Who leaked the model? Tracking IP Infringers in Accountable Federated Learning

Shuyang Yu[1], Junyuan Hong[1,2], Yi Zeng[3], Fei Wang[4], Ruoxi Jia[3] and Jiayu Zhou[1]
[1]Department of Computer Science and Engineering, Michigan State University
[2]Department of Electrical and Computer Engineering, University of Texas at Austin
[3]Department of Computer Engineering, Virginia Tech
[4]Department of Population Health Sciences, Cornell University
{yushuyan,hongju12, jiayuz}@msu.edu, {yizeng,ruoxijia}@vt.edu,
few2001@med.cornell.edu

Abstract

Federated learning (FL) emerges as an effective collaborative learning framework to coordinate data and computation resources from massive and distributed clients in training. Such collaboration results in non-trivial intellectual property (IP) represented by the model parameters that should be protected and shared by the whole party rather than an individual user. Meanwhile, the distributed nature of FL endorses a malicious client the convenience to compromise IP through illegal model leakage to unauthorized third parties. To block such IP leakage, it is essential to make the IP identifiable in the shared model and locate the anonymous infringer who first leaks it. The collective challenges call for *accountable federated learning*, which requires verifiable ownership of the model and is capable of revealing the infringer's identity upon leakage. In this paper, we propose Decodable Unique Watermarking (DUW) for complying with the requirements of accountable FL. Specifically, before a global model is sent to a client in an FL round, DUW encodes a client-unique key into the model by leveraging a backdoor-based watermark injection. To identify the infringer of a leaked model, DUW examines the model and checks if the triggers can be decoded as the corresponding keys. Extensive empirical results show that DUW is highly effective and robust, achieving over 99% watermark success rate for Digits, CIFAR-10, and CIFAR-100 datasets under heterogeneous FL settings, and identifying the IP infringer with 100% accuracy even after common watermark removal attempts.

1 Introduction

Federated learning (FL) [15] has been widely explored as a distributed learning paradigm to enable remote clients to collaboratively learn a central model without sharing their raw data, effectively leveraging the massive and diverse data available in clients for learning and protecting the data confidentiality. The learning process of FL models typically requires the coordination of significant computing resources from a multitude of clients to curate the valuable information in the client's data, and the FL models usually have improved performance than isolated learning and thus high commercial value. Recently, the risk of leaking such high-value models has drawn the attention of the public. One notable example is the leakage of the foundation model from Meta [1] by users who gained the restricted distribution of models. The leakage through restricted distribution could be even more severe in FL which allows all participating clients to gain access to the valued model. For each iterative communication round, a central server consolidates models from various client devices, forming a global or central model. This model is then disseminated back to the clients for the next

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

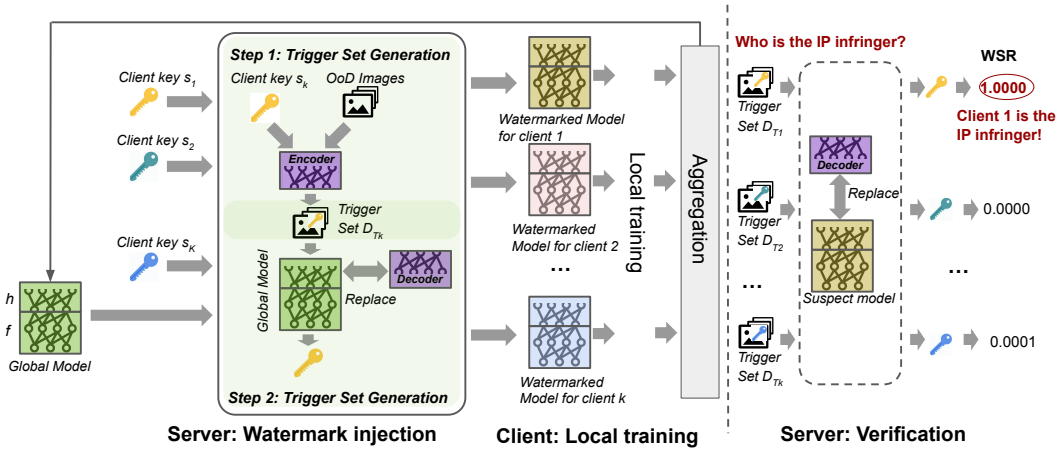


Figure 1: The proposed Decodable Unique Watermarking (DUW) for watermark injection and verification. During watermark injection, the server first uses a unique key assigned to each client and an OoD dataset as the input for the pre-trained encoder to generate trigger sets. When the server implant the watermark based on the objective function $J'(\theta_k)$ (Eq. (7)), a decoder is utilized to replace the classifier head in the FL model. During verification, the suspect model is tested on all the verification datasets of all the clients, and the client that leaked the model is identified by the one that achieves the highest WSR (Eq. (4)) in verification datasets.

update, and therefore the malicious clients have full access to the global models. As such, effectively protecting the global models in FL is a grand challenge.

