

FedMLC: White-box Model Watermarking for Copyright Protection in Federated Learning for IoT Environment

Weitong Chen[✉], *Member, IEEE*, Wei Zhang[✉], Di Wu[✉], *Senior Member, IEEE*, Anja Keskinarkaus[✉],
Tapio Seppänen[✉], Jiale Zhang[✉], *Member, IEEE*,
Longxiang Gao[✉], *Senior Member, IEEE*, Tom H. Luan[✉], *Senior Member, IEEE*

Abstract—With the widespread application of the Internet of Things (IoT), data processing has gradually migrated to edge devices that are closer to the data source. This shift has significantly improved the ability of real-time data analysis while effectively reducing bandwidth requirements and latency. Furthermore, Federated Learning (FL) has been introduced as a decentralized training method to achieve collaborative training of multiple devices while ensuring local data privacy. However, malicious clients in FL may theft trained models for unauthorized use, which causes model misuse or copyright challenges. To address these issues, this paper proposes FedMLC (*Malicious client detection, Leakage tracing, and Copyright verification*), a server-side white-box watermarking scheme. FedMLC utilizes the embedded watermark at different stages to achieve both traceability and copyright verification, simplifying the watermarking process. Additionally, the watermarking can also detect malicious clients in FL. Specifically, FedMLC uses the regularization term to guide the parameter signs of the normalization layer to be consistent with the watermark sign, thereby achieving watermark embedding. Experimental results show that our FL model watermarking scheme excels in malicious client detection, leakage tracing, and copyright verification, with minimal impact on model performance, able to resist various attacks such as fine-tuning, pruning, and quantization.

Index Terms—Federated Learning, IoT, Model watermarking, Malicious client detection, Leakage tracing, Copyright verifica-

tion

I. INTRODUCTION

IN the era of intelligence, networking, and data-driven technologies, the Internet of Things (IoT) has developed rapidly. Due to the wide distribution of IoT devices [1, 2], data is often scattered across various edge devices. Federated Learning (FL) [3, 4] has become an essential method for data processing and intelligent analysis in IoT environments with privacy-preserving between edge devices. As a decentralized machine learning approach, FL enables these devices to collaboratively train models while keeping data local, thereby making effective use of distributed data resources. However, as the number of participating devices grows, issues related to model copyright have become increasingly prominent. Especially when models are updated locally, we cannot supervise whether each edge device has made unauthorized copies [5] or whether malicious modifications have been made to the model. These attacks can degrade model performance or cause copyright infringement [6]. Additionally, once the model is leaked [7–9], tracing the source of the leak in such a distributed architecture becomes extremely difficult. Therefore, ensuring copyright verification and detecting and tracing malicious clients during the FL process in IoT environments are urgent problems that need to be addressed.

Watermarking techniques have been widely explored to protect the copyright of Deep Neural Networks(DNN) models [10]. These techniques embed specific identification information in the model’s parameters, structure, or input and output, making the model unique. In this way, even if the model is copied, tampered with, or stolen, the watermark can be used to verify its original owner and legitimacy [11–13], thus maintaining the model’s copyright. In centralized learning, model watermarking methods are mainly divided into white-box watermarking [14–16] and black-box watermarking [17–20]. White-box watermarking [21–23] embeds watermark information within the model’s internal information, requiring access to and modification of the model’s internal parameters or structure to ensure the watermark’s high security and accuracy. In contrast, black-box watermarks [24, 25] embed watermark information in the model’s input and output behaviors. This approach does not require understanding or modifying the model’s internal structure, and watermark verification is performed by observing input and output pairs.

This work was supported in part by the NSFC under Grant 42201444 and Grant 62206238; and in part by the Natural Science Foundation of Jiangsu Province under Grant BK20220562. (*Corresponding authors: Weitong Chen (e-mail: wtchen@yzu.edu.cn); Di Wu (e-mail: di.wu@unisq.edu.au).*)

Weitong Chen and Wei Zhang are with the School of Information Engineering, Yangzhou University, Yangzhou 225009, China (e-mail: wtchen@yzu.edu.cn; mx120220560@stu.yzu.edu.cn).

Di Wu is with the School of Mathematics, Physics and Computing (SoMPC), University of Southern Queensland, Toowoomba, QLD 4350, Australia (e-mail: di.wu@unisq.edu.au).

Anja Keskinarkaus and Tapio Seppänen are with Physiological Signal Analysis Team, Center for Machine Vision and Signal Analysis, University of Oulu, Oulu 90014, Finland (e-mail: Anja.Keskinarkaus@oulu.fi; Tapio.Seppanen@oulu.fi).

Jiale Zhang is with the School of Information Engineering, Yangzhou University, Yangzhou 225009, China, also with the SoMPC, University of Southern Queensland, Toowoomba, QLD 4350, Australia (e-mail: jialezhang@yzu.edu.cn).

Longxiang Gao is with the Key Laboratory of Computing Power Network and Information Security, Ministry of Education, Shandong Computer Science Center, Qilu University of Technology (Shandong Academy of Sciences), Jinan 250353, China, also with the Shandong Provincial Key Laboratory of Computer Networks, Shandong Fundamental Research Center for Computer Science, Jinan 250014, China, and also with the SoMPC, University of Southern Queensland, Toowoomba, QLD 4350, Australia (e-mail: gaolx@sdas.org).

Tom H. Luan is with the School of Cyber Science and Engineering, Xi’an Jiaotong University, Xi’an 710049, China (e-mail: tom.luan@xjtu.edu.cn).

TABLE I: State-of-the-art Model Watermarking Methods in FL. ✓: Implement; × : Not implement.

Method	Verification	Watermarks embedding	Malicious Client Detection	Leakage Tracing	Copyright Verification
Liu et al [26]	Black-Box	Client	×	×	✓
Waffle [30]	Black-Box	Server	✓	×	×
Merkle-Sign [31]	White-Box/Black-Box	Server	×	✓	✓
WMDefence [5]	Black-Box	Server	✓	×	×
Yang et al [34]	Black-Box	Client	×	×	✓
Fedipr [27]	White-Box/Black-Box	Client	×	×	✓
Fedright [33]	White-Box	Server	×	✓	✓
FedCIP [28]	White-Box	Client	×	✓	✓
FedTracker [32]	White-Box/Black-Box	Server	×	✓	✓
PersistVerify [29]	Black-Box	Client	×	×	✓
<i>FedMLC</i>	White-Box	Server	✓	✓	✓

In FL, the protection mechanisms for model watermarking become more complex and critical because the FL architecture differs from centralized learning, where all participants act as clients to collaboratively train a model, increasing the risk of model attacks. To address this issue, current FL watermarking schemes are primarily categorized into client-side watermarking [26–29] and server-side watermarking [30–33]. Client-side watermarking involves generating and embedding the watermark on the client’s device, allowing for dynamic adjustments to the watermark content based on the client or device. In contrast, server-side watermarking involves generating and embedding the watermark on the server, ensuring that the content distributed to the client already contains the watermark.

Existing watermarking technologies are primarily used for copyright protection, including leakage tracking and copyright verification (Table.I). However, ineffective malicious client detection can lead to model poisoning, affecting model convergence. These three aspects are all crucial, as they collectively ensure the security and traceability of the model. Therefore, our research aims to propose a watermarking scheme that addresses these three critical issues simultaneously.

Our work: We propose a novel white-box watermarking method that embeds watermarks at different stages of the model training process using the same technique. The watermark information is embedded into the parameters of the normalization layer through the guidance of an introduced regularization term. The proposed Federated watermarking achieves **Malicious client detection**, **Leakage tracking**, and **Copyright verification (FedMLC)** simultaneously without significantly affecting model performance. By analyzing the degradation of watermarks in the uploaded model parameters, potential malicious clients can be identified and isolated, effectively preventing malicious behavior from compromising model security. Meanwhile, the embedded watermark can track the source of model leakage, hold malicious users accountable, and assert copyright. FedMLC ensures the security, stability, and traceability of the model in a FL environment, providing an effective technical solution to address FL security issues.

Our contributions to this paper are:

- 1) To commence, we propose the first FL model watermarking scheme that simultaneously enables malicious client

detection, leakage tracking, and copyright verification. We ensure the effectiveness and ease of watermarking operation by embedding unique watermarks in the same segment of parameters across all client models using a consistent embedding method.

- 2) Moreover, to ensure the model retains its efficiency and accuracy, our proposed method re-embeds the watermark into each client during every communication round to prevent the accumulation of watermark embeddings.
- 3) Finally, the method is designed with consideration for various potential attacks. Experimental results show that the watermark can still be effectively extracted under these attacks, demonstrating strong robustness and providing a reliable security mechanism to protect the model.

The rest of FedMLC is organized as follows. Section II reviews the existing technologies of DNN and FL watermarking. Section III introduces the problem statement. The proposed method is discussed in Section IV. Section V evaluates and analyses the experiment results. Finally, Section VI summarizes this work.

II. RELATED WORKS

A. DNN Watermarking Methods

White-Box Watermarking: White-Box Watermarking involves directly embedding watermarks into the parameters or structure of a model during training. Uchida et al. [22] first proposed the model watermarking scheme that utilized parameter regularizers to modify model parameters for watermark embedding. However, this method is vulnerable to cover attacks. Later, Rouhani et al. [35, 36] embedded watermark information into the probability density functions of different layers of the model, which can resist various deletion and transformation attacks, including the cover attacks. Additionally, Fan et al. [21, 37] embedded passports in the model structure, using forged passports can significantly degrade the model’s inference performance, effectively defending against ambiguity attacks. Furthermore, Liu et al. [38] proposed a new method to construct a more robust watermark using less information, reducing the damage caused by parameter changes and effectively resisting ambiguity attacks.

