FedCIP: Federated Client Intellectual Property Protection with Traitor Tracking

Junchuan Liang
Rong Wang*
liangjunchuan@stu.sicnu.edu.cn
rwang@sicnu.edu.cn
School of Computer Science, Sichuan Normal University
Chengdu, China

ABSTRACT

Federated learning is an emerging privacy-preserving distributed machine learning that enables multiple parties to collaboratively learn a shared model while keeping each party's data private. However, federated learning faces two main problems: semi-honest server privacy inference attacks and malicious client-side model theft. To address privacy inference attacks, parameter-based encrypted federated learning secure aggregation can be used. To address model theft, a watermark-based intellectual property protection scheme can verify model ownership. Although watermarkbased intellectual property protection schemes can help verify model ownership, they are not sufficient to address the issue of continuous model theft by uncaught malicious clients in federated learning. Existing IP protection schemes that have the ability to track traitors are also not compatible with federated learning security aggregation. Thus, in this paper, we propose Federated Client-side Intellectual Property Protection (FedCIP), which is compatible with federated learning security aggregation and has the ability to track traitors. To the best of our knowledge, this is the first IP protection scheme in federated learning that is compatible with secure aggregation and has tracking capabilities.

CCS CONCEPTS

 \bullet Security and privacy \to Domain-specific security and privacy architectures.

KEYWORDS

Federated learning; intellectual property protection; machine learning security; traitor tracking

1 INTRODUCTION

Federated learning (FL) [22] has emerged as a popular framework for training deep learning models without sharing private data, with applications in medical image analysis [6, 27], recommendation systems [32], and object detection [21]. Federated learning provides the ability to protect data privacy while still utilizing large amounts of private data to generate accurate and reliable models. However, federated learning also poses risks such as privacy inference attacks [25, 33, 37, 39] and model theft [20]. In privacy inference attacks, attackers are often honest but curious servers or malicious attackers with direct access to client update parameters. In model theft, attackers are often malicious clients who compromise the security of the federated learning system, posing a significant threat to the privacy-preserving features and outcomes of federated learning.

To mitigate the risk of privacy inference attacks, various secure aggregation methods for federated learning have been proposed, including parameter encryption techniques [23] and differential privacy [2, 10, 34]. Parameter encryption allows the server to perform aggregation computation while keeping the real parameters inaccessible, effectively protecting model parameters from disclosure. Differential privacy involves adding a certain amount of noise to the model parameters according to the differential privacy budget, making the original data more difficult to infer. The core of all these approaches is to prevent the server from accessing the complete real model parameter information, ensuring that the server cannot perform any operations (such as embedding backdoors to the global model) on the model parameters other than necessary computation.

To combat model stealing attacks in federated learning, watermarking [3, 4, 19, 24, 28, 41] techniques have been proposed as an effective solution. These techniques can be divided into two categories: client-side watermarking [16, 36] and server-side watermarking [17, 26, 30]. Client-side watermarking involves injecting a watermark into the model during the local training process of the client, while server-side watermarking involves embedding a slightly different watermark into each model distributed to the client at each model aggregation. Both approaches effectively verify the ownership of suspicious models. But in federated learning, it is not enough to just verify ownership. This is because traitors are able to consistently steal models if they are not caught. However, existing methods that can track traitors are not compatible with federated learning security aggregation. Methods that are compatible with federated learning security aggregation again do not have tracking capabilities. A solution with both capabilities means that the server distributes the same model to each client and then finds who stealing the model. Even previous research has suggested that if the server distributes the same model to each participant, this can lead to difficulties in tracking traitors [26].

We propose a novel approach to solve this problem which is called FedCIP (Federated Client Intellectual Property Protection). FedCIP has the ability to track traitors while being compatible with federated learning security aggregation. In our attack scenario, there are two main adversaries: the honest but curious server, and the persistent model stealing client. To be compatible with the Federated Learning Security Aggregation approach, FedCIP servers cannot do anything to the model beyond the necessary calculations. Then, in order to track traitors in the client, FedCIP employs a fast and replaceable client-side watermarking technique along with client selection to achieve this.

FedCIP first utilizes replaceable client-side watermarking techniques to update federated learning models at each round or several rounds, allowing for unique watermarks in each model. By comparing these watermarks, we can identify the rounds in which the model was stolen. The server selection aggregation of federated learning is then utilized to select only certain clients who participated in these rounds, which results in a set of suspicious traitors. Finally, multiple sets of potential traitors obtained from suspicious models are examined to identify the same client among them, who is the traitor. FedCIP only requires the client to embed the watermark, and the server does not make any additional modifications to the model, making it compatible with federated learning security aggregation methods such as homomorphic encryption. Furthermore, it has the ability to track traitors who continuously steal the model on the client side.

Moreover, FedCIP also has the fundamental capabilities of an IP protection scheme. These capabilities include ownership verification, resistance to pruning attacks, resistance to fine-tuning attacks, etc. We compare our program (FedCIP) with the existing Federated Learning IP protection program in Table 1.

The main contributions of this article are the following three points.

- We have introduced the FedCIP framework, which is a novel approach for protecting the intellectual property of federated learning clients, capable of tracking traitors while compatible with federated learning security aggregation.
- To facilitate the implementation of FedCIP, we have proposed a Fast Federated Watermarking algorithm (FFWM1, FFWM2), which allows federated learning clients to replace watermarks.
- Experiments demonstrated the effectiveness and robustness of our approach through extensive experiments. Additionally, we have conducted a thorough theoretical analysis of the number of traitors leaking the model using our method and the probability of them being caught.

2 BACKGROUNDS

2.1 Privacy inference attack

A privacy inference attack [9] refers to an attack in which an adversary tries to infer the original data by analyzing multiple rounds of gradient values during the training process in FL. The attacker in a privacy inference attack is usually an honest but curious server or another malicious individual who has access to model updates. Such attacks pose a significant threat to the very essence of federated learning, which is to protect the privacy of participants' data.

