Computers & Security 135 (2023) 103504

Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose

FedRight: An effective model copyright protection for federated learning

Jinyin Chen [a,b], Mingjun Li [b], Yao Cheng [c], Haibin Zheng [a,b,*]

[a] Institute of Cyberspace Security, Zhejiang University of Technology, Hangzhou, China
[b] College of Information Engineering, Zhejiang University of Technology, Hangzhou, China
[c] Digital Service, TÜV SÜD Asia Pacific Pte. Ltd., Singapore

ARTICLE INFO

Keywords:
Copyright protection
Federated learning
Model fingerprints
Robustness
Black-box fingerprints

ABSTRACT

Federated learning (FL), an effective distributed machine learning framework, implements model training and meanwhile protects local data privacy. It has been applied to a broad variety of practical areas due to their great performance and appreciable profits. *Who really owns the model, and how to protect the copyright* has become a real problem. Intuitively, the existing property rights protection methods in centralized scenarios (e.g., watermark embedding and model fingerprints) are possible solutions for FL. But they are still challenged by the distributed nature of FL in aspects of the no data sharing, parameter aggregation, and federated training settings. For the first time we formalize the problem of copyright protection for FL, and propose FedRight to protect model copyright based on model fingerprints, i.e., extracting model features by generating adversarial examples as model fingerprints. FedRight outperforms previous works in four key aspects: (i) *Validity* - it extracts model features to generate transferable fingerprints to train a detector to verify the copyright of the model. (ii) *Fidelity* - it is with imperceptible impact on the federated training, thus promises good main task performance. (iii) *Robustness* - it is empirically robust against malicious attack on copyright protection, i.e., fine-tuning, model pruning and adaptive attacks. (iv) *Black-box* - it is valid in black-box forensic scenario where only application programming interface calls to the model are available. Extensive evaluations across 3 datasets and 9 model structures demonstrate FedRight's superior fidelity, validity and robustness.

1. Introduction

Federated learning (McMahan et al., 2017, 2016; Yang et al., 2019; Blanco-Justicia et al., 2021; Zhang et al., 2021; Jiang et al., 2021; Wang et al., 2021) is an emerging distributed learning framework for user data privacy protection. In the federated learning scenario, there are usually one server and multiple clients. Depending on the specific application, it can be divided into horizontal federated learning, vertical federated learning and transferable federated learning. Horizontal federated learning (HFL) is the most popular one among them. The client trains a local model with locally collected data, and then only uploads the model information to the server. The server generates a global model with aggregation algorithms by aggregating the model information uploaded by clients, and distributes the global model back to the client for further training in an iterative manner. The raw training data are safely kept in the local client during the whole training process. Thus the FL successfully maintains the privacy of clients' training data, meanwhile implements the distributed training whose accuracy is comparable to that of the centralized training. Due to its privacy-preserving nature and efficient distributed training mechanism, FL has been widely applied to practical scenarios such as bank loans (Shingi, 2020), medical diagnosis (Kuo and Pham, 2022), and recommendation systems (Wahab et al., 2022).

Unfortunately, there lacks a proper method to protect the copyright of the FL-trained model. This problem directly leads to the difficulty for the owner of the model to claim the copyright even if the model is suspected to be illegally used by others for profit. How to better protect the property rights of the model and interests of the participants is an urgent issue that needs to be solved.

There are a number of existing methods for model copyright protection in the centralized scenario. Watermark embedding (Uchida et al., 2017; Vybornova, 2021; Li et al., 2020; Guo and Potkonjak, 2018; Li et al., 2019; Tekgul et al., 2021) and model fingerprinting (Zhao et al., 2020; Lukas et al., 2021; Cao et al., 2021) are the two mainstream model ownership verifications. They require the model owner to embed a unique secret watermark or generate a unique secret fingerprint

* Corresponding author at: Institute of Cyberspace Security, Zhejiang University of Technology, Hangzhou, China.
E-mail addresses: chenjinyin@zjut.edu.cn (J. Chen), limingjuns@outlook.com (M. Li), yao.cheng@tuvsud.com (Y. Cheng), haibinzheng320@gmail.com (H. Zheng).

https://doi.org/10.1016/j.cose.2023.103504
Received 26 December 2022; Received in revised form 8 June 2023; Accepted 20 September 2023
Available online 26 September 2023

before releasing the model. When it comes to the situation where the model owner needs to claim the ownership of the model, the owner can use the secret watermark or fingerprint in combination with the behavior of the suspect model to prove so. However, embedding the watermark needs to change the model parameters, the process inevitably affects the performance and efficiency of the main task of federated training. Model fingerprints methods extract model features (e.g., gradients) to generate an adversarial example (Luo et al., 2018; Huang et al., April 24-26, 2017; Zheng et al., 2020) and use the transferability of the generated adversarial examples to determine whether the use of the suspect model is unauthorized. However, fingerprints are easily erased by adversarial retraining (Lukas et al., 2021).

Different from the centralized training, FL is distributed and involved different parties such as the server and the clients. However, according the assumption of the FL, a client may not be trustworthy but a server is, i.e., a trusted and honest server in FL. Therefore, it is more rational to rely the copyright protection on the server side than on the client side. However, there are still challenges to directly migrate the centralized copyright protection to FL.

We summarize the main challenges addressed for the model copyright protection in FL: (i) *Data limitation* - An honest server cannot access the training data owned by the clients to make watermarks or model fingerprints. (ii) *Accuracy sacrifice* - The protection should not excessively sacrifice the server model's accuracy, as well as the FL's training efficiency. (iii) *Malicious attacks* - The protection method at the server side needs to deal with downstream attacks, i.e. fine-tuning, model pruning, etc., as well as malicious clients upstream in collaborative training. (iv) *Black-box ownership verification* - The model copyright verification should be practical for black-box ownership verification scenario (i.e., without any knowledge of the model's architecture or parameters), for example, only Application Programming Interface (API) calls to query the suspect model are available.

To address these challenges, we propose FedRight, the first approach proposing to use model fingerprints to protect model copyright in FL. FedRight is based on model fingerprints techniques but with improved robust against adversarial retraining attacks. Generally, we rely on the honest FL server to generate fingerprints for the FL-trained model and train a separate model using these fingerprints and their outputs. Specifically, by using secret key samples, the server leverages on the extracted global model features and generates a set of adversarial examples as model fingerprints. Note that the key samples here are not necessarily of the same distribution with the training or testing samples. Then we use these fingerprints as input to the model and obtain the feature distributions of these fingerprints. The feature distribution of the key samples is used to train a new model, i.e., the detector, which is to predict the ground-truth class encode of the key samples. When testing a suspect model, we obtain the feature distribution of the key samples output by the suspect model, and use the detector to predict the key samples' ground-truth class. When the accuracy of the detector exceeds a threshold, the ownership of the model is claimed.

Moreover, FedRight has designed the model fingerprint to have adaptive enhancement capability. It gradually adds model features in response to the changes in the global model during federation training. During the verification phase, FedRight only accesses the model's output of key samples, which is suitable for the black-box forensic scenarios. We conducted extensive experiments to evaluate the validity, fidelity, efficiency and robustness of FedRight.

The main contributions of this paper are summarized as follows.

- We propose the FL-oriented model copyright protection method - FedRight by relying on the honest server to generate robust fingerprints without any knowledge of the training data on the client side.
- FedRight innovatively introduces a detector to capture the relationship between the feature distribution of the key samples output by the target model and the key samples' label. FedRight can effectively verify the ownership of the target model according to the accuracy of the detector, i.e., measuring to what extent a suspect model's behavior on these key samples aligns exactly with the target model.
- We conducted extensive experiments to evaluate FedRight on 3 datasets and 9 models. The experimental results show the advantages of FedRight compared to previous work and satisfy validity, fidelity, robustness, and black-box ownership verification in FL scenarios.

2. Related works

2.1. Centralized model IP protection

Existing means of intellectual property (IP) protection are mainly applied to the deep neural network (DNN) in centralized scenarios. There are two streams of IP protection methods, i.e., watermarking and fingerprinting. Since Uchida et al. (2017) first proposed a watermarking model framework in a white-box scenario. It gets further upgraded in the face of the restrictions on access rights in the black-box scenario. Wei et al. (2018) combine common data samples with exclusive "logos" and train models to predict them as specific labels so that the ownership of the model can be verified by a third party. Jia et al. (2021) propose entangled watermark embedding to address watermark removal attacks. In order to be robust against copyright evasion attacks, Li et al. (2019) propose a blind watermarking method to generate key samples with a distribution similar to the original samples. However, embedding watermarks changes the model parameters and this process inevitably affects the performance of the main task of the model. Fingerprinting is another IP protection that does not change the model parameters. Zhao et al. (2020), Lukas et al. (2021) used adversarial examples as model fingerprints and exploited their transferability to verify model IP rights. Fingerprinting methods generate model fingerprints by extracting model features, hence there is no impact on the model performance. However, model fingerprints can be removed in the face of adaptive adversarial retraining attacks. Moreover, the privacy-preserving nature of FL, where there is no training data available, poses great challenge to the fingerprinting process on the server.

2.2. IP protection in FL

The mainstream IP protection in FL is still based on watermarking (Tekgul et al., 2021; Fan et al., 2021; Li et al., 2022). Specifically, Tekgul et al. (2021) proposed WAFFLE which achieves IP protection by embedding watermarks on the server. Fan et al. (2021) proposed FedIPR which embeds and detects watermarks by each client independently. Li et al. (2022) designed the Merkle-Sign watermarking framework, which combines the state-of-the-art watermarking scheme and a security mechanism designed for distributed storage to protect both privacy and ownership. However, they either surfer the inherent limitation of the watermarking scheme or pose changes to the training of federated learning, both of which lead to negative impact on the model performance.

2.3. Attacks against IP protection

Malicious attacks may launch attacks aiming at obtaining the model without degrading its accuracy, and meanwhile preventing the model owner from proving his/her ownership. There are model modification methods (Uchida et al., 2017; Rouhani et al., 2019; Namba and Sakuma, 2019), copyright evasion attacks (Hitaj et al., 2019), and removal attacks (Shafieinejad et al., 2019). The model modification includes model fine-tuning Uchida et al. (2017), model pruning (Rouhani et al., 2019), model compression (Uchida et al., 2017), and model retraining (Namba and Sakuma, 2019). In order to evade the legitimate

owner's verification, in the copyright evasion attack against IP protection (Hitaj et al., 2019), the attacker will try to construct a detector to detect whether the queried sample is a clean or possibly critical sample. Once the detector determines that the queried instance is a possible critical sample, the stolen model will return a random label. Removal attacks, the attackers attempt to remove the watermark. Shafieinejad et al. (June, 22-25, 2021) studied removal attacks based on backdoor watermarking schemes and proposed a method to detect whether a model contains a watermark. Sun et al. (2021) used generative adversarial network (GAN) to detect and reverse the backdoor trigger in the model and then fine-tune the model with the reversed trigger to remove the backdoor based watermark. Chattopadhyay et al. (2020) used GAN to generate samples for retraining that can obtain a model with similar performance while removing the watermark.

3. Preliminaries and background

In this section, we introduce the horizontal federation framework and the background knowledge for generating model fingerprints.

3.1. Horizontal federated learning

Horizontal federated learning (Wu, 2021) is applied to scenarios where the datasets of each client have the same feature space and different sample spaces. All private data are on the client and cannot be accessed by other clients. After each client C_i performs model training locally, it uploads the model parameter w_i to the server. Then, the server performs an aggregation operation on the uploaded parameters to form a global model parameter w_g, which is then returned to each client to continue training. The commonly used aggregation rule is as follows:

$$w_g = \frac{1}{K}\sum_{i=1}^{K} w_i \tag{1}$$

where K is the total number of clients participating in training.

In HFL, only information about locally trained model is shared. It thus ensures the privacy of the client's local data.

3.2. Key samples

Key samples are the seeds to generate model fingerprints. Key samples are normally protected from attackers' access. Existing model protection methods generate fingerprints using key samples from training data. However, this could be an issue in FL since the server in FL does not have access to client data, so it cannot use the training data as key samples. If the key samples are part of the training data, attackers are prone to use the adversarial retraining attack of training data to eliminate fingerprints. Therefore, the key samples used in this paper are independent of the training data. This could 1) fits in the FL scenario where training data are not available; and 2) greatly affects the main task performance if the attacker performs retraining attack.