Black-Box Watermarking: Black-Box Watermarking involves embedding watermarks in a model’s output or behavior.

For instance, Adi et al. [17] leveraged neural networks' over-parameterization vulnerability, which makes models prone to backdoor attacks. They turned this susceptibility into an advantage for copyright verification by embedding watermarks using random colored images as triggers. Li et al. [19] enhanced FedMLC by strengthening the connection between the trigger set and labels. The generated watermark samples are almost indistinguishable from the original samples, thereby improving the undetectability and robustness of the watermark. Lounici et al. [24] proposed three trigger set generation techniques compatible with image and text data, broadening the application of model watermarking. Subsequently, Bansal et al. [18] introduced the first verifiable watermarking scheme to better resist deletion attacks. They demonstrated the irreversibility of watermarks within a certain threshold using random smoothing techniques. Additionally, Lao et al. [13] ensured that specific key samples output predetermined labels by adjusting minor parameters, enabling client identification and enhancing robustness against various transformation attacks.

B. FL Watermarking Methods

Client-side Watermarking: Client-side watermarking involves embedding specific identifiers or information directly on the client side. Liu et al. [26] introduced a method where predefined noise is added to the model. They also employed homomorphic encryption to secure model weights and gradients during data exchange, ensuring data privacy while providing reliable copyright verification. Building on Liu et al.'s [26] work, Yang et al. [34] designed an unambiguous trigger set construction mechanism based on permutation keys and noise patterns. This mechanism embeds the trigger set through gradient enhancement, guaranteeing its unforgeability. Li et al. [27] proposed the FedIpr framework, which allows each client to independently embed private watermarks to claim ownership and concluded from experiments that feature watermarks embedded in normalized scale parameters are more reliable. Liang et al. [28] embedded watermarks on the client side that are compatible with FL secure aggregation to enable traitor tracking. Nie et al. [29] integrated boundary sample selection and spatial attention mechanisms to enhance watermark robustness, tackling the challenge of undetected malicious clients continuously stealing models in FL while ensuring secure model ownership verification. As summarized in Table.I, Liu et al. [26], Yang et al. [34], Li et al. [27], and Nie et al. [29] achieved copyright verification, while Liang et al. [28] accomplished both copyright protection and leakage tracing simultaneously.

Server-side Watermarking: Server-side watermarking typically refers to embedding a watermark on the server side. Tekgul et al. [30] proposed the first server-side watermarking scheme, which involves a retraining process after server aggregation to embed the watermark into the global model. Similarly, Zheng et al. [5] introduced retraining on the server-side to embed watermarks for detecting Byzantine attacks from malicious clients. Li et al. [31] combined watermarking technique with cryptographic primitives for distributed storage, proposing a watermarking protocol to protect deep neural

networks in FL, which enhances applicability in real-world scenarios and achieves traceability. Compared to Li et al. [31], the scheme proposed by Shao et al. [32] not only achieves traceability but also provides effective copyright verification. It employed a dual protection mechanism with global watermarks and local fingerprints and designed a fingerprint generation method based on fingerprint similarity scoring. Additionally, Chen et al. [33] used key samples to generate adversarial examples as model watermarks, which are then used to extract model features and train a detector. Then, detection results from the detector are used to claim model ownership. As shown in Table.I, Li et al. [31], Shao et al. [32], and Chen et al. [33] all achieved copyright protection and leak tracing, while Tekgul et al. [30] and Zheng et al. [5] focused on malicious clients detection.

Protecting the ownership of FL models is undoubtedly critical, but detecting malicious clients during the training process and tracing leakers are equally essential. However, existing FL watermarking techniques cannot simultaneously achieve these three objectives. To address this challenge, FedMLC introduces an innovative server-side watermarking scheme that accomplishes all three functions concurrently. This approach not only ensures effective copyright verification but also enables timely detection of malicious client attacks while providing traceability in the event of model leakage, thereby enhancing the overall security and traceability.

III. PROBLEM STATEMENT

A. Threat Model

In the IoT scenario, FL is widely used in smart healthcare, autonomous driving, industrial intelligent manufacturing and other fields to improve model performance while protecting data privacy. For instance, in the FL of autonomous driving systems, various car companies or partners collaboratively train a high-performance autonomous driving model, with each participant contributing a substantial amount of local driving data. However, certain malicious clients (attacker A) may exploit the local training process to steal model parameters and sell them to competitors or the black market for illicit profits. At the same time, external attackers (attacker B) might acquire the trained model through network attacks or data theft and attempt to remove the watermark to bypass copyright protection, enabling unauthorized commercial use. Although both types of attackers are aware that the watermark is embedded in the model, they need to employ various attacks to eliminate it, as the specific watermarking method remains unknown to them. The above can be summarized into the following four points: a) Malicious client A can copy and steal model parameters at any epoch of the training process, and then redistribute or sell the model. b) External attacker B aims to obtain a high-performance DNN model for illegal commercial purposes. c) External attacker B has obtained the model illegally but does not know the model parameters or training samples. d) Both attackers are aware of the presence of the watermark in the model, but neither knows the specific watermarking method, and both attempt to remove the watermark from the model.

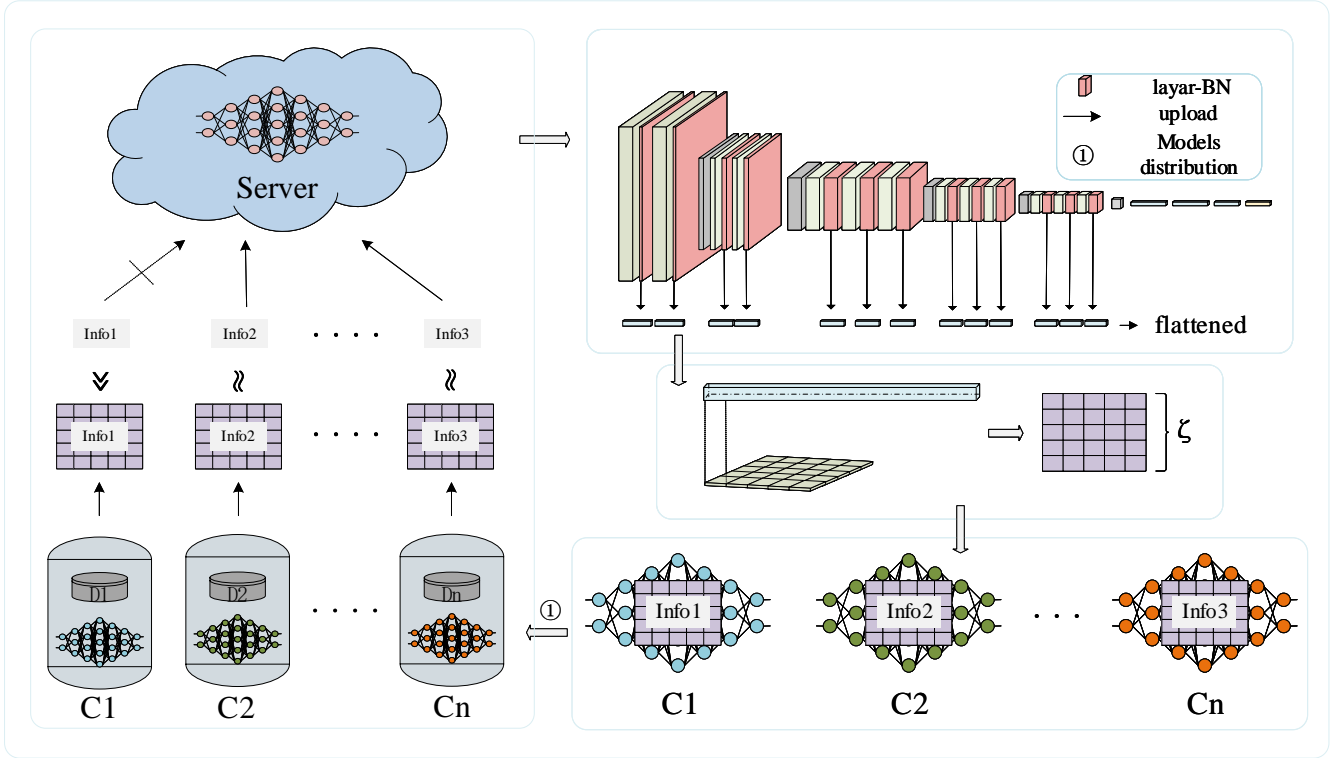


Fig. 1: Watermark embedding framework. ① First, generate global model copies for each client. ② Then, the watermark carrier is extracted from the models. ③ Next, the watermark is embedded and the watermarked model is distributed to each client. ④ After a local training round, clients upload their models to the server. ⑤ The server extracts the watermarks from the client-uploaded models and identifies malicious clients. During aggregation, client-uploaded models identified as malicious users do not participate. Repeat these steps in each communication round until the model training is completed.

B. Watermark Requirements

In our scenario, the FL server acts as a defender by embedding watermarks to achieve malicious client detection, leakage tracing, and copyright verification. The requirements commonly recognized for the watermark are as follows:

Fidelity: Ensuring that the model maintains high accuracy and reliability when performing its main task, unaffected by watermark embedding.

Effectiveness: The ability of watermarking technology to accurately identify the owner of a model or validate its legitimacy.

