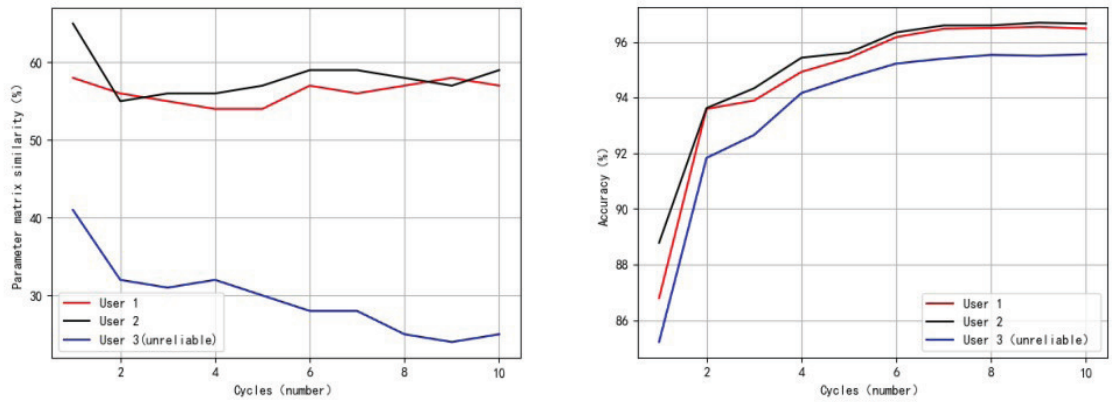
(1) All normal users: The first experiment involved three clients, all of whom were normal users, and the training data consisted of normal data. The experimental results are shown in Figure 5, where the parameter matrices of the three users were relatively similar, resulting in a comparable model similarity and accuracy between them.



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

Figure 5. Model training with only normal users. (**a**) The similarity of the parameter matrices. (**b**) The model accuracy.

(2) There are some unreliable users: In the second experiment, one unreliable user was introduced among the clients, whose data consisted of only 10% of the normal user data. Due to the smaller dataset provided, the model accuracy was relatively lower. The training results are illustrated in Figure 6. It is evident that the parameter matrix similarity of the unreliable user was lower compared with the normal users, resulting in a relatively lower model accuracy. However, the model accuracy of the normal users remained unaffected.



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

Figure 6. Model training with an unreliable user. (**a**) The similarity of the parameter matrices. (**b**) The model accuracy.

(3) Existence of data-poisoning attacks: In the third experiment, a poisoning attack was conducted. "Data poisoning" typically refers to attackers adding improper information

during the training process, enabling them to exploit this information to damage and manipulate the model, thereby achieving their intended goals. The process of a data-poisoning attack is illustrated in Figure 7.

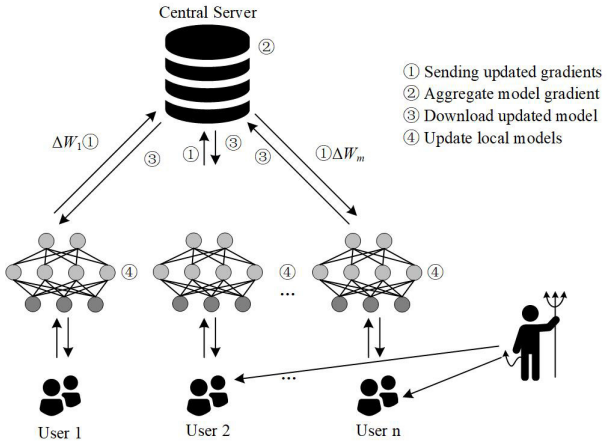
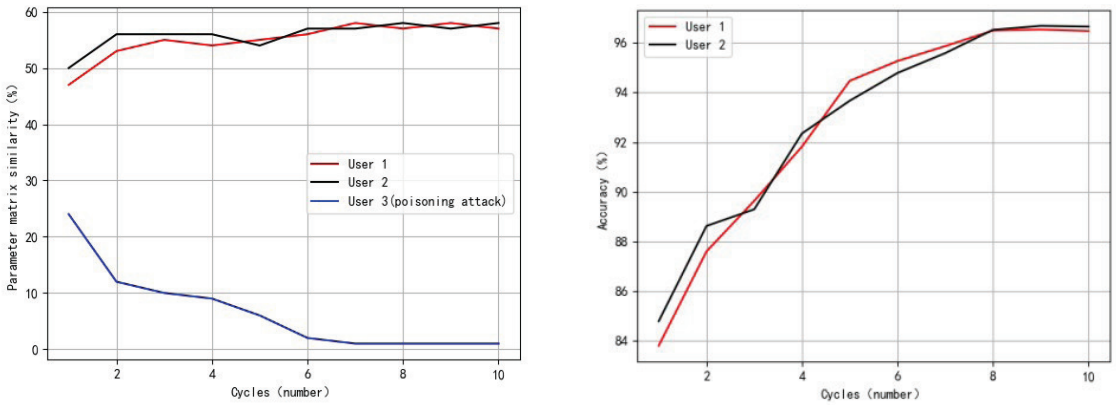