Using non-training data as key samples can also generate valid model fingerprints because FedRight's model fingerprints consist of key samples and model-specific features. Specifically, the key sample is only the carrier of the model fingerprint, which depends critically on the extracted model-specific features. Thus, FedRight proposes an effective method for generating fingerprints without training data.

3.3. Adversarial examples

Existing work (Zhao et al., 2020; Lukas et al., 2021) utilizes adversarial examples as model fingerprints. Model can easily be fooled by well-designed adversarial example x_{adv}, i.e., by adding small perturbations σ to the normal example x. There are many algorithms for generating adversarial examples, such as FGSM (Goodfellow et al., 2015), C&W (Carlini and Wagner, 2016), PGD (Madry et al., 2018), etc. Given a sample x and an output label y_{orig}, the attacker can always find well-targeted adversarial examples such that the output label is not y_{orig}, depending on the specific optimization process that can be divided into targeted and untargeted attacks.

$$\begin{cases} f(x+\sigma) \neq y_{orig}, & \text{untargeted attack} \\ f(x+\sigma) = y_{target}, & \text{targeted attack} \end{cases} \tag{2}$$

where $f(\cdot)$ is the input to the target model; y_{target} denotes the expected target label.

Since adversarial examples are generated based on the model, e.g., model structure and parameters, they can be effectively used as fingerprints to identify a model. As long as the attackers are not aware of the adversarial examples, especially the corresponding model output of these adversarial examples, model owners can verify the ownership of the model by testing the suspect model with the secret adversarial examples and the corresponding predictions.

3.4. Model fingerprints

Model fingerprinting is a method that produces a model fingerprint F according to the target model to realize copyright protection.

It includes the following two algorithms:

Generate model fingerprints. $F = Generate(G; D_{key})$. The generation process has access to global model G and key samples D_{key}, and uses this knowledge to generate the model fingerprint F for a specific label F_y.

Validate model fingerprints. $F_y^{pre} = Validate(G_{sus}; F)$. The model owner validates the suspect model G_{sus} using the model fingerprint F, and the obtained output prediction label F_y^{pre} are compared with F_y to verify the ownership.

The evaluation of the suspect model ownership using the validation algorithm is based on an empirically determined threshold α. If the matching rate between F_y^{pre} and F_y is greater than α, it is verified as a stolen model. Since the model fingerprint, i.e., the adversarial example, has transferability, the model ownership can still be verified even though the suspect model is slightly modified from the original model.

Existing model fingerprint verification algorithms only using prediction labels are vulnerable to ambiguity attacks, where a fake owner uses samples with the same output label as the original model fingerprints to falsely claim the model ownership. For example, the model owner is considered to have ownership of the model using a key sample output label '1'. However, the attacker uses a non-key sample output label also '1', which is an eventuality and makes the ownership of the model confusing. We take this into account when designing our fingerprint verification algorithm. We use the output distribution features of the model instead of prediction label only, which will be detailed in Section 5.4.

4. Threat model

This paper takes the server as the initiator of property protection, and the main threat target is the upstream and downstream attackers.

Suppose the horizontal federation contains K clients C_i and a trusted server, where each client has local data D_i, $i \in \{1, 2, ...K\}$. All clients collaborate to train a global model G, but the distributed mechanism brings uncertainties, such as the existence of malicious client C_j^m during the training process who intends to exploit the high-value global model G for illegal profit. Moreover, when the federation training is finished, the public sharing of the global model leads to a downstream attacker P_m being able to tune it to avoid the owner's IP rights traceability, and then uses the tuned model for other illegal avenues. To clearly describe such a threat scenario, we give the formal definitions of attacker and defender as follows.

Attacker ability. We suppose the attacker can be the upstream malicious client C_j^m or the downstream malicious party P_m. The upstream malicious client C_j^m has the generally accessible information

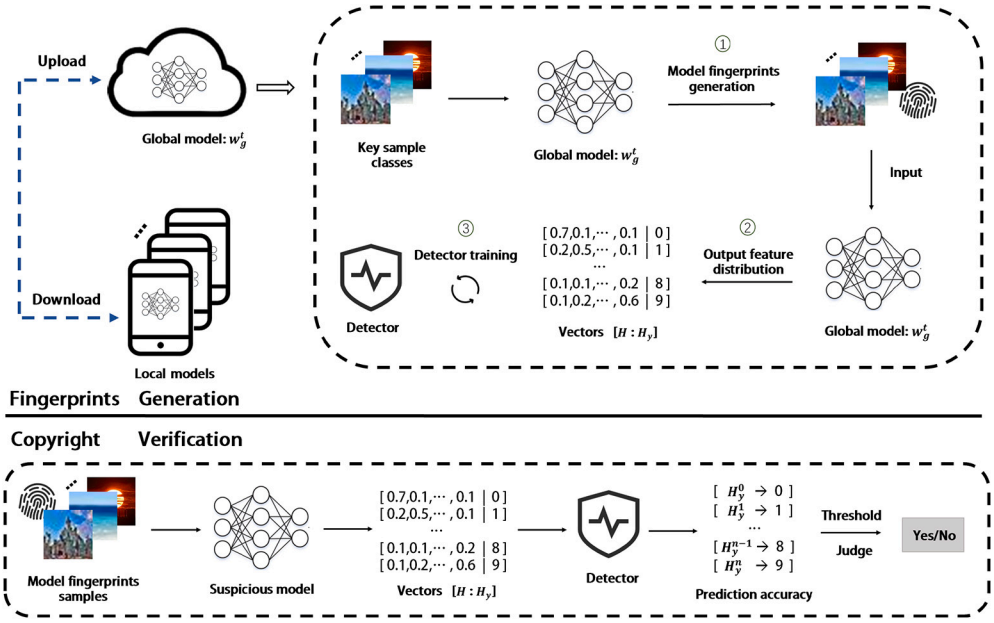


Fig. 1. Overview of FedRight. The server clones a global model, extracts the inherent features of the model, and generates adversarial examples from key samples as model fingerprints. After input to the global model, the output distribution features are obtained and labeled to train the detector. Using the transferability of the adversarial example, model owners can use the detector to determine the ownership of the suspect model.

set $< \Theta_g, L, l_r, R_{aggr} >$ in the FL system, where Θ_g is the global model parameter, L is the loss function, l_r is the learning rate and R_{aggr} is the aggregation rule. However, C_j^m cannot access or manipulate other clients' data D_i. The downstream malicious party P_m can generally only have the global model parameter Θ_g information. But both attackers can use some modification attacks against the global model (e.g., model fine-tuning Uchida et al., 2017, model pruning Rouhani et al., 2019), denoted as $\Theta_g' = Modi(\Theta_g)$, thereby making it different from the original model $G(\Theta_g) \neq G(\Theta_g')$, to prevent the trace verification of model IP protection, but still has high accuracy: $\left\| Acc(D_{test}; G(\Theta_g')) - Acc(D_{test}; G(\Theta_g)) \right\| < \delta$, where δ is a small number.

Defender ability. The goal of the defender P_d is to protect the IP of models collaboratively trained by multiple clients in the FL scenario. In a practical scenario, the P_d has no white-box access to the model stolen by the attacker. However, the P_d should have black-box access to the suspect model G_{sus}, i.e., it can query the suspicious model G_{sus} with the $< X_{key}, Y_{key} >$ and obtain the output $Y_{pre} \leftarrow f(X_{key}, G_{sus})$ to verify the model's IP $Ver(< Y_{pre} = Y_{key} ? >; G_{sus})$. A rational assumption would be 1) the defender has the white-box access to the model to be protected, since he/she owns the copyright; 2) the defender only has black-box access to the suspect model deployed by others.

5. Methodology

We rely on the server to generate fingerprints for the purpose the model copyright protection. An overview of FedRight is shown in Fig. 1. The overall process is consisted of two phases, i.e., fingerprints generation and copyright verification. Specifically, the first phase is divided into three steps: ① generate adversarial examples as model fingerprints; ② obtain the output distribution features of the model fingerprints and label them with specific labels; ③ detector training. The verification phase happens when there is a suspect model and the model owner intends to verify the ownership of the model. The owner needs to obtain the feature distribution of the fingerprint samples by querying the suspect model. Then the obtained distribution is fed to our detector to see whether the prediction accuracy is high enough to claim the model's ownership.

5.1. Model fingerprints generation

Since the server does not have access to the client's local data, but can explicitly know the specific training task. Based on this knowledge, we collect some key samples D_{key} that are not related to the training data. For example, if the server knows that the main task of federated training is the Handwriting Digit Recognition, it can use face images as the key samples. The class number of the key samples D_{key} is also selected depending on the number of classes of the main task. Usually, it is preferred to increase the number of key sample classes to improve the performance, e.g., with high detection rate and low false positive rate.

FedRight does not have a dependency on the key samples. The model fingerprint is generated with two main parts, one is the selected key samples and the other is the extracted model-specific features. The key samples are only the carrier, but the key is the extracted model-specific features, which reflect the model's unique information. Usually, the key samples collected need only be unrelated to the federated training task data, which is to prevent attackers from using the training samples to perform adversarial retraining attacks to affect the effectiveness of the property verification.

The selected key samples D_{key} are used to generate model fingerprints F. Specifically, by using adversarial attacks, key samples D_{key} and extracted model perturbations are combined to form adversarial examples D_{adv} as model fingerprints. The goal of the adversarial attack is to minimally interfere with the normal examples while maximally misleading the classifier with a high confidence level. This can be modeled as an optimization problem. We introduce a generic model of adversarial attacks against a DNN, named ρ-loss, defined as:

$$argmin \left\{ \epsilon \|\rho\| + \lambda Loss(y, f_{pre}(\Theta, x)) \right\}$$
$$s.t.\ \rho = X_{adv} - X_{key},\quad X_{adv} \in \left\{ D_{adv} \right\},\quad X_{key} \in \left\{ D_{key} \right\},\quad y \in \left\{ y_{orig}, y_{target} \right\} \tag{3}$$

where ϵ is the scale factor used to balance the order-of-magnitude difference in the perturbation; ρ denotes the perturbation between the adversarial example X_{adv} and the key example X_{key}. X_{adv} is the sample in the set of adversarial examples D_{adv}, and X_{key} is the sample in the set of key examples D_{key}; $Loss(\cdot, \cdot)$ is the loss function of the model;

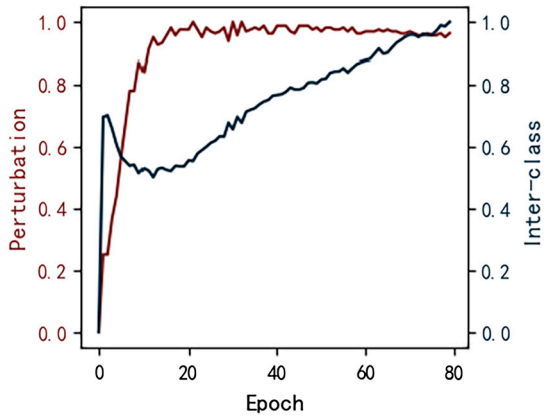


Fig. 2. The correlation between the model feature perturbations added and the distance between model fingerprint classes in the process of generating model fingerprints.

$f_{pre}(\cdot,\cdot)$ is the prediction result of the model; Θ denotes the parameters of the model; x and y denote the input key example D_{key} and the corresponding output class labels, respectively. When $\lambda=1$ and $y=y_{target}$, ρ-loss represents a targeted attack, and when $\lambda=$-1 and $y=y_{orig}$, ρ-loss represents an untargeted attack.

The targeted attack has a clear optimization direction and better attack effect compared with the untargeted attack. Thus, we use targeted attacks by setting $\lambda=1$, and generate model fingerprints F for the corresponding classes. It is worth noting that the FedRight is generic framework transferable for other adversarial attacks including C&W (Carlini and Wagner, 2016), PGD (Madry et al., 2018), etc.