Robustness: The watermark embedded in the model should strongly resist various attacks while maintaining the model's performance.

Secrecy: The watermark must be imperceptible within the model and remain undetectable by unauthorized users.

Capacity: It determines how much information can be embedded in the model, and its size directly affects robustness and Effectiveness.

C. Definition of Malicious Client Detection, Leakage Tracing, and Copyright Verification

FedMLC aims to ensure the security of FL models through white-box watermarking techniques, focusing on malicious client detection, leakage tracing, and copyright verification. The specific definitions are detailed below:

Malicious client detection: Malicious client detection in FL aims to identify clients that may exhibit behaviors threatening model security or performance. In each communication aggregation round, the watermark decline rate in the client-uploaded models is calculated to determine if any clients are acting maliciously.

Leakage Tracing: The leaking client only leaks the model and does not perform other malicious operations. By extracting and verifying the watermark information in the model, the source can be traced, and the responsible party can be identified.

Copyright Verification: After the model is deployed, attackers might illegally copy and sell the model, claiming ownership. Copyright verification involves extracting the watermark information in the model to effectively identify the true owner, thereby safeguarding intellectual property rights.

IV. PROPOSED METHOD

A. Overall Design

FedMLC employs white-box watermarking to detect and trace malicious clients, as well as verify model copyrights. Specifically, in the server, a unique watermark is embedded in each client's model and then distributed. After each local training session, clients upload their model parameters to the server. The server then calculates the watermark decline

rate in the uploaded models to verify whether the models have been tampered with. After the final aggregation, the same watermarking technique is used to embed a copyright watermark in the global model to indicate ownership. The framework is illustrated in fig.1.

B. Watermark Generation

The proposed method involves embedding watermarks into the model after server aggregation but before distribution, allowing for tracking of the model's training status. The watermark information generation process utilizes the RSA encryption algorithm for digital signature, ensuring the uniqueness of the watermark information.

The server randomly generates a unique key k for each client. Then, the corresponding binary information b is calculated using the following Equation:

$$b = S(msg, k) \quad (1)$$

where msg represents the information to be embedded into the model, and S denotes the RSA algorithm.

To benefit the subsequent process, we use a mapping function to convert binary information b into sign factors represented by positive and negative symbols. Specifically, a bit "1" is mapped to the sign "+1", and a bit "0" is mapped to the sign "-1".

The client's watermark information mapping sequence key is obtained using the following equation:

$$key = sgn(b)_{i=0}^t \quad (2)$$

where sgn is the mapping function from 0, 1 to -1, +1, and t represents the bit length of the watermark. The copyright watermark information mapping sequence key_{cr} is obtained using the same way.

C. Watermark Embedding

The proposed watermarking scheme involves embedding each client's watermark key_i into their respective copies of the global model before the distribution by the server. We achieve watermark embedding for each client by introducing an additional regularization term into the original loss function. The global loss function is defined as Eq.3.

$$E(w) = \alpha E_0(w) + \beta E_R(w) \quad (3)$$

where w represents the global model parameters, $E_0(w)$ is the original loss function, $E_R(w)$ is the regularization term, and α and β are adjustable parameters. When training locally, $\alpha=1$, $\beta=0$, and when embedding watermarks, $\alpha=0$, $\beta=1$.

Embedding watermarks into normalization layers is more robust against removal attacks under the FL strategy compared to embedding them into other layers as demonstrated by Li et al. [27]. The proposed FedMLC method also embeds watermarks into parameters of normalization layers in order to effectively resist removal attacks. The specific procedure is as follows: First, the parameters from all normalization layers are extracted and flattened into a one-dimensional vector, denoted

Algorithm 1: watermark embedding

Input: Client number C and each client number C_i , client watermark information $\{key_i\}_{i=1}^C$, Clients datasets $\{D_i\}_{i=0}^C$, Total training rounds T and current training round e .

Output: Global model M

```

1 Initialize the global model  $M$ .
2 for  $0 < e < T$  do
3   Generate  $c$  copies of the global model  $\{M'_i\}_{i=0}^c$ 
4   for  $0 < i < C$  do
5     Extract BN layer parameters
6     Parameter flattening  $\rightarrow \phi^n$ 
7      $\varphi^m = (\phi^n * K)$ 
8      $\theta_i = \frac{1}{t'} \sum_{j \in \varphi_i} j$ 
9      $M'_i = \text{Insert}(M'_i, \theta, key_i)$ 
10    send  $M'_i$  to  $C_i$ 
11  end
12  for  $0 < i < C$  do
13     $M_i^{e+1} = \text{LocalTrain}(M'_i, D_i)$ 
14    send  $M_i^{e+1}$  to Server
15  end
16  Malicious Client Detection
17   $M_i^{e+1} = \leftarrow \text{FedAVG}(M_i^{e+1})$ 
18 end
19 Extract BN layer parameters
20 Parameter flattening  $\rightarrow \phi^n$ 
21  $\varphi^m = (\phi^n * K)$ 
22  $\theta = \frac{1}{t'} \sum_{j \in \varphi_i} j$ 
23  $M = \text{Insert}(M, \theta, key_{cr})$ 
24 return  $M$ 

```

as ϕ . Next, a convolution operation is performed on ϕ to obtain a new vector φ . The convolution operation in this context involves splitting the flattened parameter vector into multiple segments and multiplying each segment by a coefficient that approximates the length of that segment. The convolution operation is carried out as follows:

$$\varphi = (\phi * K) = \sum_{\varepsilon=-\infty}^{\infty} \phi(\varepsilon) K_d(\varepsilon), d \in [1, m] \quad (4)$$

$$K_d(\varepsilon) = \begin{cases} \frac{1}{\lfloor \frac{dn}{m} \rfloor - \lceil \frac{(d-1)n}{m} \rceil}, & \text{if } \lfloor \frac{(d-1)n}{m} \rfloor < \varepsilon < \lceil \frac{dn}{m} \rceil \\ 0, & \text{other} \end{cases} \quad (5)$$

where ε is the position of the target element in K , d is the position of the target element in φ^m , n and m represent the dimensions of ϕ and φ respectively, and K represents the convolution kernel.

Through the above convolution operation, we obtain a matrix φ with t rows and t' columns, where t represents the bit length of the key in Eq.2 and $t' = m/t$. We define a parameter rate λ , which is used to select the proportion of model parameters for the watermark carrier. For each row of φ , we first take the absolute value of the parameters, then select a portion of the parameters with smaller absolute values

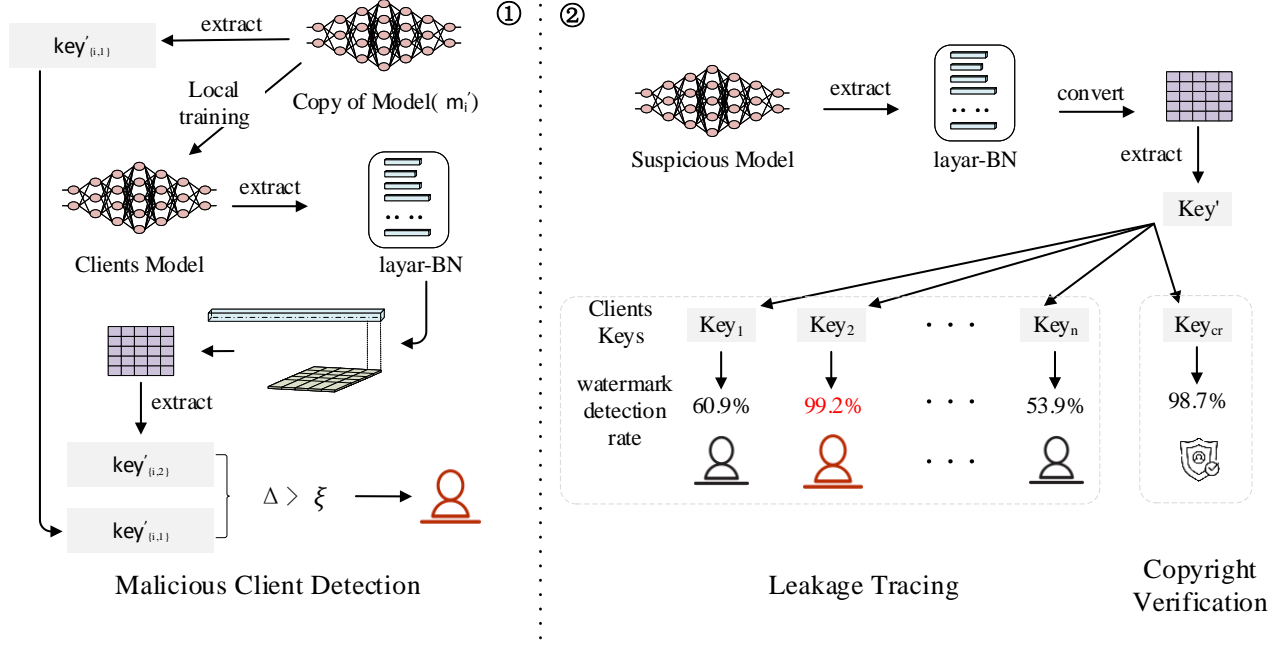


Fig. 2: Watermark verification framework. ① Malicious client detection: Extract the watermark information key from the client model and calculate the watermark decline rate. If the watermark decline rate exceeds the threshold ξ , the client is identified as a malicious client. ② Leakage tracing and copyright verification: Perform watermark extraction on the suspicious model and compare key' with the original watermark of each client to calculate the watermark detection rate. If the highest watermark detection rate exceeds the threshold η , the corresponding client will be identified as a leaking client. If the suspicious model is not leaked from the client, proceed with copyright verification. The verification method is the same as for leakage tracing. key' is compared with key_{cr} to calculate watermark detection rate, and if it exceeds η , the model is deemed to be infringing.

and set them to 0, this portion of parameters accounts for a proportion λ of each row. Next, we take the average of the parameters in each row to obtain a parameter set, denoted as θ , as shown in Eq.6.

$$\theta = \frac{1}{t'} \sum_{j \in \phi_i} j, \quad i \in [0, t) \quad (6)$$

where i represents the i -th row of ϕ , and j represents the j -th parameter of the i -th row.