Figure 7. The process of a poisoning attack.

As shown in Figure 8, the training data used by the malicious user for the poisoning attack consisted of either malicious or tampered data, which differed significantly from the data of other normal users. By comparing the similarity of the parameter matrices, it was evident that the malicious user's parameter matrix similarity was lower compared with the normal users. Consequently, their corresponding user contribution level decreased, leading to a smaller proportion in the model aggregation. Although the poisoning attack slowed down the convergence of the model, after convergence, the malicious user that employed the poisoning attack method hardly affected the overall model accuracy.



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

Figure 8. Model training with a user that conducted a poisoning attack. (**a**) The similarity of the parameter matrices. (**b**) The model accuracy.

(4) Existence of model attacks: In the fourth experiment, a model attack was conducted. Model attacks involve altering or replacing the model parameters of clients to modify the global model, as depicted in Figure 9.

As depicted in Figure 10, a model attack involves tampering with or replacing the model parameters of clients to alter the global model. Unlike poisoning attacks, where abnormal data are used for training before sending the model to the server, model attacks directly manipulate the client's model parameters. In both cases, the uploaded client models are incorrect, exhibiting significant dissimilarity from those of other normal users, resulting in lower model similarity. Consequently, the user contribution levels of these attackers

decrease, leading to a smaller proportion in model aggregation during fusion. From the graph, it is evident that although the model attacks also slowed down the convergence of the model, the malicious users employing model attack methods hardly affected the overall model accuracy after the convergence.

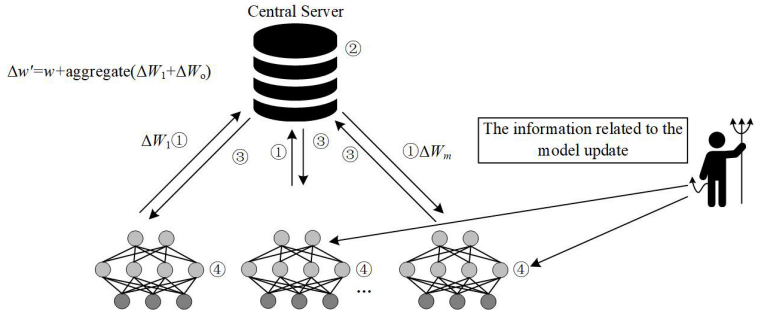
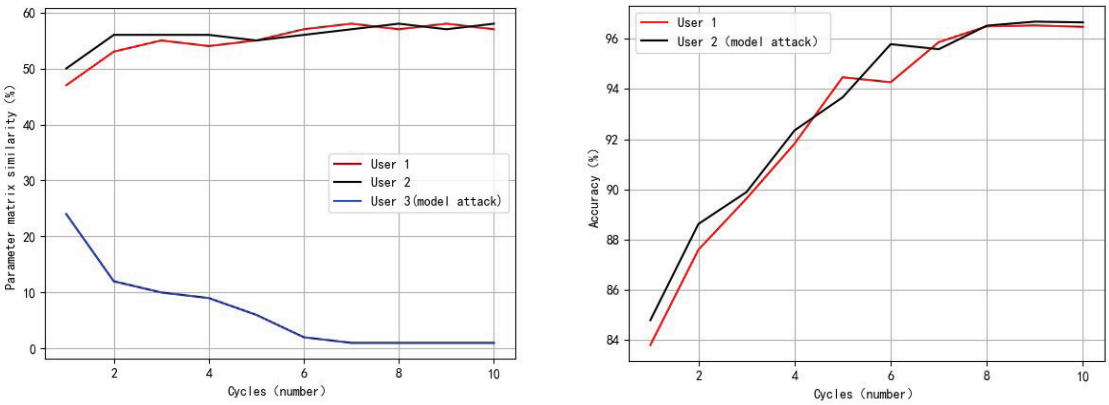


Figure 9. The process of model attacks.