Zhu et al. [40] proposed Deep Leakage from Gradients (DLG) algorithm which utilizes leaked gradient information to make inferences about the raw data. Building on this work, Zhao et al. [39] present the iDLG algorithm, which offers improved inference under the same conditions. Additionally, Yin et al. [37] demonstrate high-volume privacy data recovery, while Ren et al. [33] and Wang et al. [25] propose privacy inference using Generative Adversarial Network (GAN) based on client-side gradient information.

2.2 Defenses against privacy inference attack

To defend against privacy inference attacks in federated learning, the main strategies are based on cryptographic principles such as homomorphic encryption [23] and differential privacy [2, 10, 34].

Homomorphic encryption. Homomorphic encryption [23] is a powerful cryptographic technique that allows computations to be performed on encrypted data. This means that the server can perform computations on models while protecting its confidentiality. In federated learning, homomorphic encryption is used to further enhance the privacy protection of participants' model. This approach enables the server to compute while protecting the real model parameters. This greatly reduces the risk of data leakage during the federated learning process.

Differential privacy. Differential privacy [2, 10, 34] is a technique that involves adding noise to data in order to protect the privacy of the data. Unlike indiscriminately adding noise, differential privacy involves adding noise with specific size and distribution requirements. The key characteristic of differential privacy is that it enables the protection of the privacy of the data while maintaining the usability of the data.

2.3 Model theft

Federated learning is vulnerable to free-rider attackers and illegal copy [20]. Due to the need to protect the privacy of client data and processes are not visible to others in the federated learning process. This provides an opportunity for attackers to masquerade as benign participants in federated learning and steal models, and continuously leak them. These attackers also try to disable any potential watermarking in the model through watermarking removal methods such as model fine-tuning [29] and pruning [12, 31]. Model fine-tuning involves the attacker training the model for a certain number of rounds using local data instead of directly leaking the stolen model. By changing the model weight parameters through this fine-tuning method, the potential watermarking can be destroyed while ensuring the model's availability. Model pruning, on the other hand, is a technique used by attackers to prune the unimportant weight nodes in the stolen model to remove the potential watermarking.

2.4 Defenses against model theft

2.4.1 DNN watermarking. Watermarking [3, 4, 24, 28] is a technique in Deep Neural Networks (DNN) to safeguard intellectual property, and can be classified into backdoor-based [1, 11, 19, 38, 41] and parameter-based [5, 7, 8, 18, 31] methods.

Backdoor-based watermarks. The backdoor-based [1, 11, 19, 38, 41] approach was first introduced as an attack method [3, 4, 28], but it is also being used for intellectual property protection [19, 24, 41] due to its ability to directly control the output of the model using specific triggers without modifying the model parameters. This method has a minimal impact on the model's primary task and is thus considered a popular black-box approach [1, 11, 38].

Parameter-based watermarks. One of the approaches to embedding watermarks in deep neural networks is the parameter-based method [5, 7, 8, 18, 31], which is a white-box technique that involves modifying the model parameters to embed the watermarking. This method allows for quick watermarking injection but

Method	Pruning resistant	Fine-tuning resistant	Secure FL	Traitor tracking
Merkle-Sign [17]	Yes	Yes	No	Yes
WAFFLE [30]	Yes	Yes	No	No
FedIPR [16]	Yes	Yes	Yes	No
FedTracker [26]	Yes	Yes	No	Yes
Watermarking in Secure FL [36]	Yes	Yes	Yes	No
FedTracker [26]	Yes	Yes	No	Yes
FedCIP	Yes	Yes	Yes	Yes

Table 1: Comparison of the federated learning IP protection program.

requires direct access to the model parameters. In the parameter-based watermarking approach, the watermarking is represented as an N-bit string and is directly embedded into the model parameters W.

2.4.2 FL watermarking. Relying on the (DNN) intellectual property protection approach is inadequate for the safeguarding of intellectual property in FL models. Firstly, this is because multiple clients participate in FL and pose potential threats to the model's security. Consequently, intellectual property protection in FL must continue throughout the entire training process. Secondly, the model aggregation process in FL may weaken watermarks that lack robustness. Lastly, tracking traitors among participating clients in FL and excluding them is crucial to better safeguard the intellectual property rights of other participants. There are two main areas of research on intellectual property protection in the field of FL.

Client-based method. One of the most widely used methods for protecting intellectual property in federated learning is client-side watermarking [16, 36]. Among these methods, a client-side watermarking scheme that can satisfy federated learning security aggregation was proposed in [16]. This approach allows for ownership verification of leaked models and ensures the robustness of the watermarking. Similarly, Yang et al. [36] proposed a new scheme for protecting intellectual property in federated learning scenarios, assuming the existence of special clients and their watermarking of models, while also allowing for verification of the ownership of leaked models. However, neither of these methods has the capability to track traitors in federated learning. This leads to the issue that the leaker can leak with impunity because they will not be caught.

Server-based method. This approach begins with the server and uses watermarking injection on the model through the server. Tekgul et al. [30] proposed a server backdoor approach, which can generate backdoor triggers without training data and can be effectively triggered by the model to achieve intellectual property protection. Shao et al. [26] proposed a server watermarking method, which can prove the ownership of the model while tracking traitor among the clients through a unique watermarking for each client. A similar approach for model ownership verification and traitor tracking through the client is presented in [17]. In addition, Li et al. [17] proposed a design scheme for recovering the identity of watermarked lost participants. However, these server-side watermarking methods are at risk of privacy inference attacks as they fail to meet the requirements of federated learning security aggregation schemes.