Watermarking techniques [2, 4, 7, 8, 34, 36] are recently introduced to verify the IP ownership of models. Among them, backdoor-based watermarking shows strong applicability because of its model-agnostic nature, which repurposes the backdoor attacks of deep models and uses special-purposed data (trigger set) to insert hidden patterns in the model to produce undesired outputs given inputs with triggers [36, 19, 11, 21]. A typical backdoor-based watermarking operates as follows: The model owner first generates a trigger set consisting of samples paired with pre-defined target labels. The owner then embeds the watermark into the model by fine-tuning the model with the trigger set and the original training samples. To establish the ownership of the model, one evaluates the accuracy of the suspect model using the trigger set. The mechanism safeguards the assumption that only the watermarked model would perform exceptionally well on the unique trigger set. If the model's accuracy on the trigger set surpasses a significant threshold, the model likely belongs to the owner.

Conventional backdoor-based watermarking, however, does not apply to FL settings because of the required access to the training data to maintain model utility. To address the challenge, Tekgul *et al.* [33] proposed WAFFLE, which utilized only random noise and class-consistent patterns to embed a backdoor-based watermark into the FL model. However, since WAFFLE injected a unified watermark for all the clients, it cannot solve another critical question: *Who is the IP infringer among the FL clients?* Based on WAFFLE, Shao *et al.* [30] introduced a two-step method FedTracker to verify the ownership of the model with the central watermark from WAFFLE, and track the malicious clients in FL by embedding unique local fingerprints into local models. However, the local fingerprint in [30] is a parameter-based method, which is not applicable for many practical scenarios, where many re-sale models are reluctant to expose their parameters, and the two-step verification is redundant. Therefore, how to spend the least effort on changing the model while verifying and tracking the IP infringers using the same watermark in FL remains to be a challenging and open problem.

The aforementioned challenges call for a holistic solution towards *accountable federated learning*, which is characterized by the following essential requirements: R1) Accurate IP tracking: Each client has a unique ID to trace back. IP tracking should be confident to identify one and only one client. R2) Confident verification: The ownership verification should be confident. R3) Model utility: The watermark injected should have minimal impact on standard FL accuracy. R4) Robustness: The watermark should be robust and resilient against various watermark removal attacks. In this paper, we propose a practical watermarking framework for FL called Decodable Unique Watermarking (DUW) to comply with these requirements. Specifically, we first generate unique trigger sets for each client by using a pre-trained encoder [24] to embed client-wise unique keys to one randomly chosen out-of-distribution (OoD) dataset. During each communication round, the server watermarks the

aggregated global model using the client-wise trigger sets before dispatching the model. A decoder replaces the classifier head in the FL model during injection so that we can decode the model output to the client-wise keys. We propose a regularized watermark injection optimization process to preserve the model's utility. During verification, the suspect model is tested on the trigger sets of all the clients, and the client that achieves the highest watermark success rate (WSR) is considered to be the IP infringer. The framework of method is shown in Fig. 1.

The contributions of our work can be summarized in four folds:

- We make the FL model leakage from anonymity to accountability by injecting DUW. DUW enables ownership verification and leakage tracing at the same time without access to model parameters during verification.
- DUW is model-agnostic and can be incorporated into a wide spectrum of DNN types.
- With utility preserved, both the ownership verification and IP tracking of our DUW are not only accurate but also confident without collisions.
- Our DUW is robust against existing watermarking removal attacks, including fine-tuning, pruning, model extraction, and parameter perturbation.