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

Figure 10. Model training with a user that conducted a model attack. (**a**) The similarity of the parameter matrices. (**b**) The model accuracy.

(5) Existence of inference attacks: In the fifth experiment, an inference attack was employed. Inference attacks are techniques that gather information in various ways, including eavesdropping and monitoring, and utilize logical thinking and reasoning abilities to obtain desired information. The process of the inference attack is illustrated in Figure 11.

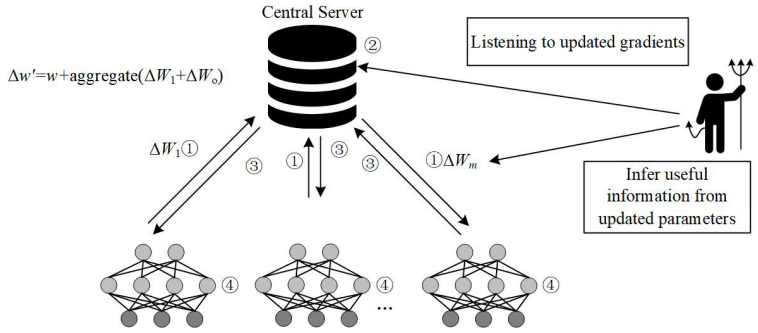
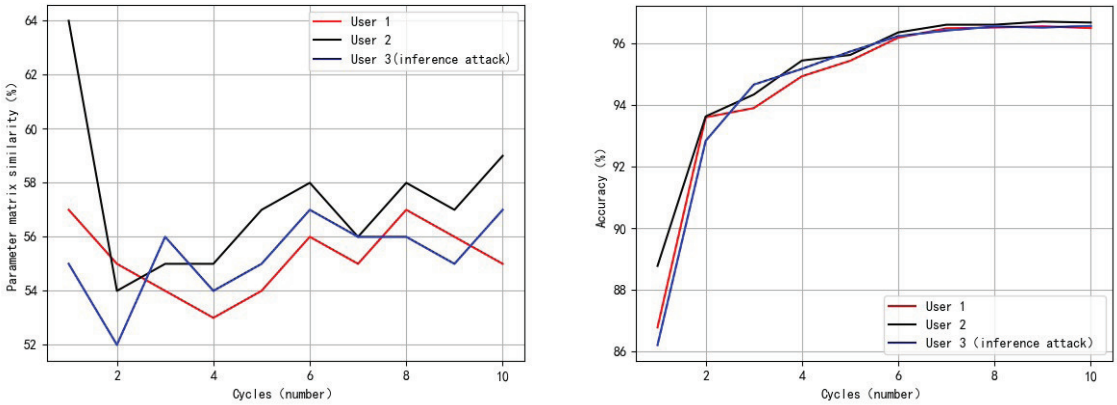


Figure 11. The process of inference attacks.

The results are illustrated in Figure 12. Within the federated-learning framework, attackers can leverage their insights into the current and future environments to acquire valuable data. They can utilize gradient analysis of the training model to better achieve predictive objectives. Therefore, in inference attacks, malicious users can train with normal

data and only need to recover the original data of clients from the global model to obtain data from other users. As a result, the overall model accuracy of the server attacked using this method remains unchanged, and the model can be used normally.



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

Figure 12. Model training with a user that conducted an inference attack. (**a**) The similarity of the parameter matrices. (**b**) The model accuracy.

Although inference attacks can retrieve original data, the federated-learning framework constructed in this study utilizes differential privacy for data transmission encryption. By sacrificing a certain level of data accuracy, it provides strict privacy protection for user data. Adding appropriate noise to the output of statistical queries makes it nearly impossible for attackers to distinguish between statistical differences in two adjacent datasets, thus maximizing the privacy of the user data.