3 THREAT MODEL AND SECURITY REQUIREMENTS

3.1 Threat model

There are two main threat models that we face. The first one is the honest but curious server. The second one is model theft in the client.

The honest but curious server is a server that performs honest computation as requested but is curious about the client's private data. This curiosity may come from the server itself or from a potential attacker hidden in the server. Curious servers can pose a threat to the privacy and security of federated learning.

Model theft masquerades as a benign client to join FL. The model thief behaves like a benign client except for leaking models. And since federated learning is an ongoing, long-term process, multiple global models are generated during the process. The model thief can continuously steal and leaks the global model of federated learning.

3.2 Security requirements

To address the threats mentioned above, we suggest that an intellectual property protection solution in a federated learning setting should satisfy the following two aspects. On the one hand, this solution is able to resist honest and curious servers. On the other hand, it can resist consistent model theft. For the latter, in addition to verifying the ownership of suspicious models, it should be able to track traitors in the client and address model theft at the root. Therefore, we summarize the security requirements of the federated learning IP protection scheme in the following points.

Secure aggregation. The approach should satisfy the Federated Learning Security Agency's requirements. It must ensure that the server cannot directly modify the parameters inside the model.

Traitor tracking. In federated learning, verification of suspicious model ownership alone does not prevent model theft. Therefore, it is necessary to track down traitors who persistently steal models.

Fidelity. The approach should ensure that the model performance does not impact significantly after watermarking.

Robustness. The approach should be resistant to watermarking removal techniques such as model fine-tuning and pruning to maintain ownership of the verification capability.

Reliability. The added watermarking should have a high recognition rate under the verification mechanism and a low recognition rate for non-watermarked models.

Capacity. A reliable watermarking mechanism should have a flexible capacity to bury a varying number of watermarking bits based on the model's size, ensuring a good recognition rate.

Independence. The watermarking generation should not rely on or modify the original training data to ensure the integrity of the original data.

4 FEDCIP FRAMEWORK

4.1 Overall design

Table 2: provides the notation used in this section.

Parametric	Description
α	The watermark embedding power parameter
	when the round is not the first in a cycle
β	The watermark embedding power parameter
	when the round is the first in a cycle
K	Total number of clients participating in FL
C	Coefficient of the number of client aggregations
	selected by the server per round
T_{i}	Watermark update cycle; T_i =1, 2, 3round
η	Purning rate
FWM_i	Watermark embedded by all client in T_i
$FWM_{\mathcal{W}}$	Watermark extracted from the model
FWM_{ik}	Watermark embedded by the kth client in T_i
FWM_{wk}	Watermark extracted from the <i>kth</i> client
M	The model before the watermark was embedded
$ ilde{M}$	The model after being embedded with watermark
W	Weights before being selected by the client to
	embed the watermark
$ ilde{W}$	The weight after being selected by the client to
	embed the watermark
WMdacc	Accuracy of watermark detection
N	Watermark length
N_t	Number of clients that may be traitors

FedCIP is designed to ensure resistance against honest and curious servers while also having the ability to track traitors in the client. To achieve this, FedCIP introduces the concept of a training cycle T_i in the federated learning training process shown in Figure 1. The training cycle T_i is defined as a T_i containing multiple rounds. For example, T_i =3, which means we consider 3 rounds as one cycle. Different cycles embed different watermarks, and the same watermark is used in the same cycle. In the usual federated learning, the server selects different clients in each round to participate in aggregation. However, in FedCIP, the server selects clients in T_i . That is, within a T_i , the clients participating in the federated aggregation are all the same. But different T_i 's participating are different. In particular, when the T_i =1 round, watermark replacement is performed for each round. Then, at each T_i , the model is embedded with the corresponding unique Federated Watermark (FWM_i). The FWM_i is embedded by clients selected by the server to participate in federated aggregation.

In addition, the server will do the work of the region division to prevent client watermark conflicts and logging. In the first model distribution phase, the server will divide the model into K regions to be watermarked according to the number of clients K. For example, the model parameters have a total of 1,000,000 weight parameters, and the number of clients K=10. Then the server will divide all parameters into 10 parameter regions and randomly assign them to K clients. Each client will be given a region with 100,000 parameters to be watermarked and used to embed the watermark. The clients will embed the watermark in their respective regions. This area is mainly divided to prevent watermark conflicts. Secondly, the server will record the number of clients participating in federated learning in each cycle, which is used to verify the model watermark and find the traitor. Moreover, FedCIP utilizes parameter-based watermarking techniques on the client side as the foundation, incorporating server selection and Fast Federated Watermarking.

4.2 Client-side federated watermark embedding

4.2.1 Cycle division. The watermark embedding process of the FedCIP algorithm is divided into two main parts. The first part involves partitioning the training rounds by T_i and then performing the watermark embedding algorithm within T_i . There can be multiple rounds inside each T_i , where the first round use Fast Federated Watermark 1 (FFWM1) and the rest use Fast Federated Watermark 2 (FFWM2).

Algorithm 1 FedCIP watermark embedding algorithm

Input: Total number of training rounds *rounds*, watermark cycle T_i , watermark FWM_i , clients participating in federated learning CK, global model M

```
Output: Watermarked model \tilde{M}.
 1: for round in range(rounds) do
        if round mod T_i == 0 then
 3:
            for k in CK do
 4:
                 local_{model} \leftarrow deepcopy(M);
 5:
                 Train(local_{model});
                 FFWM1:
 6:
            end for
 7:
 8:
        else
            for k in CK do
 9:
                 local_{mode} \leftarrow deepcopy(M);
10:
                {\bf Train}(local_{model});
11:
                 FFWM2:
12:
            end for
13:
        end if
14:
    end for
16: Execute Federated Aggregation Algorithm FedAvg;
17: M \leftarrow \text{Watermark}(M, FWM_i);
18: return M
```