To better explain the uniqueness of model fingerprints that can be exploited, we visualize the inter-class distance of the fingerprints during the global model training as shown in Fig. 2, and find that the training of the global model contains two phases, the oscillation phase T (i.e., the first 20 epochs) and the linear growth phase (i.e., the last 60 epochs). In the oscillation period T, the Mean Square Error (MSE) value of our proposed model fingerprint increases continuously with the training epoch, but the growth curve has a certain time delay t_d. During the linear growth, the MSE value of the model fingerprint is essentially constant. During the oscillation period T, we infer that the model is rapidly learning the "knowledge" in the data. Therefore, the model fingerprint is also rapidly collecting the features trained by the model, which is reflected by the increasing MSE value of the model fingerprint. The features of the model fingerprint are collected after the model learns the "knowledge", so there is a certain time delay t_d. In the linear growth process, the model just keeps converging the intra-class distance and increasing the inter-class distance. We infer that there are fewer features that can be provided to the model fingerprint at this time, and therefore the MSE value of the model fingerprint remains essentially unchanged. To sum up, the model fingerprint we propose is directly related to the global model, for which the global model is unique.

Regarding the timing of model fingerprints generation, some options are possible, such as generating fingerprints during the training process or after the training. If the generation of model fingerprints starts at the end of federated training, since the client gets the global model published by the server in each epoch of training, then it is possible for the malicious client to save the global models of previous epochs locally to evade IP verification. Therefore, the generation of model fingerprints needs to be performed during the federated training. The other way is considering the model fingerprint generated from the beginning to the end of the whole federated training. It still brings some problems, not only to consider the time overhead of IP protection, but also at the early stage of training, the parameters of the global model change sharply with the iterative update in FL, thus the model fingerprint generated in the previous epoch to be input to the global model in the next epoch of training will lead the target label shifted, resulting in great change in the output distribution vector, i.e., the model fingerprint is invalid. As shown in Fig. 3(a), we randomly select the model fingerprint with target label '5' generated in epoch 50 and show the visualization of its output distribution vector in epochs 50 to 52. It can be found that the model fingerprint generated in epoch 50 has been invalidated in the subsequent epochs. It indicates that the static generation method is difficult to maintain the validity of fingerprints in dynamic training.

To solve this problem and ensure the stability of the model fingerprint, we propose a model fingerprint adaptive enhancement mechanism.

5.2. Model fingerprinting adaptive enhancement

The model parameters usually change sharply at the early stage of FL training, and meanwhile, the initial model accuracy is low and not of high value for use. Therefore, fingerprint generation should be executed from the middle stage of training. The middle stage of performing fingerprint generation is not absolute, but rather a relative range interval. We visualized the global model parameter update rate and the timing of selecting fingerprint generation in 10 round intervals to get the set threshold for MNIST on LeNet-5 model as shown in Fig. 4. For the model parameter update rate, it can be found that the global model parameter change rate increases rapidly in the first 5 epochs, decays rapidly from epochs 5 to 20, and gradually levels off afterwards. For the threshold α, the α starts to gradually decrease in epochs 0-30 as the timing of selecting the execution of generating fingerprints shifts backward; after 30 epochs, the α starts to increase. This indicates that in the preliminary model training, the initialized model parameters take some time to learn the features of the data, and the parameters are highly variable and not stable enough in this process, resulting in high α. As the timing of generating fingerprints moved backward, the α began to gradually decrease. It is not until after 30 epochs that the α starts to increase, which may be due to the fact that as the overall number of epochs to generate fingerprints decreases, the amount of data generated is decreasing, which can affect the training of the detector M. It is worth noting that the overall α is not higher than 28%, which is within the acceptable range. To sum up, the fingerprint generation is performed starting in the middle of training having a lower threshold α.

On the other hand, adaptive enhancement of fingerprints is achieved by setting a fingerprint validity detection checkpoint, which is defined as follows:

$$F^{t+1}=\begin{cases} D^t_{adv}+\epsilon\|\rho\|, & Checkpoint(D^t_{adv})\times \\ D^t_{adv}, & Checkpoint(D^t_{adv})\surd \end{cases} \tag{4}$$

where $Checkpoint(\cdot)$ is to determine whether the target label of the adversarial sample has been shifted, i.e., whether the fingerprint is invalidated; $\epsilon\|\rho\|$ denotes adding the feature perturbation of the current epoch model; The "×" indicates invalid, and the "$\surd$" indicates valid.

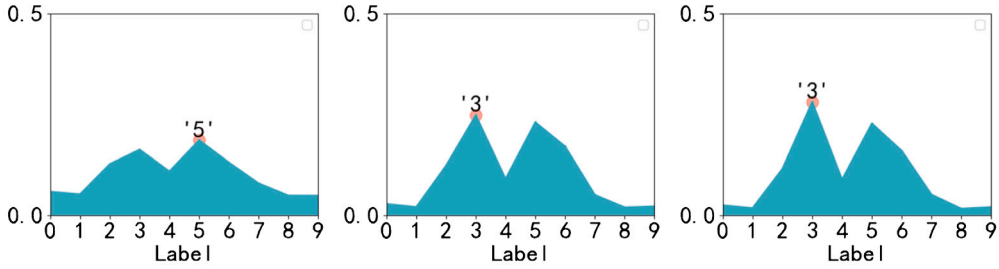
In each epoch, we use Algorithm 1 to judge whether the previous model fingerprint is valid and if it is invalid, we continue to add the model feature perturbation in the current epoch to achieve the effect of dynamic enhancement.

From Fig. 3(b), we can also see that the model fingerprint generation algorithm with adaptive enhancement can maintain better across epochs, which effectively improves the validity of the generated fingerprints.

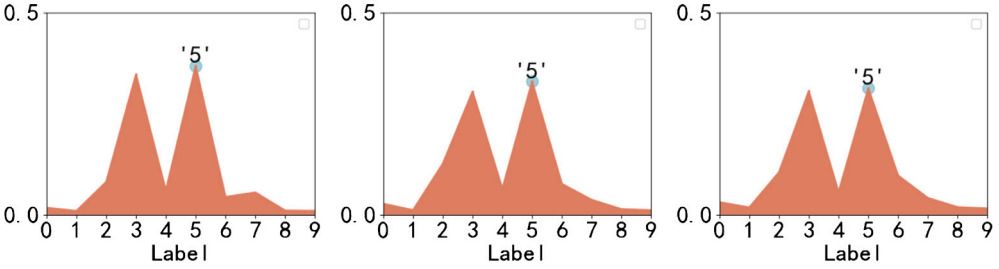
5.3. Detector training

The detector M uses the MLP structure, whose training data is constructed from the vector H of feature distributions obtained from the model fingerprint F input to the global model G. And H is labeled with a specific label H_y to train the detector M, where H_y is consistent with the model fingerprint label, i.e., H_y corresponds to the label of D_{adv}.

$$H=G(X_{adv}) \quad , \quad X_{adv}\in\{D_{adv}\} \tag{5}$$



(a) The static model fingerprint generation



(b) The adaptive enhancement to model fingerprint generation

Fig. 3. The output distribution feature vectors are visualized for a LeNet-5 model trained on MNIST dataset. The fingerprints are generated in the 50^{th} epoch and verified in the 50^{th}, 51^{st} and 52^{nd} epochs.

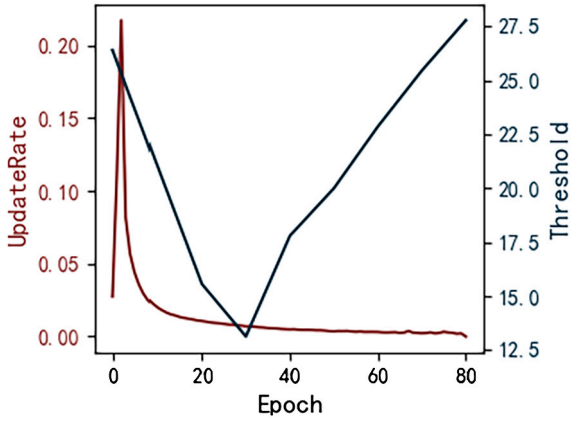


Fig. 4. The correlation between the update rate of the global model parameters and the threshold set at different time of performing the generation of fingerprints.

FedRight is robust to model modification attacks. For the model modification attack during training, because the detector M does not directly rely on the parameter features of the model, but trains the detector based on the output feature H of the model fingerprint, the output feature H will change according to the model, thus making the detector M dynamic for training. In this way, it achieves a property rights statement for the model. In the face of model modification attacks at the end of the training, the model fingerprint output features do not change dramatically while maintaining the model's accuracy, so the detector can still effectively verify model property rights.

Training the detector M using the output distribution feature H effectively eliminates the risk arising from malicious clients using ambiguity attacks during the property rights verification process, i.e., using non-key samples consistent with the output labels of the key samples for property rights obfuscation. The reason is that the behavior of the distribution features H of key samples and non-key samples are non-correlated.

Meanwhile, in order to improve the validation capability of the detector M, it can be trained by increasing the number of key sample classes c to generate different classes of adversarial examples.

$$Train(< H, H_y >; M(w_m, c)) \tag{6}$$

where $Train(.;.)$ represents the training of the detector M using the feature distribution vector H and the given label H_y. w_m denotes the detector model parameters and c denotes the number of classes of key samples.

The training of detector M is independent of the main task FL training. Once the server aggregates the global model, it is immediately distributed to individual clients, while the current global model is cloned for generating model fingerprints F and training the detector M. This independence means that the copyright protection is not at the expense of model accuracy.

5.4. IP verification

The trained detector M can be used to identify the ownership of a suspect model. In the verification phase, the model fingerprint is input to the suspect model and the output distribution vector H is input to the detector M to check whether the output label is consistent with the set label H_y. Ownership of the model is determined based on whether the accuracy of the detector is above a set threshold α. In this process, only suspicious model inputs and outputs are utilized with no access to the internal parameters of the model. The detail of FedRight is shown in Algorithm 1.

Algorithm 1 FedRight.

Input: total number of clients K; dataset $\{D_i\}$ of each client, $i \in \{1, 2, ..., K\}$; key samples $\{D_{key}\}$ in the server; the global epochs t; initial global model parameters $w_g^{t=0}$, detector parameters $w_m^{t=0}$; hyperparameter $\epsilon = 0.05$.
Output: the global model G and detector M.

Initialization: local model $w_i^{t=0} = w_g^{t=0}$.
Role: Client C_i
  $w_i^{t+1} \Leftarrow Update(w_g^t)$
  Local updates w_i^{t+1} upload to Server
Role: Server
  Calculate w_g^{t+1} according to Eq. (1)
  Fingerprints generation:
    Generating F according to Eq. (3)
    Get the output vector H of F according to Eq. (5)
    Enhancing F according to Eq. (4)
    Train the M according to Eq. (6)
Return: the global model G and detector M

Table 1
Dataset and model parameter settings.

Datasets	Samples	Dimensions	Classes	Models	Learning Rate	Momentum	Epoches	Bach Size
MNIST	70,000	28×28	10	LeNet-1, LeNet-4, LeNet-5	0.005	0.0001	80	32
CIFAR-10	60,000	32×32	10	VGG-11, VGG-13, VGG-16	0.01	0.9	100	32
CIFAR-100	60,000	32×32	100	ResNet-18, ResNet-34, ResNet-50	0.001	0.001	100	32

The validity of the property rights verification is attributed to the elaboration of the model fingerprint, which reflects the model-specific "fingerprint information". The trained detector effectively captures the output features specific to the model fingerprint.

5.5. Algorithm complexity

We analyze the complexity of FedRight on client-side and server-side, respectively. On the client side, the time complexity is mainly dependent on the number of training epochs for the local model.

$$T_{client} \sim \mathcal{O}(t_l) \tag{7}$$

where t_l is the local training epochs.

On the server side, the server implements two main parts of work, and the two parts of work are independent of each other. The first part of the work focuses on parameter aggregation of model parameters uploaded by clients and then distributed to each client.

$$Part_1 : T_{server} \sim \mathcal{O}(K) \tag{8}$$

where K is the number of clients participating in the training.

The second part of the work focuses on the IP declaration process, i.e., the time complexity of the FedRight framework. It consists of three parts: (1) generating the model fingerprints F; (2) detecting whether the model fingerprints are invalid and performs enhancements; (3) training the detector M.

$$Part_2 : T_{server} \sim \mathcal{O}(Iter * n) + \mathcal{O}(n + Iter * n_{inv}) + \mathcal{O}(t_m) \tag{9}$$

where $Iter$ is the number of optimization iterations, n is the number of key samples, n_{inv} is the number of invalid model fingerprints and t_m is the number of training epochs of the detector M.