The comparison of the results from the five experiments is shown in Table 4. It can be observed that the model average accuracy was above 96%, and the poisoning attacks, model attacks, and inference attacks had minimal impacts on the security of the user data. The experimental results indicate that the method of model aggregation based on the parameter similarity could effectively mitigate the harm caused by the poisoning attacks and model attacks. Additionally, this study utilized the encryption method of differential privacy to encrypt the model data during transmission, effectively addressing the issue of inference attacks stealing privacy from other users.

Table 4. Experimental results comparison.

Experiment	User Type or Attack Method	Model Average Accuracy	Impact on Model	Data Leakage
1st experiment	Normal user	96.56%	No	No
2nd experiment	Unreliable user	96.23%	No	No
3rd experiment	Poisoning attack	96.56%	No	No
4th experiment	Model attack	96.57%	No	No
5th experiment	Inference attack	96.56%	No	No

5. Conclusions

With the rapid development of new generation artificial intelligence technologies represented by deep learning, crowd-sensing computing to support artificial intelligence model training is becoming increasingly popular and rapidly evolving. Crowd-sensing applications, which utilize a large number of mobile terminal devices to establish an open and integrated system capable of processing large amounts of data and information, can easily lead to the leakage of user privacy data, thus posing significant challenges to data security protection in crowd sensing. This paper proposes FedCrow, which is a data security protection method for crowd sensing based on federated learning. FedCrow addresses

issues such as malicious user attacks and leakage risks during model data interaction in traditional crowd-sensing applications' federated-learning systems. By constructing an efficient and secure federated-learning framework, FedCrow proposes a crowd-sensing federated-learning data security protection method based on the user contribution level, which evaluates the trustworthiness of a client based on its contribution to the overall model and effectively filters legitimate users, thereby reducing the risk of user privacy data leakage caused by malicious users. Finally, the system's ability to resist malicious user attacks was verified through various attack experiments. Additionally, in order to effectively protect the security of the model data during the interaction process, comparative experiments clearly indicated that reasonable data encryption methods should be adopted for data transmission encryption based on the size of the data in crowd-sensing application scenarios.

Author Contributions: Conceptualization, Y.Y. and J.M.; methodology, Y.Y., J.X. and J.M.; software, L.C.; validation, L.C., J.X. and Y.Y.; formal analysis, Y.Y.; investigation, J.M.; resources, J.M.; data curation, L.C. and J.X.; writing—original draft preparation, Y.Y. and J.X.; writing—review and editing, Y.Y. and J.M.; visualization, L.C.; supervision, J.M. All authors read and agreed to the published version of this manuscript.

Funding: This research work received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: We used the publicly available dataset provided by Kaggle. This dataset is freely available at https://www.kaggle.com/competitions/nus-fintech-recruitment/data, Accessed on 23 August 2022. All other implementation and verification data are included in the article.

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

References

1. Liu, Y.; Kong, L.; Chen, G. Dataoriented mobile crowdsensing: A comprehensive survey. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 2849–2885. [CrossRef]
2. Howe, J. The rise of crowdsourcing. *Wired Mag.* **2006**, *14*, 176–183.
3. Yu, Z.; Ma, H.; Guo, B.; Yang, Z. Crowdsensing 2.0. *Commun. ACM* **2021**, *64*, 76–80. [CrossRef]
4. Liu, J.; Shen, H.; Narman, H.S.; Chung, W.; Lin, Z. A survey of mobile crowdsensing techniques: A critical component for the internet of things. *ACM Trans. Cyber-Phys. Syst.* **2018**, *2*, 1–26.
5. Corrigan-Gibbs, H.; Boneh, D. Prio:Private, Robust, and Scalable Computation of Aggregate Statistics. In Proceedings of the 14th USENIX Symposium on Networked Systems Design and Implementation, Boston, MA, USA, 27–29 March 2017; pp. 259–282.
6. Xiong, J.; Ma, R.; Chen, L.; Tian, Y.; Li, Q.; Liu, X.; Yao, Z. A personalized privacy protection framework for mobile crowdsensing in IIoT. *IEEE Trans. Ind. Inform.* **2020**, *16*, 4231–4241. [CrossRef]
7. McMahan, B.; Ramage, D.; Scientists, R. Federated Learning: Collaborative Machine Learning without Centralized Training Data. Google Research. 2017. Available online: https://ai.googleblog.com/2017/04/federated-learning-collaborative.html (accessed on 8 October 2021).
8. Yang, Q.; Liu, Y.; Chen, T.J.; Tong, Y.X. Federated machine learning: Concept and applications. *ACM Trans. Intell. Syst. Technol.* **2019**, *10*, 12. [CrossRef]
9. 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, Fort Lauderdale, FL, USA, 20–22 April 2017; pp. 1273–1282.
10. Al Rubaie, M.; Chang, J.M. Privacy-preserving machine learning: Threats and solutions. *IEEE Secur. Priv.* **2019**, *17*, 49–58. [CrossRef]
11. Geyer, R.C.; Klein, T.; Nabi, M. Differentially private federated learning: A client level perspective[EB/OL]. *arXiv* **2017**, arXiv:1712.07557.
12. Triastcyn, A.; Faltings, B. Federated learning with bayesian differential privacy. In Proceedings of the 2019 IEEE International Conference on Big Data (Big Data), Los Angeles, CA, USA, 9–12 December 2019; pp. 2587–2596.
13. Zhu, H.; Wang, R.; Jin, Y.; Liang, K.; Ning, J. Distributed Additive Encryption and Quantization for Privacy Preserving Federated Deep Learning. *Neurocomputing* **2020**, *463*, 309–327. [CrossRef]
14. Rasheed I. Enhanced privacy preserving and truth discovery method for 5G and beyond vehicle crowd sensing systems. *Veh. Commun.* **2021**, *32*, 100395. [CrossRef]