The penalty term $E_R(w)$ introduced in Eq.3 serves as a regularization term, imposing additional constraints on the model parameters. Specifically, the role of this penalty term is to ensure that the signs of the model parameters θ are the same as the sign of the key . This enables the embedding of watermark information into the model without degrading its original performance. The definition of the regularization term is given in Eq.7.

$$E_R(w) = \sum_{i=0}^t ReLU(\mu - \theta * key_i) \quad (7)$$

where $\mu = 0.01$ by default.

After that, the server distributes a uniquely watermarked model m_i to each client C_i , ensuring that each client receives an exclusive copy of the global model. During the local training process, clients train the models using local datasets. Once the local training is completed, each client uploads the updated model m_i^c back to the server.

After the last round of aggregation, a watermark is added to the global model again, known as the model copyright watermark. The embedding process is the same as m_i^c embedding, with the distinction lying in the embedded information. Unlike the client watermark that represents a specific client, the copyright watermark signifies ownership of the entire model.

The watermark embedding process is summarized in Algorithm.1.

D. Watermark Verification

Malicious client detection: The detection process for malicious clients is conducted during training, as shown in ① of Fig.2.

Considering that the proposed method modulates the signs of model parameters by guiding model training to achieve watermark embedding, malicious client detection aims to measure the watermark decline rate. When a malicious client performs malicious operations on the model, the originally embedded watermark in the model undergoes significant changes. This tampering behavior may involve adjustments to the model's weights or biases, resulting in the destruction or degradation of the embedded watermark information, causing the watermark to decay beyond the predefined threshold. Therefore, before distributing the watermarked model to the client, we first extract the watermark $key'_{i,1}$ from it to record the success rate of watermark embedding.

Then, after local training, we extract watermark $key'_{i,2}$ from the client model. The method for extracting the watermark

is the same as the process of calculating θ . Finally, the watermark detection rates for $key'_{i,1}$ and $key'_{i,2}$ are calculated.

The watermark detection rate ψ refers to the accuracy of successfully extracting the pre-embedded watermark from the model and is defined as follows:

$$\psi_{key'} = 1 - \frac{1}{t}H(key, key') \quad (8)$$

where $H(key, key')$ measures Hamming distance between extracted watermark key' and the original watermarks key ;

Finally, a judgment is made based on the watermark decline rate $\Delta = \psi_{key'_{i,1}} - \psi_{key'_{i,2}}$. The client is marked as malicious if the decline rate exceeds a predefined threshold. The malicious client verification process is defined as:

$$V_{mclient} = \begin{cases} \Delta > \xi, & True \\ \text{other}, & False \end{cases} \quad (9)$$

Where ξ represents the threshold for judging whether the client is malicious.

The client-uploaded models verified to be potentially malicious are excluded from the current round of aggregation, ensuring the security and integrity of the global model.

Leakage Tracing: If a client illegally copies or leaks the model during local training, the model will inevitably contain watermark information related to that client. To detect the source of the leak, we follow the steps outlined in ② of Fig.2. The process of extracting key' is the same as the method used for detecting malicious clients (Eq.9). First, calculate the watermark detection rate of each client based on the extracted key' . Then, by comparing the watermark detection rate of all clients, identify the client with the highest detection rate $\max(\psi_{key'_i})$ that is also higher than the threshold η . This client is considered the source of the leak. If all extraction results are below η , the model is not considered stolen from us. This process can be expressed by Eq.10.

$$V_{track} = \begin{cases} \max(\psi_{key'_i}) > \eta, & True \\ \text{other}, & False \end{cases} \quad (10)$$

Where η defines the threshold for judging whether the watermark exists.

Copyright verification: Once a model is leaked, it becomes challenging to determine whether the leak originated from the client side or post-deployment. If the previous section's leakage tracing fails to detect the leaking client, it becomes necessary to verify whether the model was leaked after deployment. This process is also described in ② of Fig.2. The model copyright verification process is similar to the leakage tracing process, with a key difference: the watermark detection rate is calculated based on the Hamming distance between the key' and the copyright information key_{cr} . If $\psi_{key'_{cr}}$ exceeds the threshold η , the model is judged to be infringing. This can be defined as follows:

$$V_{copyright} = \begin{cases} \psi_{key'_{cr}} > \eta, & True \\ \text{other}, & False \end{cases} \quad (11)$$

V. EXPERIMENTAL EVALUATION

A. Experiment Settings

The study utilized the PyTorch 1.13.1 framework. The hardware configuration included an NVIDIA GeForce RTX 4060 Ti GPU with 8 GB of memory, an Intel Core i5-13600KF CPU. The experiment involved 50 participants. Based on independent and identically distributed (IID) samples, the dataset was divided into 50 simulated client datasets. The dataset was divided into training and test sets in a ratio of 8:2. Under the non-independent and identically distributed (Non-IID) setting, the dataset was divided into three different concentrations of 0.5, 0.7 and 0.9 according to the Dirichlet distribution generation, and verify the impact of different data set partition concentrations on the experimental results. The training process includes 150 rounds of communication. The learning rate of each round is set to 0.01, the momentum is set to 0.9, the minibatch size is set to 64, and the selected client is trained locally for 10 epochs. The watermark length t was set to 256 bits.

1) *Models:* In our experiments, we selected three classic models: AlexNet, VGG16, and ResNet-18. AlexNet introduced by Krizhevsky et al. [39] in 2012, consists of 8 layers, including 5 convolutional layers and 3 fully connected layers. It employs ReLU activation functions, local response normalization, and overlapping max pooling. VGG16 developed by the Oxford Visual Geometry Group [40], comprises 16 layers, including 13 convolutional layers with 3x3 receptive fields and 3 fully connected layers. It is known for its simplicity and depth, allowing it to capture fine-grained features efficiently. ResNet-18 part of the ResNet series introduced by He et al. [41], includes 18 layers with residual connections that address the vanishing gradient problem in deep networks, ensuring robustness and efficiency for various image classification tasks.

2) *Datasets:* In our experiments, we used three datasets: MNIST [42], CIFAR-10 [43], and CIFAR-100 [43].

MNIST: The MNIST (Modified National Institute of Standards and Technology) dataset is a handwritten digit classification dataset. It contains 60,000 grayscale images for training and 10,000 grayscale images for testing. Each image is 28*28 pixels in size. Each image contains a handwritten digit (from 0 to 9). The task of the MNIST dataset is to classify handwritten digit images into corresponding digital labels.

CIFAR-10: The goal of the CIFAR-10 (Canadian Institute For Advanced Research) dataset is to classify color images of 10 different categories. It contains 60,000 images with an image size of 32*32 pixels and is divided into 10 categories, each containing 6,000 images.

CIFAR-100: CIFAR-100 is an extended version of the CIFAR-10 dataset. The categories are more refined, so the classification task is more complex. There are 100 categories in total, and each category has only 600 images, totaling 60,000 images. The size of each image is still 32*32 pixels, and the number of images contained in each category is also the same.

These three datasets—MNIST, CIFAR-10, and CIFAR-100—serve different purposes in model evaluation. The MNIST dataset is suitable for validating basic image classification models. CIFAR-10 contains color images across 10 categories, while CIFAR-100 includes 100 categories, presenting a higher level of challenge. These three datasets are widely utilized as benchmark datasets for deep learning, enabling direct comparisons with previous studies and thereby ensuring the reproducibility and reliability of experimental results.

3) *Threshold: Setting of Threshold ξ* : To determine an appropriate threshold ξ in Eq.9, we conducted a statistical experiment. We measured the differences in the watermark detection rate ψ for 1,000 malicious clients and 1,000 benign clients after local training, as shown in Fig.3. The comparison of these differences clearly shows that the watermark decline for malicious clients is significantly greater than that for benign clients. The average watermark decline rate for malicious clients is 47%, whereas for benign clients, it ranges only from 0.1% to 8%. Based on these findings, we set ξ to 15% to effectively distinguish malicious clients from benign clients.

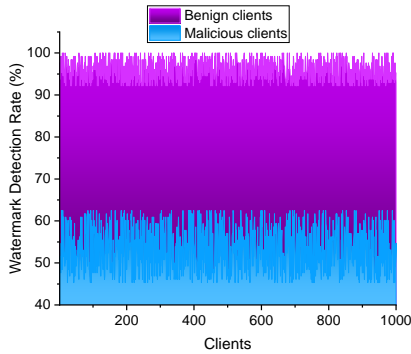


Fig. 3: Watermark degradation after local training by malicious and benign clients.

Setting of Threshold η : To determine an appropriate threshold η in Eq.10 and Eq.11, we compared the watermark detection rate ψ of 1,000 watermarked models and 1,000 non-watermarked models. The results show that ψ for watermarked models are mostly between 90% and 100%, while ψ for the non-watermarked models ranges only between 45% and 55%, as illustrated in fig.4. Based on these observations, we set $\eta = 85%$, which effectively distinguishes between watermarked and non-watermarked models.