Algorithm 1 is used for cycle division in FedCIP. The algorithm randomly selects a certain percentage of clients to participate in the current T_i for watermark embedding. The watermark is then embedded in the selected clients' models using the Fast Federated Watermark algorithm. After T_i rounds of federated learning, the watermark is replaced with a new one. The new watermark replaces the previous one completely and is embedded in the models of the selected clients for the next T_{i+1} cycle. This process continues until

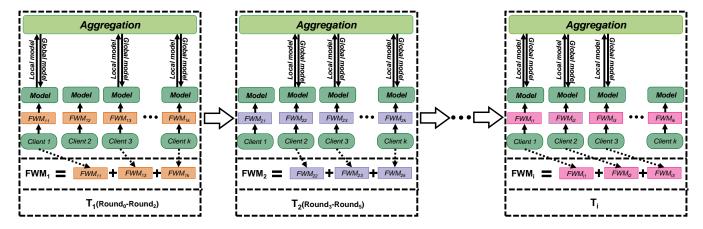


Figure 1: FedCIP watermark embedding training process.

the end of the training process. It is important to note that in each T_i cycle, the same watermark is embedded in the models of all the selected clients. However, the watermark is different in each T_i cycle.

The algorithm operates in multiple training rounds, with the watermark applied at specific intervals defined by the watermark cycle T_i . The input parameters for the algorithm include the total number of training rounds, watermark cycle T_i , watermark FWM_i , participating clients CK, and the initial global model M.

After the above process, in each cycle, T_i , the global model of federated learning is embedded in the corresponding watermark FWM_i

4.2.2 FFWM1. The FFWM1 algorithm is executed at only the first round of each T_i . The purpose of FFWM1 is to quickly replace the original watermark with a new one. the FFWM1 algorithm is as follows.

$$\tilde{W} = W + \beta \times (FWM_{ik} - FWM_{Wk}) \tag{1}$$

$$FWM_{Wk} = \begin{cases} 1 & (if & W > 0) \\ -1 & (if & W < 0) \end{cases}$$
 (2)

 \tilde{W} is the weight after the client embeds the watermark; W is the weight before the watermark is embedded; β is the watermark embedding power; FWM_{ik} is the watermark that will be embedded in that cycle by kth client. FWM_{Wk} is the watermark extracted from the weight before the watermark is embedded in the kth client model. FWM_{Wk} is calculated as shown in Equation 2. The algorithm takes as input the weight vector W used by the client to embed the watermark, which can be considered as a vector, and the watermark FWM_{Wk} used in the current cycle, which is a binary array of equal length to W containing only 1 and -1. The output is the weight vector \tilde{W} after embedding the watermark.

The algorithm first initializes the watermark FWM_{Wk} for the initial weight vector W. If the weight vector W is greater than 0, the corresponding element of FWM_{Wk} is set to 1, otherwise it is set to -1. Then, the algorithm calculates the weight vector \tilde{W} using the

formula $\tilde{W} = W + \beta \times (FWM_{ik} - FWM_{Wk})$, where β is the power factor of the embedded watermark.

Note that the position of W is selected by the client during the first training, and the position remains unchanged thereafter. Furthermore, the binary array FWM_W is used to determine which weights need to be replaced with the new watermark. Only those weights that are different from the previous watermark need to be replaced, as this can reduce the changes to the model.

4.2.3 FFWM2. Algorithm FFWM2 is used for embedding water-marks in non-first rounds when T_i is greater than 1. The FFWM2 algorithm is as follows.

$$\tilde{W} = \alpha \times FWM_{ik} \tag{3}$$

Where \tilde{W} is the weight after embedding the watermark. α is the watermark strength factor, and FWM_{ik} is the current cycle watermark.

Two reasons exist for using a different watermarking method than the first round. First, the first round requires rapid experimental watermark replacement, so the change in parameters is relatively large. However, if the case where T_i is greater than 1, the subsequent rounds do not need a large number of parameter changes. Second, for the first round, excessive or small parameter changes can be constrained to the FWM_{ik} neighborhood, making the watermark more robust.

4.3 Ownership verification

4.3.1 Federated watermark. The ownership of the model is verified using the federated watermark. The federated watermark is the sum of all the watermarks of the participating aggregation clients in the T_i cycle. It is defined as follows.

$$FWM_i = Concat[FWM_{ik}(k = 1, 2, 3, ..., k \in CK)]$$
 (4)

 FWM_i is the federated watermark of T_i cycle and FWM_{ik} is the kth client's watermark. For example, in the T_i cycle, the server selects a total of 8 clients to participate in the federated learning aggregation, and the length of each client watermark is 10 bits, so the federated watermark is 80 bits.

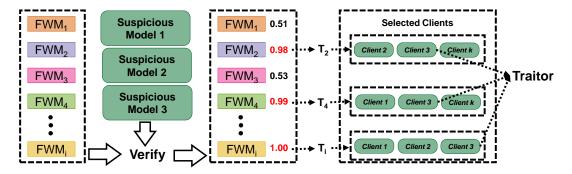


Figure 2: Watermark verification and traitor tracking.

4.3.2 Suspicious model watermark extraction. Next, we will extract the watermark from the suspicious model by determining the corresponding weight values based on the FWM_w calculated in Equation 5. The segmentation threshold value of 0 is utilized during the extraction process. The extraction method can be described as follows.

$$FWM_{w} = \begin{cases} 1 & (if & W > 0) \\ -1 & (if & W < 0) \end{cases}$$
 (5)

 FWM_{w} is the watermark extracted from the suspicious model.