The FedRight framework has low time complexity. The overhead introduced by FedRight is only $Part_2 : T_{server}$. Selecting the output features as the samples for detector training and its parameter size is within an acceptable range, making the complexity of detector training low. Moreover, the property declaration process of FedRight does not affect each other with federated training, which effectively mitigates the impact on the efficiency of federated training.

6. Experiments design and setup

Platform: i7-7700K 4.20GHzx8 (CPU), TITAN Xp 12GiB x2 (GPU), 16GBx4 memory (DDR4), Ubuntu 16.04 (OS), Python 3.6, pytorch1.8.2.

Datasets: We evaluate FedRight on three datasets, i.e., MNIST (LeCun et al., 1998), CIFAR-10 (Ayi and El-Sharkawy, 2020) and CIFAR-100 (Singla et al., 2022). MNIST dataset contains 70,000 real-world handwritten images with digits ranging from 0 to 9. Both the CIFAR-10 and CIFAR-100 datasets contain 60,000 color images of size 32 x 32, with 10 classes of 6,000 images each for CIFAR-10 and 100 classes of 600 images each for CIFAR-100. The detailed information of datasets is shown in Table 1.

Number of clients: We adopt 5 clients for FL training in all experiments except the parameter sensitivity experiments with different client numbers.

Models: A number of classifiers are used for verification on various datasets. For MNIST, LeNet-1, LeNet-4 and LeNet-5 (El-Sawy et al., 2016) are used for classification. For more complex image datasets, CIFAR-10 and CIFAR-100, VGG-11, VGG-13, VGG-16 (Chen, 2022), and ResNet-18 (He et al., 2016), ResNet-34, ResNet-50 are adopted, respectively. Please refer to Table 1 for the above specific parameter settings. All evaluation results are the average of ten runs under the same setting.

Adversarial Attacks: In the main experiments a targeted attack method based on FGSM is used. Meanwhile, in sensitivity analysis experiments, C&W and PGD adversarial attack methods are adopted to illustrate the trasferability of FedRight.

Hyper-Parameters: For all experiments, we set the hyperparameter $\epsilon = 0.08$, and select the key sample of $c = 10$.

Attack Methods: To assess the robustness of FedRight, we used two well-known model modification attacks: fine-tuning and model pruning, as well as one adaptive attack: adversarial retraining. Attacks by malicious clients during training are also considered, and copyright evasion attacks are analyzed.

FtuningAtt (Model Fine-tuning Attack (Uchida et al., 2017)): It is a classic approach, allowing an attacker to re-train the model with comparable performance to the original model, but of different parameters. In this paper, the model is fine-tuned by using 10% of the data in the test set.

PruningAtt (Model Pruning Attack (Rouhani et al., 2019)): It is designed to cut down targeted parameters and to obtain a new model that is different from the original model but still has similar accuracy. We use the pruning algorithm in Rouhani et al. (2019), setting a certain percentage of the parameters with the smallest absolute value to 0. The percentage is set between 30% to 90% with an interval of 10%.

AdaptiveAtt (Adaptive Attack): We consider the AdaptiveAtt to eliminate the fingerprints and perform adversarial retraining of the model according to the different knowledge possessed by the attacker.

CollabAtt (Collaborative Attack): Malicious clients can collaborate, including multiple upstream clients collaborating and upstream and downstream malicious parties to combine to damage IP.

MaliClientAtt (Malicious Client Attack): In comparison with the WAFFLE method, the malicious client is set up to perform removal attack (Shafieinejad et al., June, 22-25, 2021) during the fingerprint generation phase to counteract IP protection.

CopEvaAtt (Copyright Evasion Attack (Hitaj et al., 2019)): To evade the legitimate owner's verification, the attacker attempts to construct a detector to detect whether the queried sample is a clean sample or possibly a fingerprinting sample. The attacker deliberately returns a random label if the detector detects any fingerprinting sample.

Baselines: For the watermarking IP protection scheme under FL discussed in related work, we use WAFFLE (Tekgul et al., 2021), which is the latest and also on the server side, as a baseline. And we also compare $\text{FedRight}_{\text{D}}$, which is with the adaptive enhancement to model fingerprinting, with the static generation model fingerprints method $\text{FedRight}_{\text{S}}$ where the fingerprints are statically generated in each epoch.

Evaluation Metrics: We analyzed FedRight's performance by measuring the following metrics. (1) Fidelity: side effects on the main classification tasks, i.e., the global model accuracy (GMC). (2) Validity: whether ownership of the model can be successfully verified, i.e., detector accuracy (DMC). (3) Robustness: resistance to attacks, i.e., detector accuracy after being attacked (DMC_{Att}). The evaluation results regarding GMC, DMC and DMC_{Att} are all shown in percentage in this section.

Threshold α: We set the threshold α mainly based on the statistics of two experiments. 1) Counting the DMC of the property rights model and the DMC_{Att} in the face of various attacks. 2) Using different training strategies (including federated learning and single-machine training),

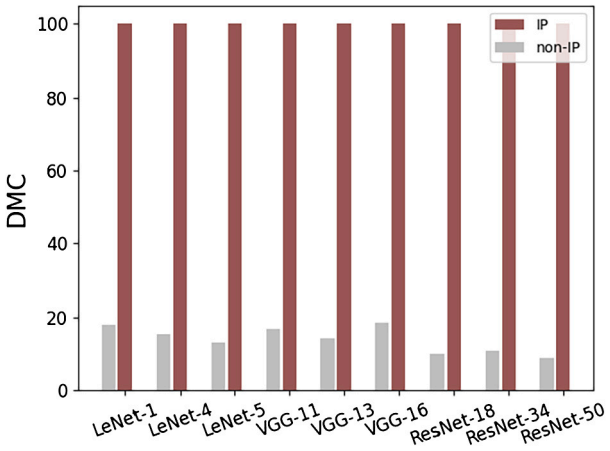


Fig. 5. The accuracy of detector for nine different models with three datasets. High DMC indicates the ownership of the model is verified.

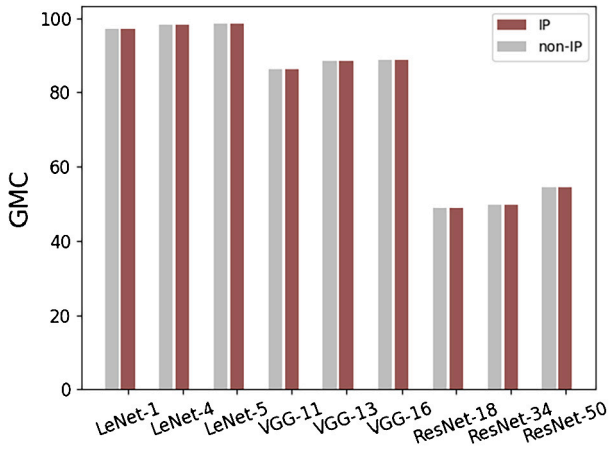


Fig. 6. The accuracy of the global model with IP protection and the global model for normal federated training.

the same dataset and model training to get a non-IP model, i.e., without making an IP statement on it, and statistics of its DMC and DMC_{Att}.

7. Evaluation and analysis

In this section, we assess FedRight's performance by answering the following five research questions (RQs).

- **RQ1**: Can FedRight effectively be used to claim the ownership of a given DNN model?
- **RQ2**: Regarding the fidelity of FedRight, what is its impact on main task performance?
- **RQ3**: How robust is FedRight in the face of attacks against IP protection?
- **RQ4**: What are the advantages of model fingerprinting over watermarking in FL setting?
- **RQ5**: How is the parameter sensitivity of FedRight?

7.1. RQ1: validity

In this section, we evaluate the effectiveness of FedRight. The purpose is to measure whether we can successfully verify the ownership of the target model under the protection of FedRight. We test DMC between the IP models and the non-IP models (i.e., models that are not under our ownership). In this case, the non-IP models were centrally trained with the same training data and model structure as the IP models. The detector M is used to validate these models and to evaluate the final DMC.

The results show that the detector effectively identifies the feature distribution of model fingerprints and predicts them to predefined labels with high accuracy. As shown in Fig. 5, all non-IP models only achieve an accuracy of 8.80%-18.40%, which is at about the same level of random guessing, i.e., the FedRight does not falsely claim ownership of non-IP models. In contrast, FedRight is encouragingly shown to achieve 100% accuracy on models that declare IP rights. The reason is that each declared IP model has the unique model fingerprint, and using its output vector to train the detector can effectively predict it in the subsequent validation process with high accuracy to avoid misclassification.

> ***Answer to RQ1:*** FedRight effectively verifies the ownership of the target model and achieves 100% DMC without falsely claiming the ownership of the non-IP model.

7.2. RQ2: fidelity

In this section, we evaluate the potential side effects of FedRight on the main task performance. Fidelity requires that the IP protection is implemented without significant impact on the main task. We tested on three datasets with nine models to compare the difference in global model accuracy between normal federated training and federated training using the FedRight framework.

The experiments demonstrate that Fedright has excellent fidelity. The results are shown in Fig. 6, the global model that is with IP protection is consistent with the master task accuracy of the normally trained global model. The reason is that FedRight and the main task training process are independent of each other. FedRight does not actively change any parameters of the global model. It only uses the input and output of the model to realize the purpose of IP protection.

> ***Answer to RQ2:*** FedRight is independent of the main task training, thus it has a strong fidelity and does not have any side effects on the main task accuracy.

7.3. RQ3: robustness

In this section, we use FtuningAtt, PruningAtt and AdaptiveAtt to evaluate the robustness of FedRight, and also illustrate the CopEvaAtt.

FtuningAtt. FtuningAtt is a common strategy in practice, and we use 10% of the data from the test set to fine-tune the trained model to measure the robustness of FedRight. Fig. 7 shows that even after 100 epochs, FedRight still has the high accuracy of all models (only 19.50% drop in the worst case). The reason behind this may be that FtuningAtt does not produce significant changes in the weights of the model. Such modification that does not have significant side effects on the main task does not lead to a completely different model.

PruningAtt. It is well known that DNNs have many layers and many parameters, leading to the possible existence of redundant parameters. Therefore, a plagiarist can use PruningAtt to cut some redundant parameters and obtain a new model that is different from the original model but still has similar accuracy. Using a PruningAtt algorithm, some percentage of the parameters with the smallest absolute value is set to 0. The experiments compress the parameters by 30% to 90% with an interval of 10%. We then evaluate the accuracy of the model using the original test data to determine the impact on the original functionality of the model and also to assess the impact on FedRight. Ideally, a plagiarist would like to prune the stolen model and still maintain its performance.