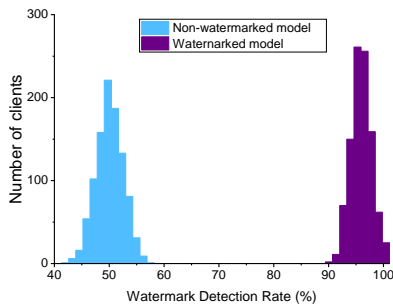


Fig. 4: Watermark extraction results by watermarked model and non-watermarked model.

B. Fidelity

To evaluate fidelity, we conducted experiments using various models on three different datasets, comparing the accuracy (ACC) of watermarked models and the non-watermarked models for the main task. We benchmarked these results against FedTracker [32], the state-of-the-art method of the same type. FedTracker embeds the watermarks in different segments, while we embed the watermarks within the same segment of parameters.

As shown in Fig.5, we compared the watermark embedding method used in FedTracker with our proposed FedMLC watermarking scheme. Although the watermark embedding of FedMLC affects model performance, the accuracy of the main task decreases by less than 1% compared to non-watermarked models. FedMLC consistently outperforms FedTracker across all four experimental groups. This indicates that FedMLC maintains high fidelity.

C. Effectiveness

1) *Effectiveness of Malicious Client Detection*: In this experiment, we set a total of 50 clients and simulated scenarios where malicious clients made up 20%, 40%, 60%, and 80% of the total. The results of watermark degradation under different proportions of malicious clients are shown in fig.6. The Δ of all models uploaded by malicious clients are greater than threshold ξ , indicating that our detection mechanism successfully identified all malicious clients, even when they constituted a substantial portion. This suggests that our method demonstrates robust detection capabilities across various proportions of malicious clients, confirming the effectiveness of our proposed malicious client detection mechanism.

2) *Effectiveness of Traceability*: We tested the impact of different number of clients on the effectiveness of leakage tracing. The experimental results are shown in Fig.7. The watermark detection for the leaker is highly significant, whether there are 30 clients or 50 clients. Specifically, for the model with 30 clients, the watermark detection rate of the leaker is 99.28%. For the model with 50 clients, the watermark detection rate reaches 98.4%. The watermark detection rate of the benign client watermarks in the leaker model is around 50%. This demonstrates that the proposed watermarking algorithm can effectively trace the leaker, thereby ensuring data security and traceability.

3) *Effectiveness of Copyright Verification*: In our study, we tested the effectiveness of copyright verification on models trained with 30 and 50 clients respectively. As shown in Table.II, the complexity of the cifar100 dataset, makes it difficult for the model to converge, which also affects the performance of the watermark embedding. Despite this, for all other datasets, the watermark detection rate exceeded 99% in each group of experiments. These findings demonstrate that our proposed Copyright Verification scheme is reliable and stable in most scenarios.

D. Robustness

1) *Robustness against Fine-tuning Attack*: To evaluate the robustness of our watermarking algorithm against fine-tuning

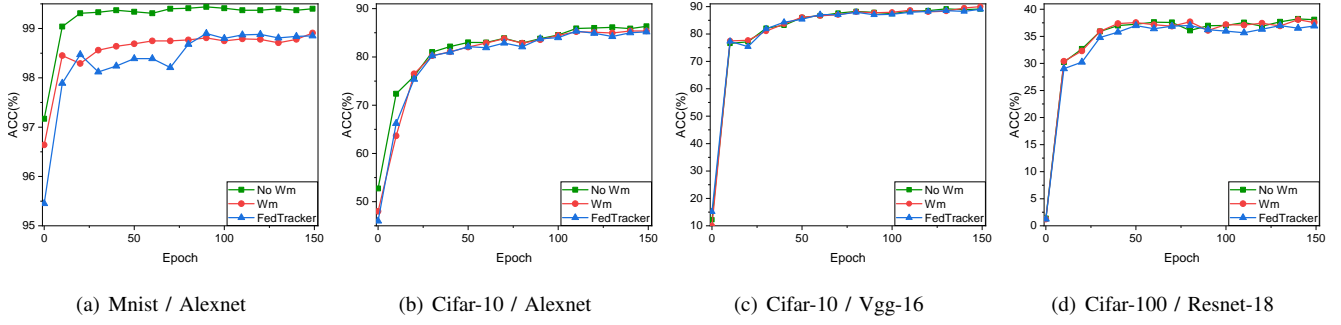


Fig. 5: Impact of watermark embedding on main task accuracy.

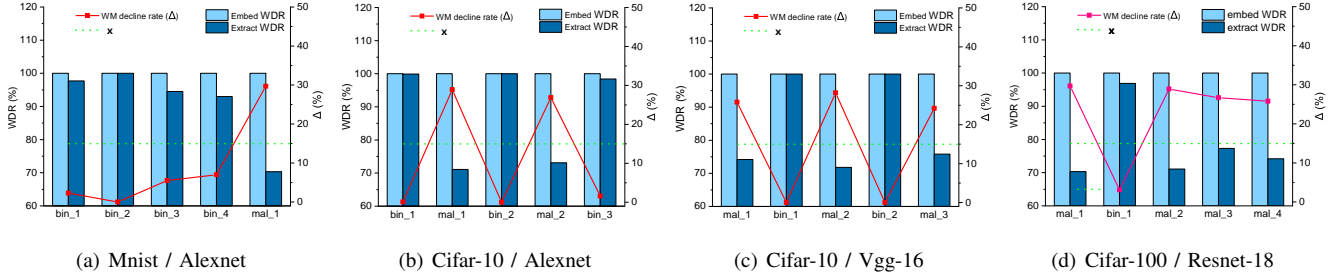


Fig. 6: Watermark degradation under different proportions of malicious clients.

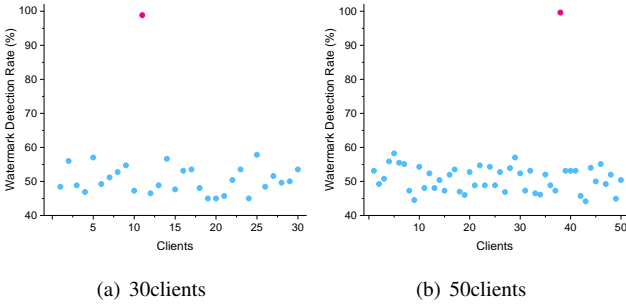


Fig. 7: Effectiveness of traceability under different numbers of clients. Red dots represent malicious users, while blue dots represent benign users.

TABLE II: Effectiveness of copyright verification under the different number of clients.

Dateset / Model	Watermark detection rate (%)	
	30 clients	50 clients
Mnist / Alexnet	100.00	100.00
Cifar10 / Alexnet	100.00	99.98
Cifar10 / Vgg-16	100.00	100.00
Cifar100 / Resnet18	90.32	89.03

attacks, we trained the model without the regularization term and then assessed the watermark detection rate after 50 training epochs.

As shown in Fig.8, the watermark detection rate of our proposed algorithm remains nearly unchanged after fine-tuning. In contrast, the copyright watermark detection rate of the FedTracker scheme shows a slight decrease under the same conditions, yet it is still lower than that of our scheme.

This indicates that our watermarking algorithm exhibits higher robustness against fine-tuning attacks.

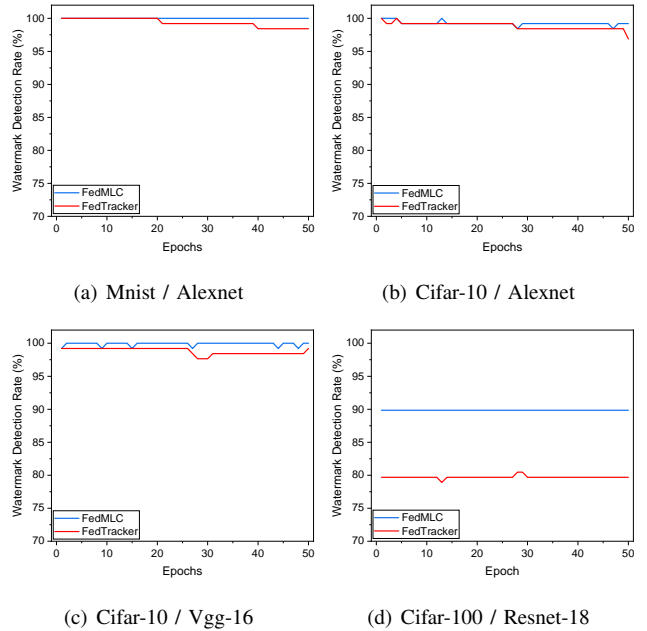


Fig. 8: Robustness against fine-tuning attacks

2) *Robustness against Pruning Attack*: Pruning attacks aim to reduce the model's complexity to undermine watermarks, making them difficult to detect. FedMLC uses a weight-based pruning method to evaluate the robustness of the watermark under such attacks.

As shown in Fig.9, after 50% pruning, both the watermark detection rate and main task accuracy of the watermark algorithm proposed in FedMLC and FedTracker algorithm have

decreased significantly. It is important to note that the watermark remains effective as long as the models' performance does not significantly decline. When the pruning strength is sufficient to remove the watermark, the model also loses its protective value. Furthermore, while maintaining effective performance on the model's main task, our watermark detection rate exhibits a lower decline compared to FedTracker. This suggests that the watermarking scheme proposed in FedMLC demonstrates greater robustness against pruning attacks.