4.3.3 Watermark detection. We use the Hamming distance as a measure for watermark comparison. The Hamming distance can be used to measure the difference between two binary arrays. We define the watermark detection accuracy as *WMdacc* and the expression is as follows.

$$WMdacc = 1 - 1/N \times \mathcal{H}(FWM_i, FWM_w) \quad (N \ge 1)$$
 (6)

 \mathcal{H} is the Hamming distance, FWM_i is the federated watermark, and FWM_w is the suspicious model extracted watermark, N is the length of the federated watermark.

4.4 Traitor tracking

The core intuition of traitor tracking is that if a client skips training in a cycle, their local model will lack the watermark generated by the participating clients in that cycle's aggregation. Each federated watermark is unique to the cycle, allowing us to identify which cycle a leaked model originated from by analyzing its watermarks. From there, we can obtain the set of potential traitors for that specific cycle. Finally, multiple sets of potential traitors were obtained from suspicious models and the same client in all sets is the traitor as Figure 2.

We define the server's selection rate of clients per cycle as C, which means that CK clients are selected per cycle (where K is the total number of participating clients in federated learning). Assuming that the traitor leaks the model n times, we can obtain n set C_p of potential traitors, each of which contains the traitor. The problem we need to solve is determining the minimum number of leaks required before we can identify the traitor in C_p . We assume that the expected number of potential traitors in the set C_{p1} obtained from each leak is N_t . If there is only one leak, the expected

number of potential traitors is simply the number of clients in C_{p1} . However, if there is a second leak and we obtain a second set of potential traitors C_{p2} , the expected number of potential traitors becomes the number of $(C_{p1} \cap C_{p2})$. We can uniquely identify the traitor when this expected number N_t is less than one. Therefore, the required number of leaks is

$$N_t = K \times C^n \le 1 \tag{7}$$

In this scenario, K represents the total number of clients participating in federated learning, while C denotes the server's selection rate. The traitor leaks the model n times. For example, if there are 10 clients participating in federated learning with a server selection rate of 0.5 (meaning models from 5 clients are aggregated each time), then the traitor will be caught on average after stealing the model 4 times. From the traitor's perspective, the probability of being caught when leaking the model for the first time is 0.5, increasing to 0.75 for the second leak and 0.875 for the third leak. By using replaceable watermarking (which does not require additional storage) and logging the clients involved in each cycle, our approach can identify traitors with a higher risk of being caught when stealing the model.

5 EXPERIMENTS

5.1 Evaluation metrics

We use the following two evaluation metrics to assess our approach. **Model accuracy**. The accuracy of the network model on the test set during training. It reflects the basic performance of the model and will be used as a measure of model performance.

Watermark detection accuracy. Watermark detection accuracy (WMdacc) is defined in detail in section 4.3.3 of the paper. The WMdacc measures the difference between the watermark before the client embedding and the watermark extracted from the suspicious model. A value of 1 indicates that the two are identical, and this metric will be used to evaluate the effectiveness of watermark detection.

5.2 Experiment setting

5.2.1 Dataset. We use the following three data sets for our experiments.

MNIST. The MNIST [15] consists of 70,000 grayscale images of handwritten digits, from 0 to 9. Each image is 28x28 pixels in

size, and the intensity of each pixel ranges from 0 (white) to 255 (black). The dataset is split into two parts: 60,000 training images and 10,000 test images. This division allows researchers and practitioners to train their algorithms on the training set and evaluate their performance on the test set.

FashionMNIST. FashionMNIST [35] is a dataset of Zalando's article images. The dataset consists of 70,000 grayscale images, each with a resolution of 28x28 pixels. There are 10 different classes of fashion items, with each class having 7,000 images. The 10 classes are T-shirts/tops, Trousers, and Pullovers... FashionMNIST is divided into two parts: a training set of 60,000 images, used for training machine learning models, and a test set of 10,000 images, used for evaluating the performance of trained models.

CIFAR-10. The CIFAR-10 [14] dataset (Canadian Institute For Advanced Research) is a widely used dataset for image classification in the field of machine learning and computer vision. The dataset consists of 60,000 32x32 color images, divided into 10 classes such as Airplane, Automobile, Bird, and so on. Each class contains 6,000 images, and the dataset has a total of 50,000 training images and 10,000 test images.

The above three datasets are used in experiments using Independent Identical Distribution (IID) and Non-Independent Identical Distribution (non-IID). The non-IID setting uses label-based non-IID. i.e., the training data is labeled incompletely for each client. In our non-IID setting, each client has only random data with 2-5 labels.

5.2.2 Network model. We used the following three network models for our experiments.

MLP. MLP stands for multilayer perceptron, which is a neural network model used in supervised learning tasks. MLP is a type of feedforward neural network, which means that information flows through the network in one direction, from the input layer to the output layer, without any feedback loops. Our model has 2 hidden layers, each with 200 units and ReLU activation functions. The input layer has 784 nodes (28x28 pixels) corresponding to the size of the images in the dataset, and the output layer has 10 nodes corresponding to the 10 possible digits (0-9) that can be present in the images. This model has a total of 199,210 trainable parameters, which includes the weights and biases of the 3 fully connected layers.

CNN. The model consists of two convolutional layers (conv1 and conv2) with kernel sizes of 5x5, followed by max-pooling layers (pool) with a kernel size of 2x2. The convolutional layers extract features from the input images, and the max pooling layers reduce the size of the feature maps to increase computational efficiency. The ReLU activation function is applied to the output of each convolutional layer. This model has a total of 1,668,426 trainable parameters, which include the weights and biases of the convolutional layers, the weights and biases of the fully connected layer, and the weights and biases of the output layer.

ResNet18. The ResNet18 [13] architecture consists of 18 layers and uses residual connections to allow for deeper networks to be trained without encountering the vanishing gradient problem. The model is divided into four groups of layers, each consisting of multiple BasicBlock layers, which are defined in the code above. Each BasicBlock consists of two convolutional layers followed by

batch normalization and a ReLU activation function and a shortcut connection that allows the input to bypass the convolutional layers. The ResNet18 architecture has a total of 11,173,962 trainable parameters, which include the weights and biases of the convolutional layers and fully connected layers.