15. Lv, C.; Wang, T.; Wang, C.; Chen, F.; Zhao, C. ESPPTD: An efficient slicing-based privacy-preserving truth discovery in mobile crowd sensing. *Knowl. Based Syst.* **2021**, *229*, 107349. [CrossRef]
16. Nkenyereye, L.; Islam, S.R.; Bilal, M.; Abdullah-Al-Wadud, M.; Alamri, A.; Nayyar, A. Secure crowd-sensing protocol for fog-based vehicular cloud. *Future Gener. Comput. Syst.* **2021**, *120*, 61–75. [CrossRef]
17. Qiu, F.; Wu, F.; Chen, G. Privacy and Quality Preserving Multimedia Data Aggregation for Participatory Sensing Systems. *IEEE Trans. Mob. Comput.* **2015**, *14*, 1287–1300. [CrossRef]
18. Rahaman, S.; Cheng, L.; Yao, D.D.; Li, H.; Park, J.-M.J. Provably secure anonymous-yet-accountable crowdsensing withscalable sublinear revocation. *Proc. Priv. Enhancingtechnol.* **2017**, *2017*, 384–403.
19. Sucasas, V.; Mantas, G.; Bastos, J.; Damião, F.; Rodriguez, J. A Signature Scheme with Unlinkable-yet-Accountable Pseudonymity for Privacy-Preserving Crowdsensing. *IEEE Trans. Mob. Comput.* **2020**, *19*, 752–768. [CrossRef]
20. Ni, J.; Zhang, K.; Xia, Q.; Lin, X.; Shen, X.S. Enabling Strong Privacy Preservation and Accurate Task Allocation for Mobile Crowdsensing. *IEEE Trans. Mob. Comput.* **2019**, *19*, 1317–1331. [CrossRef]
21. Zhao, B.; Tang, S.; Liu, X.; Zhang, X. PACE: Privacy-Preserving and Quality-Aware Incentive Mechanism for Mobile Crowdsensing. *IEEE Trans. Mob. Comput.* **2020**, *20*, 1924–1939. [CrossRef]
22. Sun, P.; Wang, Z.; Wu, L.; Feng, Y.; Pang, X.; Qi, H.; Wang, Z. Towards Personalized Privacy-Preserving Incentive for Truth Discovery in Mobile Crowdsensing Systems. *IEEE Trans. Mob. Comput.* **2020**, *21*, 352–365. [CrossRef]
23. Panah, A.S.; Yavari, A.; van Schyndel, R.; Georgakopoulos, D.; Yi, X. Context-Driven Granular Disclosure Control for Internet of Things Applications. *IEEE Trans. Big Data* **2019**, *5*, 408–422. [CrossRef]
24. Xie, W.; Wang, Y.; Boker, S.M.; Brown, D.E. PrivLogit: Efficient Privacy-preserving Logistic Regression by Tailoring Numerical Optimizers. *arXiv* **2016**, arXiv:1611.01170. https://doi.org/10.48550/arXiv.1611.01170.
25. Hardy, S.; Henecka, W.; Ivey-Law, H.; Nock, R.; Patrini, G.; Smith, G.; Thorne, B. Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. *arXiv* **2017**, arXiv:1711.10677. https://doi.org/10.48550/arXiv.1711.10677.
26. Wu, X.; Zhang, Y.; Shi, M.; Li, P.; Li, R.; Xiong, N.N. An adaptive federated learning scheme with differential privacy preserving. *Future Gener. Comput. Syst.* **2022**, *127*, 362–372. [CrossRef]
27. Miao, Q.; Lin, H.; Wang, X.; Hassan, M. Federated deep reinforcement learning based secure data sharing for Internet of Things. *Comput. Netw.* **2021**, *197*, 108327. [CrossRef]