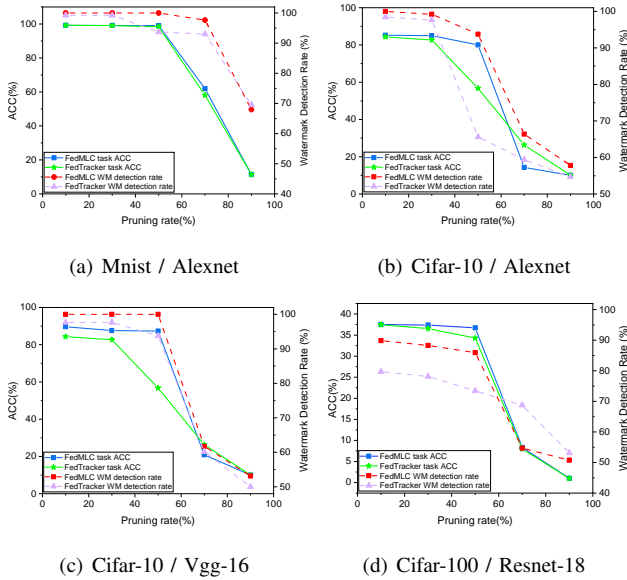


Fig. 9: Robustness against pruning attacks

3) *Robustness against Quantization Attack*: Model quantization involves converting the data types of weights and activation parameters within the model (e.g., from floating-point to lower-bit integer) to accelerate the inference speed. However, this process can lead to a decrease in model accuracy. To evaluate the robustness of our proposed method against such attacks, the parameters are quantized into the following three types: float32, float16, and int8.

According to Table.III, the experimental results indicate that under different quantization levels, the model's accuracy and watermark detection rate did not significantly decline, with the maximum impact being only 1%. This demonstrates that our method can effectively resist the interference of quantization attacks, ensuring robustness in practical applications.

TABLE III: Robustness against quantization attacks.

Dataset / Model	Metric	float32	float16	int8
Mnist / Alexnet	task ACC(%)	99.4	99.18	99.10
	ψ (%)	100.00	100.00	100.00
Cifar-10 / Alexnet	task ACC(%)	85.38	85.38	85.20
	ψ (%)	100.00	100.00	100.00
Cifar-10 / Vgg-16	task ACC(%)	89.99	89.90	89.01
	ψ (%)	100.00	100.00	100.00
Cifar-100 / Resnet-18	task ACC(%)	37.54	37.54	36.60
	ψ (%)	89.43	85.32	85.98

E. Secrecy

To evaluate the secrecy of FedMLC, we analyzed the parameter distributions of AlexNet and VGG16 models trained on the CIFAR-10 dataset. As shown in Fig.10, we compared the parameters of watermarked and non-watermarked models. The results show that both types of models exhibit no significant difference in their parameter distributions, nearly following the same distribution. This similarity makes it difficult for attackers to discern whether the model is watermarked or non-watermarked, thereby verifying the secrecy of our scheme.

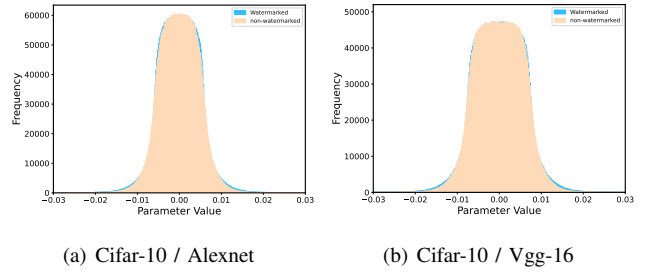


Fig. 10: Parameter distribution of watermarked and non-watermarked models.

F. Applicability under Non - Independent Identically Distribution

Evaluating the performance of the proposed method in Non - Independent Identically Distribution (non-iid) data environments is crucial, because non-iid data reflects the diversity and complexity of real-world data, and evaluating with non-iid data provides a more accurate understanding of the model's performance in varied and real-world scenarios. We employ the Dirichlet distribution to generate non-iid data. In FedMLC, the concentration parameter $\rho \in [0, 1]$ controls the degree of non-iidness in the data. When ρ is small, the generated data exhibit greater diversity, implying significant differences among data from different clients. Conversely, as ρ increases, the data tends to become more iid, making the data from different clients more similar. In our non-iid experiments, we tested with concentration parameters set to $\rho = 0.5, 0.7,$ and $0.9,$ respectively.

1) *Fidelity under Non-iid*: In non-iid scenarios, we evaluated the accuracy of models with different concentrations of Dirichlet distributions on the task set, as detailed in Table.IV. Compared to non-watermarked models, watermarked models showed a slight decrease in accuracy, with the maximum decline on the task set being 1.22%. These results indicate that even in scenarios with significant differences in data distribution and features, embedding watermarks did not significantly increase the risk of performance degradation of the models.

2) *Effectiveness under Non-iid: Effectiveness of Malicious Client Detection*: We evaluated the effectiveness of our malicious client detection mechanism under varying Dirichlet concentrations. Table.V provides examples showing that, regardless of the Dirichlet concentration, the watermark decline rate for malicious clients consistently exceeds the threshold, while the watermark decline rate for benign clients remains

TABLE IV: Fidelity under different non-iid settings

Dateset / Model	Method	Watermark detection rate (%)			
		$\rho = 0.5$	$\rho = 0.7$	$\rho = 0.9$	i.i.d
Mnist / Alexnet	WM / No WM	98.00 / 97.94	98.03 / 98.16	98.18 / 99.20	98.18 / 99.40
Cifar-10 / Alexnet	WM / No WM	84.88 / 85.78	84.92 / 85.87	85.04 / 86.21	85.38 / 86.33
Cifar-10 / Vgg-16	WM / No WM	88.98 / 89.13	88.98 / 89.70	89.10 / 89.99	89.12 / 89.99
Cifar-100 / Resnet-18	WM / No WM	37.12 / 37.30	37.16 / 37.69	37.22 / 38.00	37.54 / 38.07

below the threshold. This demonstrates that our malicious client detection mechanism can accurately distinguish between malicious and benign clients under different concentrations and that the non-iid setting does not affect its effectiveness.

TABLE V: Effectiveness of malicious client detection under non-iid settings.

Clinet	Watermark decline rate (%)			
	$\rho = 0.5$	$\rho = 0.7$	$\rho = 0.9$	iid
Bin-Client	1.6	1.6	1	1
Mal-Client	39	37.5	41.41	41.41
Bin-Client	3.12	1	0.6	0.5
Mal-Client	40.62	46.87	37.5	39.58
Mal-Client	41.28	41.60	38	37.27

Effectiveness of Traceability: To assess the impact of non-iid data on leakage tracing, we trained a VGG16 model on CIFAR-10 datasets using various Dirichlet concentrations and embedded watermarks on 50 clients. Compared to iid data, models with Dirichlet concentrations of 0.5, 0.7, and 0.9 could still accurately distinguish between malicious and benign clients. As shown in Fig.11, the watermark detection rate for malicious clients was significantly higher than for benign clients on these models. This demonstrates that our watermarking algorithm remains effective even when handling imbalanced data distributions.

Effectiveness of Copyright Verification: To assess the effectiveness of copyright verification under non-iid data distributions, we conducted an experiment using the CIFAR-10 dataset with varying Dirichlet concentration parameters. The experimental results, as shown in Table.VI, illustrate the impact of differing client data distribution on copyright verification. Notably, we observed that with a Dirichlet concentration parameter $\rho=0.5$, the watermark detection rate decreased by up to 3.1%, but copyright verification can still be achieved despite this reduction. Consequently, we conclude that copyright verification methods remain effective under non-iid conditions.

TABLE VI: Effectiveness of copyright verification under non-iid settings.

Dataset / Model	Watermark detection rate (%)			
	$\rho = 0.5$	$\rho = 0.7$	$\rho = 0.9$	iid
Cifar-10 / Vgg-16	98.40	98.40	99.22	100.00
Cifar-10 / Alexnet	96.88	99.22	100.00	100.00

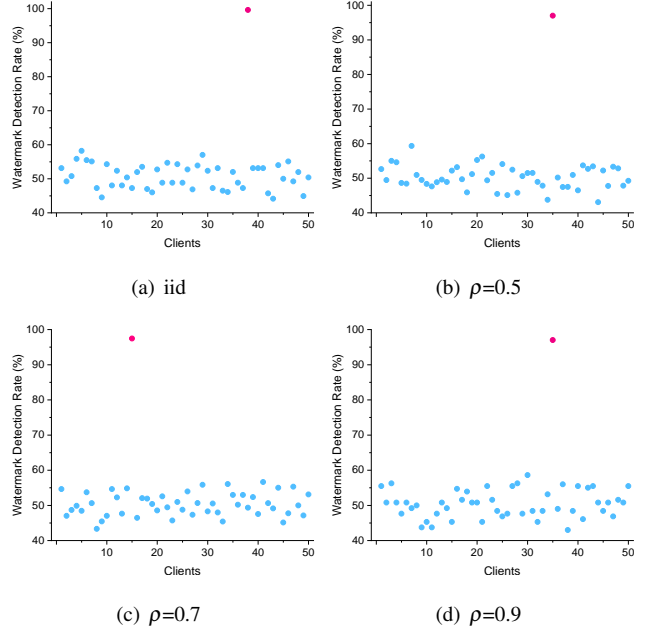


Fig. 11: Effectiveness of traceability under non-iid settings. Red dots represent malicious users, while blue dots represent benign users.