5.2.3 Federated learning settings. The federated learning aggregation method in this paper uses federated average (FedAvg) [22]. The default total number of clients participating in federated learning is 10. The server selection rate in each round is 0.5. i.e., the server selects 5 clients for model aggregation and watermark embedding in each round. The learning rate is 0.01. The number of local training rounds per client is 2.

5.3 Experimental results

5.3.1 Fidelity. To evaluate the fidelity of FedCIP, we conducted experiments on three different models and datasets, namely MLP, CNN, and ResNet, using both iid and non-iid data distributions. We compared the accuracy of the global model with and without watermarking. The results are shown in Figure 3, where the accuracy of each model with different watermark lengths on the same dataset is displayed separately.

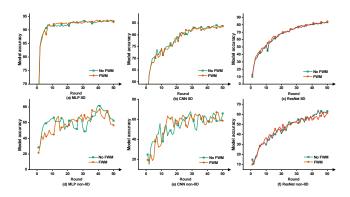


Figure 3: Impact of federated watermarking on model accuracy.

As depicted in Figure 3, we conducted experiments to test the effect of having or not having watermarks on model accuracy for different network structures on IID and non-IID datasets. The accuracy rates of MLP, CNN, and ResNet18 with and without watermark on the IID dataset are represented by a, b, and c, while d, e, and f represent the accuracies of MLP, CNN, and ResNet18 with and without watermark on the non-IID dataset. The horizontal axis in the figure shows the number of training rounds, while the vertical axis represents the accuracy rate. The green dash represents the accuracy without watermarks, while the orange line indicates the accuracy with watermarks. A comprehensive analysis of the experiments depicted in Figure 3 reveals that the presence of a watermark has little impact on the accuracy of the model. Therefore, FedCIP has a negligible effect on the performance of the federated learning model.

5.3.2 Reliability. To evaluate the reliability of the watermark detection in FedCIP, we conducted experiments varying three parameters

related to watermark embedding T_i , α , and β . T_i mainly affects the rapid replacement of the watermark in every round when T_i =1, which can impact the watermark embedding. The experiments aimed to examine the effect of different values of T_i , α , and β on the watermark detection rate. Specifically, we measured the watermark detection accuracy, which reflects the similarity between the watermark before the embedding and the watermark extracted from the suspicious model.

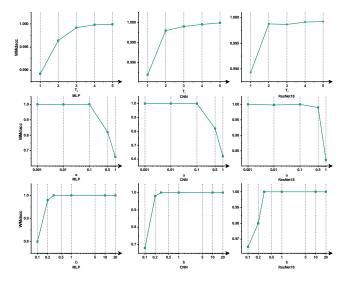


Figure 4: The effect of different watermark cycle T_i , watermark power α , and β on the watermark detection accuracy.

The results of the reliability of the watermark are presented in Figure 4. The vertical coordinate represents the watermark detection accuracy in the average of 50 detections. The first row of the figure shows the effect of T_i on the watermark detection accuracy. It can be seen that the watermark detection accuracy remains high (99%) even when T_i is set to 1, meaning that the watermark is rapidly replaced in each round. As T_i increases, the watermark detection accuracy approaches 100%.

The second row of Figure 4 shows the impact of the watermark strength parameter α on the watermark detection accuracy. The results indicate that a value of α between 0.001 and 0.1 produces 100% watermark detection accuracy. However, when α increases to around 0.5, the watermark detection accuracy rapidly decreases. This decrease occurs because large α values cause the weights to become too large, which in turn reduces the success rate during the next watermark replacement.

The third row of Figure 4 shows the impact of the watermark strength parameter β on the watermark detection accuracy. Unlike α , smaller β values (e.g., 0.1) result in a lower success rate for watermark detection. As β increases (between 0.5 and 20), the watermark detection accuracy increases to 100%. The value of β affects the first round in T_i , during which the watermark is replaced with a new one. A smaller β value means that the new watermark is weaker, resulting in a lower success rate for watermark replacement.

5.3.3 Capacity. In this section, we will test the length of the embeddable watermark. We will keep increasing the length of the

watermark and test its effect on the accuracy of the model for the same number of rounds.

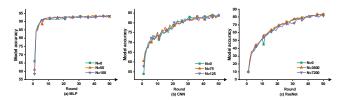


Figure 5: The effect of different lengths of watermarks on the model accuracy.

Figure 5 shows the effect of different watermark lengths on the accuracy of the model. N is the length of the watermark. From the figure, it can be seen that increasing the length of the watermark within a certain range does not significantly affect the accuracy of the model.

5.3.4 Robustness. A good watermarking scheme can make the watermark still detectable even when subjected to watermark removal attacks such as pruning [29] and fine-tuning [12, 31]. Pruning refers to the removal of unimportant parameters from the model by the model stealer after stealing the model. This approach is able to corrupt the backdoor hidden in inconspicuous parameters while keeping the model usable. Specifically, after getting the model, the thief sorts the model weights in absolute magnitude and sets the weights near 0 to 0. Fine-tuning is where the model thief continues to train the model using the same dataset as the model training features. This method is able to modify the weights inside the model while ensuring that the model is available for the purpose of corrupting the potential watermark.

We use these two methods separately to attack our watermarking scheme with the following results.

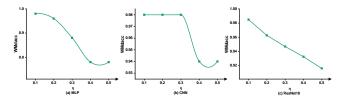


Figure 6: The effect of different purning rate η on the watermark detection accuracy.