28. Jiang, C.; Xu, C.; Zhang, Y. PFLM: Privacy-preserving federated learning with membership proof. *Inf. Sci.* **2021**, *576*, 288–311. [CrossRef]
29. Hijazi, N.M.; Aloqaily, M.; Guizani, M.; Ouni, B.; Karray, F. Secure Federated Learning with Fully Homomorphic Encryption for IoT Communications. *IEEE Internet Things J.* **2024**, *11*, 4289–4300. [CrossRef]
30. Zhang, Z.; He, S.; Chen, J.; Zhang, J. REAP: An Efficient Incentive Mechanism for Reconciling Aggregation Accuracy and Individual Privacy in Crowdsensing. *IEEE Trans. Inf. Forensics Secur.* **2018**, *13*, 2995–3007. [CrossRef]
31. Yang, L.; Zhang, M.; He, S.; Li, M.; Zhang, J. CrowdEmpowered PrivacyPreserving Data Aggregation for Mobile Crowdsensing. In Proceedings of the Eighteenth ACM International Symposium on Mo bile Ad Hoc Networking and Computing, Los Angeles, CA, USA, 26–29 June 2018; pp. 151–160.
32. Zhang, T.; Zhu, Q. Dynamic Differential Privacy for ADMM-Based Distributed Classification Learning. *IEEE Trans. Inf. Forensics Secur.* **2017**, *12*, 172–187. [CrossRef]
33. Phong, L.T.; Aono, Y.; Hayashi, T.; Wang, L.; Moriai, S. Privacy-Preserving Deep Learning via Additively Homomorphic Encryption. *IEEE Trans. Inf. Forensics Secur.* **2018**, *13*, 1333–1345. [CrossRef]
34. Li, Q.; Cao, G.; Porta, T.F.L. Efficient and PrivacyAware Data Aggregation in Mobile Sensing. *IEEE Trans. Dependable Secur. Comput.* **2014**, *11*, 115–129. [CrossRef]
35. Lyu, L.; Yu, H.; Yang, Q. Threats to federated learning: A survey. *arXiv* **2020**, arXiv:2003.02133.
36. Yuan, X.; He, P.; Zhu, Q.; Li, X. Adversarial examples: Attacks and defendese for deep learning. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *30*, 2805–2824. [CrossRef] [PubMed]
37. Yang, C.L.; Chen, G.H.; Xie, S.L. Gradient Information Based Image Quality Accessment. *Acta Electonica Sin.* **2007**, *35*, 1313–1317.
38. AbdulRahman, S.; Tout, H.; Ould-Slimane, H.; Mourad, A.; Talhi, C.; Guizani, M. A survey on federated learning: The journey from centralized to distributed on-site learning and beyond. *IEEE Internet Things J.* **2020**, *8*, 5476–5497. [CrossRef]
39. 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, Online, 26–28 August 2020; pp. 2938–2948.
40. Melis, L.; Song, C.; Cristofaro, E.D.; Shmatikov, V. Exploiting Unintended Feature Leakage in Collaborative Learning. In Proceedings of the 2019 IEEE Symposium on Security and Privacy (SP), Francisco, CA, USA, 19–23 May 2019. [CrossRef]
41. Rao, B.; Zhang, J.; Wu, D.; Zhu, C.; Sun, X.; Chen, B. Privacy Inference Attack and Defense in Centralized and Federated Learning: A Comprehensive Survey. *IEEE Trans. Artif. Intell.* **2024**, 1–22. [CrossRef]