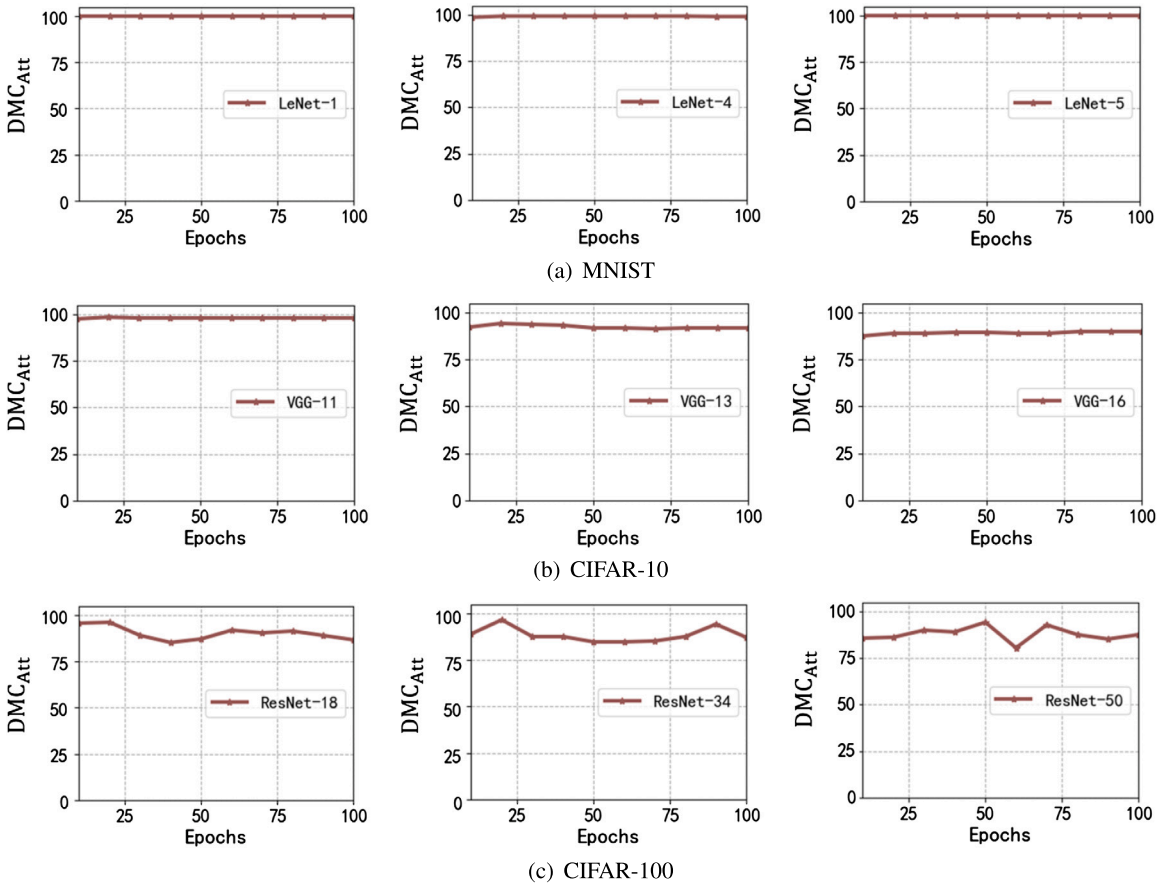


Fig. 7. The change in copyright verification accuracy in terms of DMC after performing fine-tuning using a 10% test set for the nine models.

Observation of the experimental results in Table 2. It can be seen that as the PruningAtt pruning percentage increases, it seriously affects the main task performance. Although the verification accuracy of the corresponding detector also gradually decreases, it still has the verification capability when the IP model still has high accuracy.

Even though some DMC of the detector has fallen below the threshold, we can see that the global model itself has also become of no value at that time. For example, for the three models on the CIFAR-10 dataset, when the global model is pruned by 90%, the DMC is lowered to 10% which is not able to determine the ownership anymore. However, the global model accuracy drops to 10% at the same time, which is of no meaning to protect.

CollabAtt. We design two experimental scenarios to satisfy the inter-collaboration of malicious parties.

$CollabAtt_{up}$. Upstream malicious client collaboration. Malicious clients participating in FL training can collaborate to eliminate model fingerprints by using local training data for adversarial retraining attacks.

$CollabAtt_{updo}$. Upstream and downstream malicious parties collaborative attack. In this model, the upstream malicious client performs adversarial retraining with local training data, and then the downstream malicious party attacks the trained global model (e.g. pruning, fine-tuning), thus they implement the collaborative attack by both the upstream and downstream attackers.

We test the effect of two collaborative attacks on different proportions of malicious clients. The experimental results are shown in Table 4. We find that FedRight remains robust in the face of collaborative attacks, the main threat to IP verification comes from downstream attackers. $CollabAtt_{up}$ cannot effectively affect the verification accuracy of the detector M. Facing $CollabAtt_{ftun}$ and $CollabAtt_{prun}$, the detector M verification accuracy has a certain decrease, but does not affect the effective judgment of the detector M. Besides, we study how the number of malicious clients affects the FedRight, and we find that the number of malicious clients does not affect our IP statement of the global model. Unlike the watermark embedding technique, most malicious clients can effectively remove the watermark embedded in the global model. In the FedRight framework, however, all the attacks done by the clients will only affect the performance of the global model, and the detector will be dynamically trained according to the changes in the global model to achieve the IP protection.

AdaptiveAtt. We designed AdaptiveAtt with different levels of adversary capability to test the robustness of IP verification.

AdaptiveAtt-1. In the FL scenario, it is difficult to obtain all the training data of the client, we assume that the attacker has 10% of the original training samples. They can generate adversarial examples using the original training samples, and perform adversarial retraining on the original model to eliminate fingerprints.

AdaptiveAtt-2. The attacker has the all key samples. Adversarial examples are generated using key samples. Then adversarial retraining of the original model is conducted to eliminate fingerprints.

AdaptiveAtt-3. The attacker has AdaptiveAtt-3 containing AdaptiveAtt-1 and AdaptiveAtt-2, which first eliminates the fingerprints by using adversarial retraining of the key sample and then retraining with the original training sample.

We tested on nine models, and the experimental results are shown in Table 3, where we found that the attacker using AdaptiveAtt-1 cannot remove the model fingerprint verification. And while using AdaptiveAtt-2 is effective in removing model fingerprints, it has a greater impact on the prediction performance of the main task, which renders the attack invalid since the main task performance is severely impaired. Using AdaptiveAtt-3, the model accuracy is improved by retraining the original training samples after the adversarial retraining

Table 2
Changes in global model accuracy and detector model accuracy after performing PruningAtt on the 9 models.

MNIST						
PruningAtt	LeNet-1 (Threshold = 18.00%)		LeNet-4 (Threshold = 15.50%)		LeNet-5 (Threshold = 13.14%)	
	GMC	DMC_{Att}	GMC	DMC_{Att}	GMC	DMC_{Att}
90%	50.33	45.00	93.53	58.19	72.39	46.60
80%	80.53	57.19	97.33	95.60	90.79	69.60
70%	91.18	65.20	97.32	98.40	92.36	87.40
60%	94.35	90.00	97.63	99.20	96.65	100.00
50%	95.37	98.20	97.98	100.00	96.84	100.00
40%	95.87	99.80	98.18	100.00	98.01	100.00
30%	96.15	100.00	98.28	100.00	98.46	100.00
CIFAR-10						
PruningAtt	VGG-11 (Threshold = 16.90%)		VGG-13 (Threshold = 14.20%)		VGG-16 (Threshold = 18.40%)	
	GMC	DMC_{Att}	GMC	DMC_{Att}	GMC	DMC_{Att}
90%	10.00	10.00	10.00	10.00	10.00	10.00
80%	10.00	10.00	10.00	10.00	10.00	10.00
70%	85.13	62.85	87.46	57.62	74.03	60.41
60%	86.13	100.00	88.18	99.52	88.77	95.71
50%	86.13	100.00	88.07	98.57	88.78	99.52
40%	86.03	100.00	88.12	99.52	88.89	99.52
30%	86.07	100.00	88.17	100.00	88.84	100.00
CIFAR-100						
PruningAtt	ResNet-18 (Threshold = 10.00%)		ResNet-34 (Threshold = 10.95%)		ResNet-50 (Threshold = 8.80%)	
	GMC	DMC_{Att}	GMC	DMC_{Att}	GMC	DMC_{Att}
90%	16.11	19.04	15.02	10.00	9.70	10.00
80%	38.40	20.47	39.04	19.04	40.69	26.66
70%	44.72	28.57	44.46	25.71	49.73	46.66
60%	47.18	61.42	47.34	75.71	52.49	86.19
50%	47.81	89.04	47.89	98.09	53.69	99.04
40%	48.47	100.00	48.17	99.52	53.87	100.00
30%	48.68	100.00	48.41	100.00	53.89	100.00

using AdaptiveAtt-2. However the model fingerprint can still be validated effectively.

CopEvaAtt. To evade the legitimate owner's verification, the attacker can try to construct a detector to detect whether the queried sample is a clean sample or possibly a fingerprinting sample. Once the detector determines that the queried instance is a fingerprinting sample, the attacker can diliberately return a random label.

However, the data of each client is kept locally and not shared during the federation learning process. It is also not sure that the data are independent and identically distributed (IID) among clients. It is undesirable for malicious clients to try to defend against copyright verification by CopEvaAtt. This detection method is more costly for the attacker and difficult to deploy in practice. The success rate of defending against copyright verification will come at the cost of more false positives, which reduces the utility of the attacker model for other users.

Answer to RQ3: FedRight shows strong robustness under model modification attacks including both FtuningAtt and PruningAtt. It maintains its validity in the face of AdaptiveAtts, i.e., it either maintains high verification effectiveness or invalids the purpose of stealing a model by lowering the accuracy to an undesired level.

7.4. RQ4: comparison with watermarking technology

In this section, we compare FedRight with the watermarking IP protection. The static generation of the adversarial example $FedRight_S$ is also added to the comparison experiment to illustrate the advantages of the dynamic generation of the adversarial example $FedRight_D$.

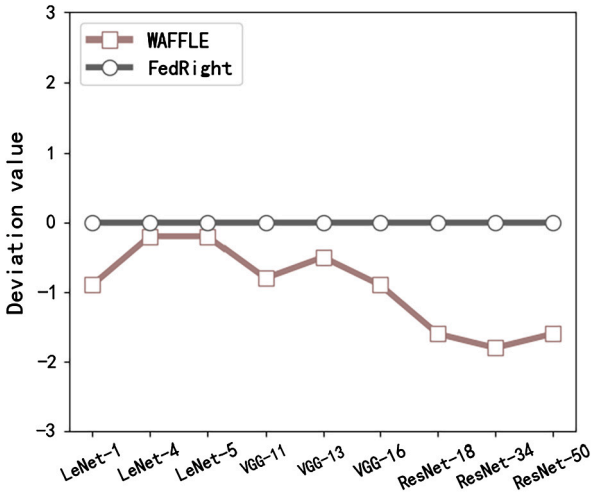


Fig. 8. The accuracy of the global models protected by FedRight and WAFFLE. It is measured by the deviation from the baseline which is the global model trained without any copyright protection.

7.4.1. Impact on main performance

RQ1 answers that FedRight has a natural advantage in terms of fidelity. To better confirm its fidelity, we performed a comparative evaluation of the watermark embedding techniques-WAFFLE and FedRight. The results are shown in Fig. 8, FedRight shows the same level of global model accuracy as the normal FL model, while the accuracy of the watermarked FL model decreases by 0.94% on average. The reason is that our FedRight IP protection is independent of the federated training process and does not have any impact on FL. The embedding of watermark changes the training settings of FL and needs to embed the watermark

Table 3
The experimental results of AdaptiveAtts where attackers have different levels of knowledge.

Knowledge	MNIST						CIFAR-10						CIFAR-100					
	LeNet-1		LeNet-4		LeNet-5		VGG-11		VGG-13		VGG-16		ResNet-18		ResNet-34		ResNet-50	
	GMC	DMC_{Att}	GMC	DMC_{Att}	GMC	DMC_{Att}	GMC	DMC_{Att}	GMC	DMC_{Att}	GMC	DMC_{Att}	GMC	DMC_{Att}	GMC	DMC_{Att}	GMC	DMC_{Att}
AdaptiveAtt-1	94.03	76.66	97.77	94.60	97.11	86.40	79.80	62.85	83.47	69.52	83.97	66.66	1.87	23.80	2.55	20.47	1.43	26.19
AdaptiveAtt-2	35.41	14.76	29.39	11.42	37.96	20.00	55.23	33.33	72.21	38.09	77.70	42.85	1.48	12.38	1.06	7.14	1.16	12.85
AdaptiveAtt-3	96.94	91.42	97.99	77.14	98.06	67.14	84.85	70.95	85.75	62.85	86.58	81.42	39.77	31.42	39.12	49.04	41.63	23.80

Table 4
The experimental results of CollabAtts where attackers have different ways of collaboration.