As shown in Figure 6, we evaluated the resistance of FedCIP against pruning attacks. The horizontal axis represents the pruning coefficient, defined as the proportion η of the weight value near 0 in the model that is set to 0. The vertical axis represents the watermark detection accuracy. The results show that the impact on watermark detection accuracy is within 5% for a pruning coefficient no greater than 0.5. However, it is important to note that the pruning coefficient should not be set too large, as it can negatively affect the accuracy of the model. These results demonstrate that FedCIP is able to maintain its watermark detection accuracy even in the face of pruning attacks, further validating its robustness against attacks on model weights.

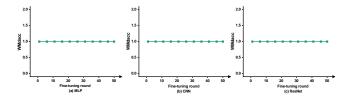


Figure 7: The effect of different fine-tuning rounds on the watermark detection accuracy.

The fine-tuning experiment aims to test the effectiveness of FedCIP against fine-tuning attacks. The results are shown in Figure 7, where the horizontal coordinate represents the number of fine-tuning rounds and the vertical coordinate represents the watermark detection accuracy. The experiment shows that even after 50 rounds of fine-tuning, the watermark detection accuracy remains at 100%. This indicates that FedCIP is resistant to fine-tuning attacks and can effectively protect the intellectual property of the federated learning model.

6 DISCUSSION

The approach proposed in this paper is a novel approach to federated learning as it addresses the issue of tracking traitors in a privacy-preserving manner. The ability to track traitors in federated learning is crucial in ensuring the security of the learning process, and FedCIP achieves this goal while being compatible with parametric encryption schemes. Although the method of tracking the traitor in this paper depends on the number of times the traitor itself stole the model, it increases the cost of the traitor compared to previous approaches that do not have the ability to track the traitor. This means that the approach in this paper puts pressure on the traitor, making it more difficult for them to steal models with impunity.

However, the parameters used in the FedCIP approach, α , and β , are conventional parameters and not optimized for the model. For future work, we can explore designing better parameter schemes that have less impact on the model and can enhance the capacity of the watermarking technique.

Overall, the FedCIP approach presented in this paper is a step forward in addressing the challenges faced by federated learning systems in terms of security and privacy. It provides a practical and effective way to track traitors, ensuring the security of the federated learning process.

7 CONCLUSIONS

In this paper, we explore the challenges of protecting intellectual property rights in federated learning and review existing research on watermarking techniques from both the client and server sides. We identify the limitations of each approach and propose a new client-side watermarking approach called FedCIP that allows for traitor tracking while maintaining compatibility with federated learning security aggregation such as parameter encryption. We provide a detailed description of the design and implementation of FedCIP and evaluate its performance in terms of model accuracy and watermark detection accuracy using three different models and datasets. Our results demonstrate that FedCIP is effective in

protecting intellectual property in federated learning while also adding pressure on traitors to reduce the risk of IP theft. Future work could focus on optimizing the parameter schemes for FedCIP to minimize its impact on model performance.

REFERENCES

- Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. 2018. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In 27th USENIX Security Symposium (USENIX Security 18). 1615–1631.
- [2] Naman Agarwal, Peter Kairouz, and Ziyu Liu. 2021. The skellam mechanism for differentially private federated learning. Advances in Neural Information Processing Systems 34 (2021), 5052–5064.
- [3] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. 2020. How to backdoor federated learning. In *International Conference* on Artificial Intelligence and Statistics. PMLR, 2938–2948.
- [4] Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. 2019. Analyzing federated learning through an adversarial lens. In *International Conference on Machine Learning*. PMLR, 634–643.
- [5] Huili Chen, Bita Darvish Rohani, and Farinaz Koushanfar. 2018. Deepmarks: A digital fingerprinting framework for deep neural networks. arXiv preprint arXiv:1804.03648 (2018).
- [6] Ananya Choudhury, Johan van Soest, Stuti Nayak, and Andre Dekker. 2020. Personal health train on fhir: A privacy preserving federated approach for analyzing fair data in healthcare. In Machine Learning, Image Processing, Network Security and Data Sciences: Second International Conference, MIND 2020, Silchar, India, July 30-31, 2020, Proceedings, Part I 2. Springer, 85–95.
- [7] Bita Darvish Rouhani, Huili Chen, and Farinaz Koushanfar. 2019. Deepsigns: An end-to-end watermarking framework for ownership protection of deep neural networks. In Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems. 485–497.
- [8] Lixin Fan, Kam Woh Ng, and Chee Seng Chan. 2019. Rethinking deep neural network ownership verification: Embedding passports to defeat ambiguity attacks. Advances in neural information processing systems 32 (2019).
- [9] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. 2020. Inverting gradients-how easy is it to break privacy in federated learning? Advances in Neural Information Processing Systems 33 (2020), 16937–16947.
- [10] Robin C Geyer, Tassilo Klein, and Moin Nabi. 2017. Differentially private federated learning: A client level perspective. arXiv preprint arXiv:1712.07557 (2017).
- [11] Jia Guo and Miodrag Potkonjak. 2018. Watermarking deep neural networks for embedded systems. In 2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD). IEEE, 1–8.
- [12] Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both weights and connections for efficient neural network. Advances in neural information processing systems 28 (2015).
- [13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition. 770–778.
- [14] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).
- [15] Yann LeCun, Corinna Cortes, Chris Burges, et al. 2010. MNIST handwritten digit database.
- [16] Bowen Li, Lixin Fan, Hanlin Gu, Jie Li, and Qiang Yang. 2022. FedIPR: Ownership verification for federated deep neural network models. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022).
- [17] Fang-Qi Li, Shi-Lin Wang, and Alan Wee-Chung Liew. 2022. Watermarking Protocol for Deep Neural Network Ownership Regulation in Federated Learning. In 2022 IEEE International Conference on Multimedia and Expo Workshops (ICMEW). IEEE, 1–4.
- [18] Guobiao Li, Sheng Li, Zhenxing Qian, and Xinpeng Zhang. 2022. Encryption Resistant Deep Neural Network Watermarking. In ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 3064-3068.
- [19] Zheng Li, Chengyu Hu, Yang Zhang, and Shanqing Guo. 2019. How to prove your model belongs to you: A blind-watermark based framework to protect intellectual property of DNN. In Proceedings of the 35th Annual Computer Security Applications Conference. 126–137.
- [20] Jierui Lin, Min Du, and Jian Liu. 2019. Free-riders in federated learning: Attacks and defenses. arXiv preprint arXiv:1911.12560 (2019).
- [21] Yang Liu, Anbu Huang, Yun Luo, He Huang, Youzhi Liu, Yuanyuan Chen, Lican Feng, Tianjian Chen, Han Yu, and Qiang Yang. 2020. Fedvision: An online visual object detection platform powered by federated learning. In Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34. 13172–13179.