3) Robustness under Non-iid: Robustness Against Fine-tuning Attack: Regarding the robustness of our watermarking scheme against fine-tuning attacks on non-iid data, we analyzed the watermark detection rate after model fine-tuning under various Dirichlet concentration levels. According to the findings from Fig.12, indicate that even with a Dirichlet concentration of 0.5, we successfully extracted 96% of watermark bits from the model and achieved effective watermark verification after 50 epochs of fine-tuning. These results indicate that non-iid data does not significantly affect the robustness of our proposed watermarking scheme against fine-tuning attacks.

Robustness Against Pruning Attack: In our experiments, we evaluated the performance of models trained in various non-iid environments under pruning attacks. Regardless of the Dirichlet concentration being 0.5, 0.7, or 0.9, the models experienced a significant performance drop after pruning 50%, similar to what was observed in experiments under the iid setting. The performance degradation trend under pruning attacks was consistent with that of models trained in an iid environment, showing no additional decline due to the non-iid setting. Additionally, the watermark remained effective as long as the model performance was effective. This indicates that, across different Dirichlet concentrations, non-iid data does not

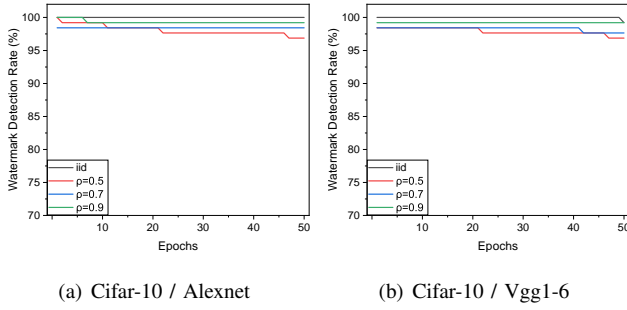


Fig. 12: Robustness against fine-tuning attacks under non-iid settings.

significantly impact the robustness against pruning attacks.

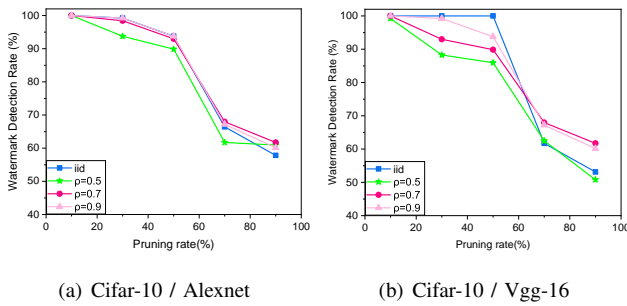


Fig. 13: Robustness against pruning attacks under non-iid settings.

Robustness Against Quantization Attack: In our experiments, we evaluated the performance of models trained on the CIFAR-10 dataset under different Dirichlet concentrations when subjected to quantization attacks. As shown in Table VII, the results indicate that varying Dirichlet concentrations have minimal impact on model performance. Although there was a slight performance decrease when using int8 precision with a Dirichlet concentration of 0.5, this decline does not significantly affect watermark verification accuracy. These findings suggest that models trained under non-iid conditions exhibit relatively stable robustness against quantization attacks.

TABLE VII: Robustness against quantization attacks under non-iid settings.

Data type	Watermark detection rate (%)			
	$\rho = 0.5$	$\rho = 0.7$	$\rho = 0.9$	iid
float32	100.00	100.00	100.00	100.00
float16	96.87	99.22	99.22	100.00
int8	93.75	98.40	99.22	100.00

VI. CONCLUSION

In this study, we propose FedMLC, a novel white-box watermarking method for protecting the copyright of FL models and detecting malicious users in IoT environments. FedMLC embeds a unique watermark for each edge node through watermarking technology, effectively tracing the source of leakage when edge nodes maliciously expose the model,

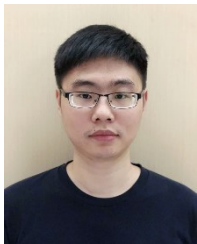
thereby enhancing the security and copyright protection of the model. Furthermore, the embedded watermark can also be used to determine whether malicious behavior occurs at edge nodes during the FL process. Before model deployment, we embed a copyright watermark into the global model with the same algorithm to ensure ownership rights. Experiments have demonstrated that FedMLC remains effective against various potential attacks, such as model pruning, fine-tuning, and quantization attacks. FedMLC consistently retains its effectiveness and reliability under both IID and non-IID conditions.

REFERENCES

- [1] D. C. Nguyen, M. Ding, Q.-V. Pham, P. N. Pathirana, L. B. Le, A. Seneviratne, J. Li, D. Niyato, and H. V. Poor, "Federated learning meets blockchain in edge computing: Opportunities and challenges," *IEEE Internet of Things Journal*, vol. 8, no. 16, pp. 12 806–12 825, 2021.
- [2] Y. Bai, L. Chen, J. Li, J. Wu, P. Zhou, Z. Xu, and J. Xu, "Multicore federated learning for mobile-edge computing platforms," *IEEE Internet of Things Journal*, vol. 10, no. 7, pp. 5940–5952, 2023.
- [3] M. Lansari, R. Bellafqira, K. Kapusta, V. Thouvenot, O. Bettan, and G. Coatrieux, "When federated learning meets watermarking: A comprehensive overview of techniques for intellectual property protection," *Machine Learning and Knowledge Extraction*, vol. 5, no. 4, pp. 1382–1406, 2023.
- [4] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *PMLR Artificial intelligence and statistics*, 2017, pp. 1273–1282.
- [5] X. Zheng, Q. Dong, and A. Fu, "Wmdefense: Using watermark to defense byzantine attacks in federated learning," in *IEEE INFOCOM 2022-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2022, pp. 1–6.
- [6] M. Fang, X. Cao, J. Jia, and N. Gong, "Local model poisoning attacks to {Byzantine-Robust} federated learning," in *29th USENIX security symposium (USENIX Security 20)*, 2020, pp. 1605–1622.
- [7] R. S. Antunes, C. André da Costa, A. Küderle, I. A. Yari, and B. Eskofier, "Federated learning for healthcare: Systematic review and architecture proposal," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 13, no. 4, pp. 1–23, 2022.
- [8] D. Li, J. Wang, L. Kong, S. Si, Z. Huang, C. Huang, and J. Xiao, "A nearest neighbor under-sampling strategy for vertical federated learning in financial domain," in *Proceedings of the 2022 ACM Workshop on Information Hiding and Multimedia Security*, 2022, pp. 123–128.
- [9] C. Yang, Q. Wang, M. Xu, Z. Chen, K. Bian, Y. Liu, and X. Liu, "Characterizing impacts of heterogeneity in federated learning upon large-scale smartphone data," in *Proceedings of the Web Conference 2021*, 2021, pp. 935–946.
- [10] Q. Liu, S. Yang, J. Liu, L. Zhao, P. Xiong, and J. Shen, "An efficient video watermark method using blockchain," *Knowledge-Based Systems*, vol. 259, p. 110066, 2023.