CollabAtt		40%						80%					
		$CollabAtt_{up}$		$CollabAtt_{ftun}$		$CollabAtt_{prun}$		$CollabAtt_{up}$		$CollabAtt_{ftun}$		$CollabAtt_{prun}$	
		GMC	DMC_{Att}	GMC	DMC_{Att}	GMC	DMC_{Att}	GMC	DMC_{Att}	GMC	DMC_{Att}	GMC	DMC_{Att}
MNIST	**LeNet-1**	96.78	100.00	96.93	98.57	94.07	90.95	97.18	100.00	96.85	81.90	87.16	71.42
	LeNet-4	98.24	100.00	97.89	95.71	97.32	94.28	98.68	100.00	98.32	93.33	98.05	94.28
	LeNet-5	98.52	100.00	98.47	97.62	97.89	88.57	98.65	100.00	98.10	96.67	97.34	82.85
CIFAR-10	**VGG-11**	86.37	100.00	85.84	98.57	85.91	100.00	85.24	100.00	85.54	96.67	84.89	100.00
	VGG-13	87.69	100.00	88.12	94.29	87.97	99.04	87.85	100.00	87.31	85.71	87.26	100.00
	VGG-16	88.01	100.00	88.83	99.52	87.99	100.00	88.92	100.00	88.29	97.62	88.51	100.00
CIFAR-100	**ResNet-18**	40.32	100.00	45.49	61.90	37.12	87.14	13.78	100.00	45.06	78.10	12.75	83.80
	ResNet-34	43.36	100.00	44.31	89.05	42.29	83.80	13.05	100.00	46.04	52.38	12.54	97.61
	ResNet-50	45.49	100.00	49.96	65.90	41.53	96.19	11.26	100.00	50.52	64.29	8.08	90.95

Table 5
The time overhead added to the main task FL training by WAFFLE and FedRight.

Method	Time(second)								
	epoch = 80			epoch = 100			epoch = 100		
	LeNet-1	LeNet-4	LeNet-5	VGG-11	VGG-13	VGG-16	ResNet-18	ResNet-34	ResNet-50
WAFFLE	424.46	462.75	480.50	1645.37	1825.40	2185.46	2415.42	2909.91	5500.70
FedRight	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**

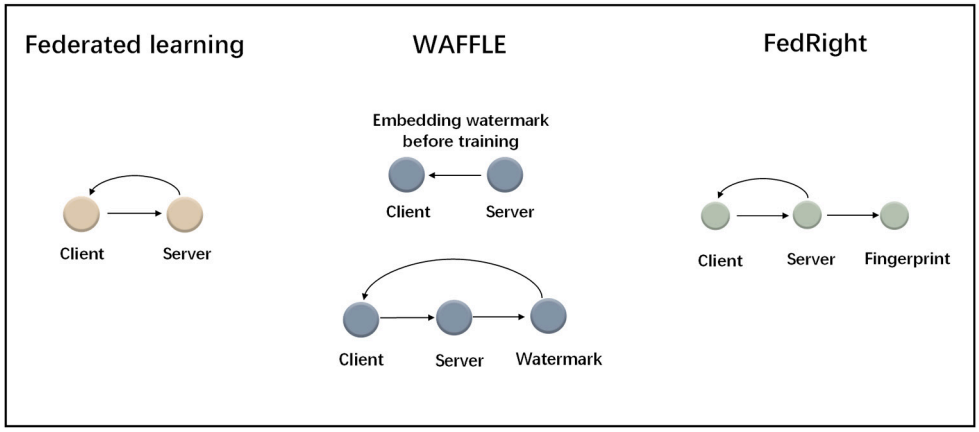


Fig. 9. Workflow illustrations of the vanilla FL, WAFFLE and FedRight.

image into the global model. Although with the development of technology, the impact from watermarking technology on the main task performance is gradually reduced, it is still difficult to eliminate this inherent defect.

7.4.2. Time overhead

We compare the time overhead caused by the WAFFLE and FedRight. As shown in Fig. 9, it can be observed intuitively that WAFFLE has changed the settings of the federated training. The server needs to embed the watermark after each epoch of aggregation operation, and then distribute the global model to each client. WAFFLE is at the expense of the main task training performance and efficiency. Differently, FedRight does not have any impact on the main task training, and the model fingerprinting is independent of the FL training. In Table 5 we give the detailed time overhead added by the two copyright protection mechanisms for federated training. WAFFLE adds an average of 1983.89 seconds of time overhead for all models. The time added on training a large dataset such as CIFAR100 is even more significant, with an average increase of 3608.68 seconds.

7.4.3. Ability to resist attacks

We consider two real-world scenarios in which attackers might execute the attack methods. (1) MaliClientAtt: upstream attackers perform the removal attack to interfere the copyright protection implementation during training. (2) Downstream attackers use FtuningAtt and PruningAtt to clean the global model.

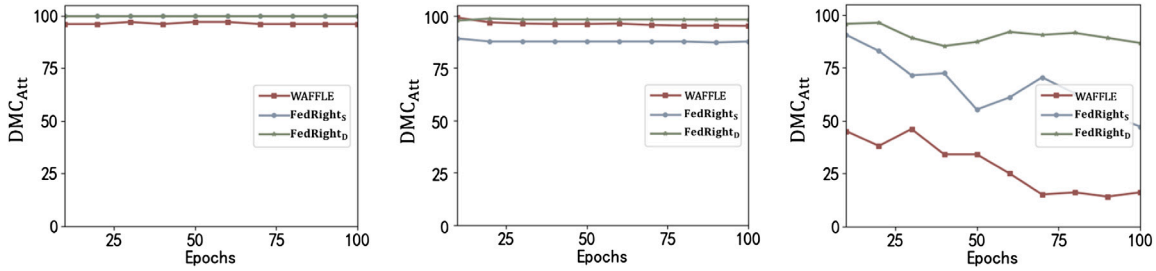
MaliClientAtt: In the federation training process, we set up 40% and 80% malicious clients to perform watermark removal attack method on the global model, respectively. The experimental results are shown in Table 7. Facing the removal attack, the overall watermark verification accuracy of WAFFLE has been lowered to less than 30%. Even worse, the verification has been failed on the CIFAR-100-based model. The reason may be that for the main task with more data classes, the difficulty of embedding watermarks may increase, leading to easier attacks.

At present, watermark removal attacks and watermark overlay attacks are becoming more and more advanced. FedRight protects IP based on model fingerprints. It does not introduce watermarks to model internal parameters, so it effectively avoids various attacks against wartermarking.

We then compare the robustness of WAFFLE and FedRight against FtuningAtt and PruningAtt. The experimental results of FtuningAtt are shown in Fig. 10, and the experimental results of PruningAtt are shown in Table 6. Compared with the watermark embedding technique WAFFLE, $FedRight_D$ shows its own advantages, In the face of FtuningAtt and PruningAtt, the overall average verification accuracy of $FedRight_D$

Table 6
Comparison of the robustness of WAFFLE and FedRight against PruningAtt.

PruningAtt	LeNet-5						VGG-11						ResNet-18					
	WAFFLE		FedRight_S		FedRight_D		WAFFLE		FedRight_S		FedRight_D		WAFFLE		FedRight_S		FedRight_D	
	GMC	DMC_{Att}	GMC	DMC_{Att}	GMC	DMC_{Att}	GMC	DMC_{Att}	GMC	DMC_{Att}	GMC	DMC_{Att}	GMC	DMC	GMC	DMC_{Att}	GMC	DMC_{Att}
90%	72.42	44.00	60.74	35.71	72.39	**46.60**	10.00	10.00	10.00	10.00	10.00	**10.00**	14.34	0.00	15.78	10.00	16.11	**19.04**
80%	90.72	68.00	89.63	67.33	90.79	**69.60**	10.00	10.00	10.00	10.00	10.00	**10.00**	38.20	1.00	10.00	10.00	38.40	**20.47**
70%	90.64	87.00	92.43	86.35	92.36	**87.40**	83.62	43.80	84.97	59.80	85.13	**62.85**	44.52	11.24	45.55	14.28	44.72	**28.57**
60%	96.21	100.00	97.66	100.00	96.65	**100.00**	84.52	62.04	85.52	99.04	86.13	**100.00**	45.73	30.46	47.04	31.42	47.18	**61.42**
50%	96.89	100.00	98.15	100.00	96.84	**100.00**	85.12	78.83	85.48	99.52	86.13	**100.00**	46.24	67.35	47.25	86.95	47.81	**89.04**
40%	98.10	100.00	98.49	100.00	98.01	**100.00**	85.25	92.51	85.47	100.00	86.03	**100.00**	46.63	93.18	47.76	99.52	48.47	**100.00**
30%	98.25	100.00	98.46	100.00	98.46	**100.00**	85.31	100.00	85.51	100.00	86.07	**100.00**	47.35	100.00	48.71	100.00	48.68	**100.00**



(a) LeNet-5 (b) VGG-11 (c) ResNet-18

Fig. 10. The accuracy of IP verification changes during PruningAtt for 100 epochs using 10% of the training set.

Table 7
The impact of 40% and 80% malicious clients performing removal attack during training.

Model		40%			80%		
		WAFFLE	FedRight_S	FedRight_D	WAFFLE	FedRight_S	FedRight_D
DMC_{Att}	**LeNet-1**	25.31	**100.00**	**100.00**	14.31	**100.00**	**100.00**
	LeNet-4	24.68	**100.00**	**100.00**	12.68	**100.00**	**100.00**
	LeNet-5	25.42	**100.00**	**100.00**	12.42	**100.00**	**100.00**
	VGG-11	20.41	**100.00**	**100.00**	10.68	**100.00**	**100.00**
	VGG-13	19.04	**100.00**	**100.00**	9.54	**100.00**	**100.00**
	VGG-16	21.52	**100.00**	**100.00**	11.52	**100.00**	**100.00**
	ResNet-18	0.00	**100.00**	**100.00**	0.00	**100.00**	**100.00**
	ResNet-34	0.00	**100.00**	**100.00**	0.00	**100.00**	**100.00**
	ResNet-50	0.00	**100.00**	**100.00**	0.00	**100.00**	**100.00**

is higher than that of WAFFLE by 12.64% and 9.79%, respectively. We also compare FedRight_S with FedRight_D, because the parameters of the global model are updated in each training epoch, the FedRight_S may cause the fingerprints generated in the previous epoch to be invalid in the next epoch without adaptive capability. The FedRight_D, on the other hand, effectively solves this problem. The validity of the model fingerprint is checked using Algorithm (4) so that the model features of the current epoch can be added adaptively to achieve an enhanced fingerprint. This makes the overall verification accuracy of FedRight_D higher than FedRight_S by 11.32% and 4.05% on average, respectively, in the face of FtuningAtt and PruningAtt.

> ***Answer to RQ4:*** The FedRight technology based on model fingerprints effectively makes up for the inherent defects of watermarking technology and improves the performance by 11.22%. The dynamically enhanced FedRight_D effectively solves the fingerprint invalidation problem and improves the performance by 7.69% on average compared to FedRight_S.

7.5. RQ5: parameter sensitivity

In this section, we conduct experiments to test whether the number of clients participating in training, the non-independent and identically distributed (Non-IID) data distribution and hyperparameter settings have an impact on the effectiveness of FedRight.

7.5.1. Number of clients

For the federated learning scenario with a large number of clients trained, we conduct experiments on the sensitivity of the parameters for the number of clients $K \in \{5, 20, 100\}$ in the three models. It can be intuitively found from Table 8 that the number of clients K in FL training mainly affects the global model accuracy GMC because the aggregation algorithm is related to K. And it has no effect on the detector verification accuracy DMC, the reason is that the detector is trained based on the output features of the model fingerprint, and the output features will change dynamically according to the global model. Therefore the detector will dynamically adapt to the changes in the global model. FedRight also designs the adaptive enhancement capability of the model fingerprint to stabilize the output features, and the global model accuracy reaches a certain level, the impact on the change of output features also becomes smaller, which makes the detector a good training environment and ensures the validation capability of the detector.

Table 8
The impact of the number of clients K on FedRight.

		LeNet-5	VGG-11	ResNet-18
$K=5$	**GMC**	98.12	85.86	51.26
	DMC	100.00	100.00	100.00
$K=20$	**GMC**	98.99	87.83	51.71
	DMC	100.00	100.00	100.00
$K=100$	**GMC**	98.57	87.31	56.60
	DMC	100.00	100.00	100.00

We select three models to test the robustness of the detector for $K \in \{5, 20, 100\}$ under the three AdaptiveAtt settings, and the results are shown in Table 9. The experiments show that the models after being attacked can still be effectively verified by FedRight for their property rights while ensuring their own accuracy GMC. For example, at $K=5$, LeNet-5 maintains a GMC above 97% when facing AdaptiveAtt-1 and AdaptiveAtt-3, so the DMC_{Att} is well above the threshold value, averaging at 76.77%. And facing AdaptiveAtt-2, the GMC drops sharply, causing the DMC to drop as well. The reason for this is that the output features of the model fingerprint do not change dramatically when the property rights model guarantees its own accuracy, allowing the detector to effectively identify the output features of the model fingerprint. Comparing the experimental results under different number K of clients, DMC_{Att} does not show an obvious regular change due to the increase of number K, which further illustrates that the robustness of the detector is not affected by the number of clients.