- [22] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In Artificial intelligence and statistics. PMLR, 1273-1282.
- [23] Pascal Paillier. 1999. Public-key cryptosystems based on composite degree residuosity classes. In Advances in Cryptology—EUROCRYPT'99: International Conference on the Theory and Application of Cryptographic Techniques Prague, Czech Republic, May 2–6, 1999 Proceedings 18. Springer, 223–238.
- [24] Xiangyu Qi, Tinghao Xie, Ruizhe Pan, Jifeng Zhu, Yong Yang, and Kai Bu. 2022. Towards practical deployment-stage backdoor attack on deep neural networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 13347–13357.
- [25] Hanchi Ren, Jingjing Deng, and Xianghua Xie. 2022. Grnn: generative regression neural network—a data leakage attack for federated learning. ACM Transactions on Intelligent Systems and Technology (TIST) 13, 4 (2022), 1–24.
- [26] Shuo Shao, Wenyuan Yang, Hanlin Gu, Jian Lou, Zhan Qin, Lixin Fan, Qiang Yang, and Kui Ren. 2022. FedTracker: Furnishing Ownership Verification and Traceability for Federated Learning Model. arXiv preprint arXiv:2211.07160 (2022).
- [27] Micah J Sheller, Brandon Edwards, G Anthony Reina, Jason Martin, Sarthak Pati, Aikaterini Kotrotsou, Mikhail Milchenko, Weilin Xu, Daniel Marcus, Rivka R Colen, et al. 2020. Federated learning in medicine: facilitating multi-institutional collaborations without sharing patient data. Scientific reports 10, 1 (2020), 1–12.
- [28] Ziteng Sun, Peter Kairouz, Ananda Theertha Suresh, and H Brendan McMahan. 2019. Can you really backdoor federated learning? arXiv preprint arXiv:1911.07963 (2019).
- [29] Nima Tajbakhsh, Jae Y Shin, Suryakanth R Gurudu, R Todd Hurst, Christopher B Kendall, Michael B Gotway, and Jianming Liang. 2016. Convolutional neural networks for medical image analysis: Full training or fine tuning? *IEEE transactions* on medical imaging 35, 5 (2016), 1299–1312.
- [30] Buse GA Tekgul, Yuxi Xia, Samuel Marchal, and N Asokan. 2021. WAFFLE: Watermarking in federated learning. In 2021 40th International Symposium on Reliable Distributed Systems (SRDS). IEEE, 310–320.
- [31] Yusuke Uchida, Yuki Nagai, Shigeyuki Sakazawa, and Shin'ichi Satoh. 2017. Embedding watermarks into deep neural networks. In Proceedings of the 2017 ACM on international conference on multimedia retrieval. 269–277.

- [32] Omar Abdel Wahab, Gaith Rjoub, Jamal Bentahar, and Robin Cohen. 2022. Federated against the cold: A trust-based federated learning approach to counter the cold start problem in recommendation systems. *Information Sciences* 601 (2022), 189–206.
- [33] Zhibo Wang, Mengkai Song, Zhifei Zhang, Yang Song, Qian Wang, and Hairong Qi. 2019. Beyond inferring class representatives: User-level privacy leakage from federated learning. In IEEE INFOCOM 2019-IEEE conference on computer communications. IEEE, 2512–2520.
- [34] Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H Yang, Farhad Farokhi, Shi Jin, Tony QS Quek, and H Vincent Poor. 2020. Federated learning with differential privacy: Algorithms and performance analysis. IEEE Transactions on Information Forensics and Security 15 (2020), 3454–3469.
- [35] Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747 (2017).
- [36] Wenyuan Yang, Shuo Shao, Yue Yang, Xiyao Liu, Zhihua Xia, Gerald Schaefer, and Hui Fang. 2022. Watermarking in Secure Federated Learning: A Verification Framework Based on Client-Side Backdooring. arXiv preprint arXiv:2211.07138 (2022).
- [37] Hongxu Yin, Arun Mallya, Arash Vahdat, Jose M Alvarez, Jan Kautz, and Pavlo Molchanov. 2021. See through gradients: Image batch recovery via gradinversion. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 16337–16346.
- [38] Jialong Zhang, Zhongshu Gu, Jiyong Jang, Hui Wu, Marc Ph Stoecklin, Heqing Huang, and Ian Molloy. 2018. Protecting intellectual property of deep neural networks with watermarking. In Proceedings of the 2018 on Asia Conference on Computer and Communications Security. 159–172.
- [39] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. 2020. idlg: Improved deep leakage from gradients. arXiv preprint arXiv:2001.02610 (2020).
- [40] Ligeng Zhu, Zhijian Liu, and Song Han. 2019. Deep leakage from gradients. Advances in neural information processing systems 32 (2019).
- [41] Renjie Zhu, Xinpeng Zhang, Mengte Shi, and Zhenjun Tang. 2020. Secure neural network watermarking protocol against forging attack. EURASIP Journal on Image and Video Processing 2020 (2020), 1–12.