- [11] P. Lv, P. Li, S. Zhang, K. Chen, R. Liang, H. Ma, Y. Zhao, and Y. Li, "A robustness-assured white-box watermark in neural networks," *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 6, pp. 5214–5229, 2023.
- [12] Z. Ren, H. Fang, J. Zhang, Z. Ma, R. Lin, W. Zhang, and N. Yu, "A robust database watermarking scheme that preserves statistical characteristics," *IEEE Transactions on Knowledge and Data Engineering*, 2023.
- [13] Y. Lao, P. Yang, W. Zhao, and P. Li, "Identification for deep neural network: Simply adjusting few weights!" in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, 2022, pp. 1328–1341.
- [14] Y. Yan, X. Pan, M. Zhang, and M. Yang, "Rethinking {White-Box} watermarks on deep learning models under neural structural obfuscation," in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023, pp. 2347–2364.
- [15] T. Qiao, Y. Ma, N. Zheng, H. Wu, Y. Chen, M. Xu, and X. Luo, "A novel model watermarking for protecting generative adversarial network," *Computers & Security*, vol. 127, p. 103102, 2023.
- [16] G. Hua and A. B. J. Teoh, "Deep fidelity in dnn watermarking: A study of backdoor watermarking for classification models," *Pattern Recognition*, vol. 144, p. 109844, 2023.
- [17] Y. Adi, C. Baum, M. Cisse, B. Pinkas, and J. Keshet, "Turning your weakness into a strength: Watermarking deep neural networks by backdooring," in *27th USENIX security symposium (USENIX Security 18)*, 2018, pp. 1615–1631.
- [18] A. Bansal, P.-y. Chiang, M. J. Curry, R. Jain, C. Wigginton, V. Manjunatha, J. P. Dickerson, and T. Goldstein, "Certified neural network watermarks with randomized smoothing," in *International Conference on Machine Learning*, 2022, pp. 1450–1465.
- [19] Z. Li, C. Hu, Y. Zhang, and S. Guo, "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*, 2019, pp. 126–137.
- [20] Z. Yin, H. Yin, and X. Zhang, "Neural network fragile watermarking with no model performance degradation," in *2022 IEEE International Conference on Image Processing (ICIP)*, 2022, pp. 3958–3962.
- [21] L. Fan, K. W. Ng, C. S. Chan, and Q. Yang, "Deepipr: Deep neural network ownership verification with passports," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 10, pp. 6122–6139, 2021.
- [22] Y. Uchida, Y. Nagai, S. Sakazawa, and S. Satoh, "Embedding watermarks into deep neural networks," in *Proceedings of the 2017 ACM on international conference on multimedia retrieval*, 2017, pp. 269–277.
- [23] H. Chen, B. D. Rohani, and F. Koushanfar, "Deepmarks: A digital fingerprinting framework for deep neural networks," *arXiv preprint arXiv:1804.03648*, 2018.
- [24] S. Lounici, M. Njeh, O. Ermis, M. Önen, and S. Trabelsi, "Yes we can: watermarking machine learning models beyond classification," in *2021 IEEE 34th Computer Security Foundations Symposium (CSF)*, 2021, pp. 1–14.
- [25] S. Lounici, M. Önen, O. Ermis, and S. Trabelsi, "Blindspot: Watermarking through fairness," in *Proceedings of the 2022 ACM Workshop on Information Hiding and Multimedia Security*, 2022, pp. 39–50.
- [26] X. Liu, S. Shao, Y. Yang, K. Wu, W. Yang, and H. Fang, "Secure federated learning model verification: A client-side backdoor triggered watermarking scheme," in *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2021, pp. 2414–2419.
- [27] B. Li, L. Fan, H. Gu, J. Li, and Q. Yang, "Fedipr: Ownership verification for federated deep neural network models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 4, pp. 4521–4536, 2022.
- [28] J. Liang and R. Wang, "Fedcip: Federated client intellectual property protection with traitor tracking," *arXiv preprint arXiv:2306.01356*, 2023.
- [29] H. Nie and S. Lu, "Persistverify: Federated model ownership verification with spatial attention and boundary sampling," *Knowledge-Based Systems*, vol. 293, p. 111675, 2024.
- [30] B. G. Tekgul, Y. Xia, S. Marchal, and N. Asokan, "Waffle: Watermarking in federated learning," in *2021 40th International Symposium on Reliable Distributed Systems (SRDS)*, 2021, pp. 310–320.
- [31] F.-Q. Li, S.-L. Wang, and A. W.-C. Liew, "Towards practical watermark for deep neural networks in federated learning," *arXiv preprint arXiv:2105.03167*, 2021.
- [32] S. Shao, W. Yang, H. Gu, Z. Qin, L. Fan, and Q. Yang, "Fedtracker: Furnishing ownership verification and traceability for federated learning model," *IEEE Transactions on Dependable and Secure Computing*, 2024.
- [33] J. Chen, M. Li, Y. Cheng, and H. Zheng, "Fedright: An effective model copyright protection for federated learning," *Computers & Security*, vol. 135, p. 103504, 2023.
- [34] W. Yang, S. Shao, Y. Yang, X. Liu, X. Liu, Z. Xia, G. Schaefer, and H. Fang, "Watermarking in secure federated learning: A verification framework based on client-side backdooring," *ACM Transactions on Intelligent Systems and Technology*, vol. 15, no. 1, pp. 1–25, 2023.
- [35] B. D. Rouhani, H. Chen, and F. Koushanfar, "Deepsigns: A generic watermarking framework for ip protection of deep learning models," *arXiv preprint arXiv:1804.00750*, 2018.
- [36] B. Darvish Rouhani, H. Chen, and F. Koushanfar, "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*, 2019, pp. 485–497.
- [37] L. Fan, K. W. Ng, and C. S. Chan, "Rethinking deep neural network ownership verification: Embedding passports to defeat ambiguity attacks," vol. 32, 2019.
- [38] H. Liu, Z. Weng, and Y. Zhu, "Watermarking deep neural networks with greedy residuals," in *ICML*, 2021, pp. 6978–6988.

- [39] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Advances in neural information processing systems*, vol. 25, 2012.
- [40] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.
- [41] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” pp. 770–778, 2016.
- [42] C. J. B. Yann LeCun, Corinna Cortes, “Mnist handwritten digit database,” *ATT Labs*, 2010.
- [43] A. Krizhevsky, G. Hinton *et al.*, “Learning multiple layers of features from tiny images,” 2009.



WeiTong Chen (Member, IEEE) received the B.S. degree in geography, the M.S. degree in marine geography, and the Ph.D. degree in cartography and geography information system from Nanjing Normal University, Nanjing, China, in 2014, 2017, and 2021, respectively. He is currently a Lecturer with the School of Information Engineering, Yangzhou University. His research interests include information security and edge computing.

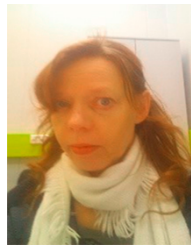


Wei Zhang received the B.E. degree from the School of Economics and Technology, Anhui Agricultural University, China, in 2022. She is currently working toward a M.S. degree in software engineering at Yangzhou University, Yangzhou, China. Her research interests include information security and federated learning.

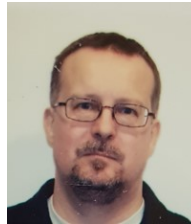


Di Wu (Senior Member, IEEE) received his B.Sc. degree in Information Technology (Honours) as a first class honours student and M.Sc. degree in Information Technology (Professional) from Deakin University, Australia in 2013 and 2012, respectively. He received Ph.D. degree from the School of Computer Science & Australian Artificial Intelligence Institute(AAII), University Technology Sydney, Sydney, Australia in 2020. He is a Senior Lecturer at the School of Mathematics, Physics, and Computing, the University of Southern Queensland (UniSQ). Prior

to that, he was a Lecturer at the same School. Prior to that, he was a Researcher Fellow at the Australian Institute for Machine Learning (AIML) and School of Computer Science, University of Adelaide, Adelaide, Australia. Previous to this, he was an Associate Research Fellow, Artificial Intelligence at Deakin Blockchain Innovation Lab, School of Information Technology, Deakin University, Melbourne, Australia, and worked as a Postdoc Fellow at the School of Computer Science, University of Technology Sydney (UTS), Sydney, Australia. He has more than 10 years of experience in research & development and academia. He has substantial industry experience in large project management, software development, and large system maintenance experience while working on various projects at China Telecom (Global 500), Shanghai. His research area focuses on applying federated learning, AI security and privacy, and trustworthy AI. He has published papers in high-quality refereed books, conferences, and journals, including top-tier venues such as ICLR, IJCAI, TKDE etc. He also serves as an associate editor in NLPJ and a reviewer for many high-quality academic conferences and journals, such as ICLR, SIGKDD, ACM MM, CoRL, TMC, TNNLS, TETCI, PR, etc.



Anja Keskinarkaus works as a postdoctoral researcher in the Center for Machine Vision and Signal Analysis in the University of Oulu. She received her M.Sc degree in information engineering in 1998 from the University of Oulu, Finland and the Dr. Tech. degree from the University of Oulu, Finland, in 2012. Her research interests include digital watermarking, multimedia signal processing and management, intelligent transport systems, biobanks, signal processing for biomedical signals and biomedical engineering.



Tapio Seppänen is Professor of biomedical engineering in the Center for Machine Vision and Signal Analysis in the Faculty of Information Technology and Electrical Engineering, University of Oulu, Finland. He has conducted research on, e.g., multi-channel ECG signal processing, heart-rate variability, autonomic nervous system signal processing, respiration signal processing, EEG signal processing, affective computing, speech signal processing, digital watermarking, and multimedia search engines.



Jiale Zhang (Member, IEEE) received the Ph.D. degree in computer science and technology the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2021. He is currently an Associate Professor with the School of Information Engineering, Yangzhou University, Yangzhou, China. He has published over 40 research papers in refereed international journals and conferences, such as IEEE TIFS, IEEE TDSC, IEEE TSC, and IJCAI. His research interests are mainly federated learning, AI

security, and blockchain security.



Longxiang Gao (Senior Member, IEEE) received the Ph.D. in computer science from Deakin University, Melbourne, VIC, Australia. He is currently a Professor with Shandong Computer Science Center, Qilu University of Technology (Shandong Academy of Sciences), Jinan, China. He is also an Adjunct Professor with the University of Southern Queensland, Toowoomba, QLD, Australia. He was a Senior Lecturer with the School of Information Technology, Deakin University and a Postdoctoral Research Fellow with IBM Research and Development, Mel-

bourne. He has over 130 publications, including patent, monograph, book chapter, journal and conference papers. Some of his publications have been published in the top venue, such as IEEE TRANSACTIONS ON MOBILE COMPUTING, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE INTERNET OF THINGS JOURNAL, IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE TRANSACTIONS ON COMPUTATIONAL SOCIAL SYSTEMS, IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, IEEE TRANSACTIONS ON NETWORK SCIENCE AND ENGINEERING, IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, and ACM Computing Surveys. He has been chief investigator for more than 20 research projects (the total awarded amount is over \$5 million), from pure research project to contracted industry research. His research interests include fog/edge computing, blockchain, data analysis, and privacy protection.



Tom H. Luan (Senior Member, IEEE) received the B.E. degree in electrical and computer engineering from Xi'an Jiaotong University, Xi'an, Shaanxi, China, in 2004, the M.Phil. degree in electrical and computer engineering from The Hong Kong University of Science and Technology, Hong Kong, in 2007, and the Ph.D. degree in electrical and computer engineering from the University of Waterloo, Waterloo, ON, Canada, in 2012. He is with the School of Cyber Science and Engineering, Xi'an Jiaotong University. He has authored or coauthored

more than 150 peer-reviewed papers in journal and conferences. His research interests include content distribution and media streaming in vehicular ad hoc networks, peer-to-peer networking, protocol design, performance evaluation of digital network, and edge computing. Dr. Luan was the recipient of the 2017 IEEE VTS Best Land Transportation Best Paper Award and the IEEE ICCS 2018 Best Paper Award.