7.5.2. Data distribution sensitivity

In real-world scenarios, the data among clients is heterogeneous, we consider using Dirichlet distribution to divide the training data among clients. Specifically, we sample $Dir(\beta)$ and divide the dataset according

Table 9
The impact of the number of clients K on the robustness of FedRight.

Knowledge	$K=5$						$K=20$						$K=100$					
	LeNet-5		VGG-11		ResNet-18		LeNet-5		VGG-11		ResNet-18		LeNet-5		VGG-11		ResNet-18	
	GMC	DMC_{Att}	GMC	DMC_{Att}	GMC	DMC_{Att}	GMC	DMC_{Att}	GMC	DMC_{Att}	GMC	DMC_{Att}	GMC	DMC_{Att}	GMC	DMC_{Att}	GMC	DMC_{Att}
AdaptiveAtt-1	97.11	86.40	79.80	62.85	1.87	23.80	98.31	99.52	84.43	86.66	2.10	28.57	93.13	65.71	84.22	62.85	1.72	43.33
AdaptiveAtt-2	37.96	20.00	55.23	33.33	1.48	12.38	47.65	12.38	62.84	57.14	0.97	32.85	23.20	5.23	60.73	31.90	1.18	30.00
AdaptiveAtt-3	98.06	67.14	84.85	70.95	39.77	31.42	98.36	56.19	86.71	74.28	42.76	49.04	97.22	60.00	87.41	87.61	36.30	26.66

Table 10
The influence of the data distribution parameter β on FedRight.

		LeNet-5	VGG-11	ResNet-18
$\beta=0.1$	**GMC**	94.01	77.29	38.72
	DMC	100.00	100.00	100.00
$\beta=0.5$	**GMC**	96.18	83.33	48.92
	DMC	100.00	100.00	100.00
$\beta=1.0$	**GMC**	98.12	85.86	51.26
	DMC	100.00	100.00	100.00

Table 11
The influence of the key samples on FedRight.

		LeNet-5	VGG-11	ResNet-18
GMC		98.12	85.86	51.26
Scenic samples	**DMC**	100.00	100.00	100.00
Face samples	**DMC**	100.00	100.00	100.00
Traffic instruction samples	**DMC**	100.00	100.00	100.00

Table 12
Different adversarial example generation methods for generating model fingerprints.

		LeNet-5	VGG-11	ResNet-18
DMC	**C&W**	100	100	100
	PGD	100	100	100
	FGSM	100	100	100

to the distribution of concentration parameter β, and then assign them to each client. $Dir(\beta)$ is the Dirichlet distribution with β. With the above partitioning strategy, the gap between the number of categories and samples owned by the clients grows as the β value decreases. We use $\beta=0.1$, $\beta=0.5$ and $\beta=1$ in the experiments to explore the sensitivity of data distribution to property rights protection.

From the experimental results of Table 10, it is found that FedRight is not affected by differences in data distribution. The reason is that in the process of generating the model fingerprint, the fingerprint will change with the change of the global model. However, the accuracy of global GMC is decreased as the value of β decreases. This is the widely acknowledged issue faced by FL where the data among clients is Non-IID distribution.

7.5.3. Key samples

We collect three key samples different from the federated training data for generating model fingerprints. The experimental results are shown in Table 11. The FedRight is not sensitive to the key samples, and the detector M can all achieve 100% validity. The reason is that the key samples are only part of the carriers of the model fingerprints. The model fingerprint is formed by extracting the unique features of the model and adding them to the carrier, in which the unique features are more important than the carrier. Therefore, the type of key sample does not affect the validity of the model fingerprint.

7.5.4. Adversarial example generation methods

This experiment is to explore whether other adversarial example generation methods can also be used in FedRight. As shown in Table. 12, when using FGSM, C&W, and PGD targeted attacks to generate the model fingerprints, FedRight can also achieve good performance.

7.5.5. Hyperparameter sensitivity

In this section we discuss the impact of the hyperparameters ϵ and c on FedRight.

When generating adversarial examples, the hyperparameter ϵ is used to limit the extracted intrinsic features of the model. We explore the effect of hyperparameters on property rights protection by setting

Table 13
Hyperparameter sensitivity analysis of $\epsilon=0.01$, $\epsilon=0.08$, and $\epsilon=0.80$.

		LeNet-5	VGG-11	ResNet-18
DMC	$\epsilon=$**0.01**	100	100	100
	$\epsilon=$**0.08**	100	100	100
	$\epsilon=$**0.80**	100	100	100

$\epsilon=0.01$, $\epsilon=0.08$, and $\epsilon=0.8$. The results are shown in Table. 13. We can see the impact from ϵ is limited on FedRight performance.

Regarding the sensitivity of selecting the specific number of key sample classes c, We test under three models, LeNet-5, VGG-11 and ResNet-18, setting $c = 5$, 7, and 10, the elevation rate of DMC_{Att} obtained in the face of PruningAtt compared to the average accuracy of random guesses (100/c). As shown in Table 14, at PruningAtt is 0%, it can be found that the number of classes does not seriously affect the validity of the IP validation. However, the elevation rate of DMC_{Att} can be effectively improved by increasing c compared with the average accuracy of random guesses.

	Pruning attack					Non-IP model
	50%	60%	70%	80%	90%	
GMC	96.44	94.25	87.35	81.79	67.36	96.21
DMC	100.00	100.00	100.00	100.00	100.00	100.00

Answer to RQ5: The effectiveness of FedRight is not affected by the number of clients, data distribution and hyperparameter values.

8. Discussion

With the widespread use of federated learning, the federated models are becoming more and more commercially valuable, and how to protect the property rights of federated models and prevent illegal profits after copying and stealing has practical significance. Our proposed FedRight framework can be effectively applied to several real-world scenarios, such as news recommendation systems, which have high business value and people benefit from personalized push messages from the recommendation system to meet their needs. In the FedRight framework, each user is a participant and the central server is the initiator of the IP declaration, which dynamically trains the detector according to the changes of the recommender system to achieve property rights protection.

In the FedRight framework, the server holds the whole process of federated aggregation operation, model fingerprint generation and detector training. If the server is controlled by a malicious party, the model fingerprint generation and detector training will be attacked, such as generating wrong model fingerprints, randomly generating output features for detector training, etc., which will lead to the loss of FedRight's model ownership verification capability. Therefore, defending against malicious server attacks is of practical importance. There have been many defense techniques against malicious servers, such as secure multi-party computation (Li et al., 2021), homomorphic encryption (Zhang et al., 2020), and trusted execution environment (TEE) (Mo et al., 2021). For FedRight framework, the TEE can be utilized to target the server vulnerability problem through the technique of hardware isolation.TEE is a separate processing environment with computing and storage capabilities that provides security and integrity protection. This area provides a more secure space for data and code execution. We can place key samples and detectors in it so that model fingerprint generation and detector training have a secure execution environment and malicious servers cannot access the execution logic and data in this area, guaranteeing FedRight's ability to verify model ownership.

Table 14
Hyperparameter sensitivity analysis of $c = 5$, 7 and 10.

PruningAtt	Elevation rate(%)								
	LeNet-5			VGG-11			ResNet-18		
	C=5	C=7	C=10	C=5	C=7	C=10	C=5	C=7	C=10
90%	0.40	0.77	**0.79**	0.00	**0.02**	0.00	-0.05	0.02	**0.47**
60%	0.80	0.86	**0.90**	0.80	0.86	**0.90**	0.74	0.76	**0.84**
30%	0.80	0.86	**0.90**	0.80	0.86	**0.90**	0.80	0.86	**0.90**
0%	0.80	0.86	**0.90**	0.80	0.86	**0.90**	0.80	0.86	**0.90**

9. Conclusion

This paper presents FedRight, the FL copyright protection technique based on model fingerprints. FedRight uses the features of the model to generate the model fingerprints with adaptive enhancement. Using the distribution features of the model fingerprint input by the model, a detector is trained to claim the ownership of a suspect model. FedRight does not require any training data, which is especially designed for FL scenario. Moreover, FedRight only requires the input and output of the suspect model to verify the copyright, which fits the practical situation where model owner does not have access to the deployed stolen model. We extensively evaluate FedRight on nine models on three benchmark datasets to demostrate that it achieves remarkable results in terms of fidelity, effectiveness, and robustness. To the best of our knowledge, we are the first to apply model fingerprints techniques to FL. In future work, we plan to adapt FedRight to other high-value models such as speech recognition, and extend to other forms of deep learning architectures, such as recurrent neural networks. It is interesting and challenging to verify the attribution of the model by comparing the features of the suspicious model and the original model, and it has important implications for future research, which we will investigate in our future work. Meanwhile, more comprehensively consider the case where the client is the initiator of property protection and the server has the possibility of being maliciously controlled.

CRediT authorship contribution statement

Jinyin Chen: Conceptualization, Funding acquisition, Methodology, Supervision, Writing – review & editing. **Mingjun Li:** Investigation, Methodology, Resources, Software, Writing – original draft. **Yao Cheng:** Formal analysis, Visualization, Writing – review & editing. **Haibin Zheng:** Investigation, Methodology, Project administration, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgements

This research is supported by the National Natural Science Foundation of China (No. 62072406), Zhejiang Provincial Natural Science Foundation (No. LDQ23F020001), Chinese National Key Laboratory of Science and Technology on Information System Security (No. 61421110502), and National Key R&D Projects of China (No. 2018AAA0100801).

References

McMahan, B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A., 2017. Communication-efficient learning of deep networks from decentralized data. In: Proceedings of the 20th International Conference on Artificial Intelligence and Statistics. AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA. In: Ser. Proceedings of Machine Learning Research, vol. 54. PMLR, pp. 1273–1282. [Online]. Available: http://proceedings.mlr.press/v54/mcmahan17a.html.

McMahan, H.B., Moore, E., Ramage, D., y Arcas, B.A., 2016. Federated learning of deep networks using model averaging. CoRR. arXiv:1602.05629 [abs]. [Online]. Available: http://arxiv.org/abs/1602.05629.

Yang, Q., Liu, Y., Chen, T., Tong, Y., 2019. Federated machine learning: concept and applications. ACM Trans. Intell. Syst. Technol. 10 (2), 12. https://doi.org/10.1145/3298981. [Online].

Blanco-Justicia, A., Domingo-Ferrer, J., Martínez, S., Sánchez, D., Flanagan, A., Tan, K.E., 2021. Achieving security and privacy in federated learning systems: survey, research challenges and future directions. Eng. Appl. Artif. Intell. 106, 104468. https://doi.org/10.1016/j.engappai.2021.104468. [Online].

Jiang, C., Xu, C., Zhang, Y., 2021. PFLM: privacy-preserving federated learning with membership proof. Inf. Sci. 576, 288–311. https://doi.org/10.1016/j.ins.2021.05.077. [Online].

Wang, F., Zhu, H., Lu, R., Zheng, Y., Li, H., 2021. A privacy-preserving and non-interactive federated learning scheme for regression training with gradient descent. Inf. Sci. 552, 183–200. https://doi.org/10.1016/j.ins.2020.12.007. [Online].

Zhang, C., Xie, Y., Bai, H., Yu, B., Li, W., Gao, Y., 2021. A survey on federated learning. Knowl.-Based Syst. 216, 106775. https://doi.org/10.1016/j.knosys.2021.106775. [Online].

Shingi, G., 2020. A federated learning based approach for loan defaults prediction. In: Fatta, G.D., Sheng, V.S., Cuzzocrea, A., Zaniolo, C., Wu, X. (Eds.), 20th International Conference on Data Mining Workshops, ICDM Workshops 2020. Sorrento, Italy, November 17-20, 2020. IEEE, pp. 362–368. [Online].

Kuo, T., Pham, A., 2022. Detecting model misconducts in decentralized healthcare federated learning. Int. J. Med. Inform. 158 (February), 104658. https://doi.org/10.1016/j.ijmedinf.2021.104658. [Online].

Wahab, O.A., Rjoub, G., Bentahar, J., Cohen, R., 2022. Federated against the cold: a trust-based federated learning approach to counter the cold start problem in recommendation systems. Inf. Sci. 601, 189–206. https://doi.org/10.1016/j.ins.2022.04.027. [Online].

Uchida, Y., Nagai, Y., Sakazawa, S., Satoh, S., 2017. Embedding watermarks into deep neural networks. In: Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval. CMR 2017, Bucharest, Romania, June 6-9, 2017. ACM, pp. 269–277. [Online].

Vybornova, Y.D., 2021. Method for copyright protection of deep neural networks using digital watermarking. In: Fourteenth International Conference on Machine Vision. ICMV 2021, Rome, Italy, November 8–12, 2021. In: Ser. SPIE Proceedings, vol. 12084. SPIE, p. 1208412. [Online].

Li, M., Zhong, Q., Zhang, L.Y., Du, Y., Zhang, J., Xiang, Y., 2020. Protecting the intellectual property of deep neural networks with watermarking: the frequency domain approach. In: 19th IEEE International Conference on Trust, Security and Privacy in Computing and Communications. TrustCom 2020, Guangzhou, China, December 29, 2020 - January 1, 2021. IEEE, pp. 402–409. [Online].

Guo, J., Potkonjak, M., 2018. Watermarking deep neural networks for embedded systems. In: Proceedings of the International Conference on Computer-Aided Design. ICCAD 2018, San Diego, CA, USA, November 05-08, 2018. ACM, p. 133. [Online].

Li, Z., Hu, C., Zhang, Y., Guo, S., 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. ACSAC 2019, San Juan, PR, USA, December 09-13, 2019. ACM, pp. 126–137. [Online].

Zhao, J., Hu, Q., Liu, G., Ma, X., Chen, F., Hassan, M.M., 2020. AFA: adversarial fingerprinting authentication for deep neural networks. Comput. Commun. 150, 488–497. https://doi.org/10.1016/j.comcom.2019.12.016. [Online].

Lukas, N., Zhang, Y., Kerschbaum, F., 2021. Deep neural network fingerprinting by conferrable adversarial examples. In: 9th International Conference on Learning Representations. ICLR 2021, Virtual Event, Austria, May 3-7, 2021. OpenReview.net. [Online]. Available: https://openreview.net/forum?id=VqzVhqxkjH1.

Cao, X., Jia, J., Gong, N.Z., 2021. Ipguard: protecting intellectual property of deep neural networks via fingerprinting the classification boundary. In: ASIA CCS '21: ACM Asia

Conference on Computer and Communications Security, Virtual Event. Hong Kong, June 7-11, 2021. ACM, pp. 14–25. [Online].

Tekgul, B.G.A., Xia, Y., Marchal, S., Asokan, N., 2021. WAFFLE: watermarking in federated learning. In: 40th International Symposium on Reliable Distributed Systems. SRDS 2021, Chicago, IL, USA, September 20-23, 2021. IEEE, pp. 310–320. [Online].

Li, F., Wang, S., Liew, A.W., 2022. Watermarking protocol for deep neural network ownership regulation in federated learning. In: IEEE International Conference on Multimedia and Expo Workshops, ICME Workshops 2022. Taipei, Taiwan, July 18-22, 2022. IEEE, pp. 1–4. [Online].

Fan, L., Li, B., Gu, H., Li, J., Yang, Q., 2021. Fedipr: ownership verification for federated deep neural network models. CoRR. arXiv:2109.13236 [abs]. [Online]. Available: https://arxiv.org/abs/2109.13236.

Luo, B., Liu, Y., Wei, L., Xu, Q., 2018. Towards imperceptible and robust adversarial example attacks against neural networks. In: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th Innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18). New Orleans, Louisiana, USA, February 2-7, 2018. AAAI Press, pp. 1652–1659. [Online]. Available: https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16217.

Huang, S.H., Papernot, N., Goodfellow, I.J., Duan, Y., Abbeel, P., April 24-26, 2017. Adversarial attacks on neural network policies. In: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings. OpenReview.net. [Online]. Available: https://openreview.net/forum?id=ryvlRyBKl.

Zheng, H., Zhang, Z., Gu, J., Lee, H., Prakash, A., 2020. Efficient adversarial training with transferable adversarial examples. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition. CVPR 2020, Seattle, WA, USA, June 13-19, 2020. Computer Vision Foundation / IEEE, pp. 1178–1187. [Online]. Available: https://openaccess.thecvf.com/content_CVPR_2020/html/Zheng_Efficient_Adversarial_Training_With_Transferable_Adversarial_Examples_CVPR_2020_paper.html.

Wei, L., Luo, B., Li, Y., Liu, Y., Xu, Q., 2018. I know what you see: power side-channel attack on convolutional neural network accelerators. In: Proceedings of the 34th Annual Computer Security Applications Conference. ACSAC 2018, San Juan, PR, USA, December 03-07, 2018. ACM, pp. 393–406. [Online].

Jia, H., Choquette-Choo, C.A., Chandrasekaran, V., Papernot, N., 2021. Entangled watermarks as a defense against model extraction. In: 30th USENIX Security Symposium, USENIX Security 2021. August 11-13, 2021. USENIX Association, pp. 1937–1954. [Online]. Available: https://www.usenix.org/conference/usenixsecurity21/presentation/jia.

Hitaj, D., Hitaj, B., Mancini, L.V., 2019. Evasion attacks against watermarking techniques found in mlaas systems. In: 6th International Conference on Software Defined Systems. SDS 2019, Rome, Italy, June 10-13, 2019. IEEE, pp. 55–63. [Online].

Shafieinejad, M., Wang, J., Lukas, N., Kerschbaum, F., 2019. On the robustness of the backdoor-based watermarking in deep neural networks. CoRR. arXiv:1906.07745 [abs]. [Online]. Available: http://arxiv.org/abs/1906.07745.

Rouhani, B.D., Chen, H., Koushanfar, F., 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. ASPLOS 2019, Providence, RI, USA, April 13-17, 2019. ACM, pp. 485–497. [Online].

Namba, R., Sakuma, J., 2019. Robust watermarking of neural network with exponential weighting. In: Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security. AsiaCCS 2019, Auckland, New Zealand, July 09-12, 2019. ACM, pp. 228–240. [Online].

Goodfellow, I.J., Shlens, J., Szegedy, C., 2015. Explaining and harnessing adversarial examples. In: 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings. [Online]. Available: http://arxiv.org/abs/1412.6572.

Carlini, N., Wagner, D.A., 2016. Towards evaluating the robustness of neural networks. CoRR. arXiv:1608.04644 [abs]. [Online]. Available: http://arxiv.org/abs/1608.04644.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A., 2018. Towards deep learning models resistant to adversarial attacks. In: 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings. OpenReview.net. [Online]. Available: https://openreview.net/forum?id=rJzIBfZAb.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., 1998. Gradient-based learning applied to document recognition. Proc. IEEE 86 (11), 2278–2324. https://doi.org/10.1109/5.726791. [Online].

Ayi, M., El-Sharkawy, M., 2020. Rmnv2: reduced mobilenet V2 for CIFAR10. In: 10th Annual Computing and Communication Workshop and Conference. CCWC 2020, Las Vegas, NV, USA, January 6-8, 2020. IEEE, pp. 287–292. [Online].

El-Sawy, A., El-Bakry, H.M., Loey, M., 2016. CNN for handwritten Arabic digits recognition based on lenet-5. In: Proceedings of the International Conference on Advanced Intelligent Systems and Informatics. AISI 2016, Cairo, Egypt, October 24-26, 2016. In: Ser. Advances in Intelligent Systems and Computing, vol. 533, pp. 566–575. [Online].

Chen, H., 2022. Reliable and efficient distributed machine learning. Ph.D. dissertation. Royal Institute of Technology, Stockholm, Sweden. [Online]. Available: https://nbn-resolving.org/urn:nbn:se:kth:diva-310374.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016. IEEE Computer Society, pp. 770–778. [Online].

Wu, W., 2021. Towards efficient horizontal federated learning. Ph.D. dissertation. University of Warwick, Coventry, UK. [Online]. Available: https://ethos.bl.uk/OrderDetails.do?uin=uk.bl.ethos.856355.

Shafieinejad, M., Lukas, N., Wang, J., Li, X., Kerschbaum, F., June, 22-25, 2021. On the robustness of backdoor-based watermarking in deep neural networks. In: IH&MMSec '21: ACM Workshop on Information Hiding and Multimedia Security, Virtual Event. Belgium, June, 22-25, 2021. ACM, pp. 177–188. [Online].

Sun, S., Wang, H., Xue, M., Zhang, Y., Wang, J., Liu, W., 2021. Detect and remove watermark in deep neural networks via generative adversarial networks. In: Information Security - 24th International Conference, ISC 2021, Virtual Event, November 10-12, 2021, Proceedings. In: Ser. Lecture Notes in Computer Science, vol. 13118. Springer, pp. 341–357. [Online].

Chattopadhyay, N., Viroy, C.S.Y., Chattopadhyay, A., 2020. Re-markable: stealing watermarked neural networks through synthesis. In: Security, Privacy, and Applied Cryptography Engineering - 10th International Conference, SPACE 2020, Kolkata, India, December 17-21, 2020, Proceedings. In: Ser. Lecture Notes in Computer Science, vol. 12586. Springer, pp. 46–65. [Online].

Singla, S., Singla, S., Feizi, S., 2022. Improved deterministic l2 robustness on CIFAR-10 and CIFAR-100. In: The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event. April 25-29, 2022. OpenReview.net. [Online]. Available: https://openreview.net/forum?id=tD7eCtaSkR.

Li, Y., Zhou, Y., Jolfaei, A., Yu, D., Xu, G., Zheng, X., 2021. Privacy-preserving federated learning framework based on chained secure multiparty computing. IEEE Int. Things J. 8 (8), 6178–6186. https://doi.org/10.1109/JIOT.2020.3022911. [Online].

Zhang, C., Li, S., Xia, J., Wang, W., Yan, F., Liu, Y., 2020. Batchcrypt: efficient homomorphic encryption for cross-silo federated learning. In: Gavrilovska, A., Zadok, E. (Eds.), 2020 USENIX Annual Technical Conference. USENIX ATC 2020, July 15-17, 2020. USENIX Association, pp. 493–506. [Online]. Available: https://www.usenix.org/conference/atc20/presentation/zhang-chengliang.

Mo, F., Haddadi, H., Katevas, K., Marin, E., Perino, D., Kourtellis, N., 2021. PPFL: privacy-preserving federated learning with trusted execution environments. In: Banerjee, S., Mottola, L., Zhou, X. (Eds.), MobiSys '21: The 19th Annual International Conference on Mobile Systems, Applications, and Services, Virtual Event. Wisconsin, USA, 24 June, 2 July, 2021. ACM, pp. 94–108. [Online].

Jinyin Chen received BS and PhD degrees from Zhejiang University of Technology, Hangzhou, China, in 2004 and 2009, respectively. She studied evolutionary computing in Ashikaga Institute of Technology, Japan in 2005 and 2006. She is currently a professor in the Institute of Cyberspace Security and the College of Information Engineering, Zhejiang University of Technology. Her research interests include artificial intelligence security, graph data mining and evolutionary computing.

Mingjun Li is a MS student at the college of Information Engineering, Zhejiang University of Technology. His research interests include deep learning, artificial intelligence attack and defense.

Yao Cheng, born in 1987. PhD, senior researcher in Digital Service, TUV SUD Asia Pacific Pte. Ltd. Her main research interests include in deep learning and privacy security systems, blockchain technology applications, Android framework vulnerability analysis.

Haibin Zheng received B.S. and Ph.D. degrees from Zhejiang University of Technology, Hangzhou, China, in 2017 and 2022, respectively. He is currently a university lecturer at the Institute of Cyberspace Security, Zhejiang University of Technology. His research interests include deep learning and artificial intelligence security.