2 Related Work and Background

Federated learning. Federated learning (FL) is a distributed learning framework that enables massive and remote clients to collaboratively train a high-quality central model [16]. FedAvg [26] is one of the representative methods for FL, which averages local models during aggregation. This work is based on the FedAvg setting. Suppose we have K clients, and our FL model M used for standard training consists of two components, including a feature extractor $f : \mathcal{X} \to \mathcal{Z}$ governed by θ^f, and a classifier $h : \mathcal{Z} \to \mathcal{Y}$ governed by θ^h, where $\mathcal{Z}$ is the latent feature space. The collective model parameter is $\theta = (\theta^h, \theta^f)$. The objective for a client's local training is:

$$J_k(\theta) := \frac{1}{|\mathcal{D}_k|} \sum\nolimits_{(x,y)\in\mathcal{D}_k} \ell(h(f(x;\theta^f);\theta^h), y), \tag{1}$$

where $\mathcal{D}_k$ is the local dataset for client k, and l is the cross-entropy loss. The overall objective function of FL is thus given by:

$$\min_{\boldsymbol{\theta}} \frac{1}{K} \sum\nolimits_{k=1}^{K} J_k(\theta). \tag{2}$$

DNN watermarking. Existing DNN watermarking can be categorized into two main streams: parameter-based watermarking and backdoor-based watermarking.

Parameter-based watermarking approaches [7, 34, 18, 27] embed a bit string as the watermark into the parameter space of the model. The ownership of the model can be verified by comparing the watermark extracted from the parameter space of the suspect model and the owner model. Shao *et al.* [30] proposed a parameter-based watermarking method for FL called FedTracker. It inserts a unique parameter-based watermark into the models of each client to verify the ownership. However, all parameter-based watermarking requires an inspection of the parameters of the suspect models, which is not applicable enough for many re-sale models.

Backdoor-based watermarking [36, 19, 11, 21] does not require access to model parameters during verification. The watermark is embedded into the model by fine-tuning the model with a trigger set $\mathcal{D}_T$ and clean training samples $\mathcal{D}$. Pre-defined target label t is assigned to the trigger set $\mathcal{D}_T$. The objective for the backdoor-based watermarking is formulated as:

$$J(\theta) := \frac{1}{|\mathcal{D}|} \sum\nolimits_{(x,y)\in\mathcal{D}} \ell(h(f(x;\theta^f);\theta^h), y) + \frac{1}{|\mathcal{D}_T|} \sum\nolimits_{(x,t)\in\mathcal{D}_T} \ell(h(f(x;\theta^f);\theta^h), t), \tag{3}$$

Upon verification, we verify the suspect model M_s on the trigger set $\mathcal{D}_T$. If the accuracy of the trigger set is larger than a certain threshold σ, the ownership of the model can be established. We formally define the ownership verification of the backdoor-based model as follows:

Definition 2.1 (Ownership verification)**.** We define watermark success rate (WSR) as the accuracy on the trigger set $\mathcal{D}_T$:

$$\text{WSR} = Acc(M_s, \mathcal{D}_T). \tag{4}$$

If WSR $> \sigma$, the ownership of the model is established.

WAFFLE [33] is the first backdoor-based watermarking for FL, which utilized only random noise and class-consistent patterns to embed a backdoor-based watermark into the FL model. However, WAFFLE can only verify the ownership of the model, yet it cannot track the specific IP infringers among the clients.

3 Method

Watermarking has shown to be a feasible solution for IP verification, and the major goal of this work is to seek a powerful extension for traceable IP verification for accountable FL that can accurately identify the infringers among a scalable number of clients. A straightforward solution is injecting different watermarks for different clients. However, increasing the number of watermarks could lower the model's utility as measured by the standard accuracy due to increased forged knowledge [32] (R3). Meanwhile, maintaining multiple watermarks could be less robust to watermark removal because of the inconsistency between injections (R4). Accurate IP tracking (R1) is one unique requirement we seek as compared with traditional watermarking in central training. The greatest challenge in satisfying R1 is addressing the watermark *collisions* between different clients. A watermark collision is when the watermarking mechanism produces similar watermark responses on different individuals in FL systems. Formally:

Definition 3.1 (Watermark collision)**.** During verification in Definition 2.1, we test the suspect model M_s on all the verification datasets $\mathcal{D}_T = \{\mathcal{D}_{T_1}, \ldots, \mathcal{D}_{T_k}, \ldots, \mathcal{D}_{T_K}\}$ of all the clients to identify the malicious client, and WSR for the k-th verification datasets is defined as WSR_k. If we have multiple clients k satisfying $\text{WSR}_k = Acc(M_s, \mathcal{D}_{T_k}) > \sigma$, the ownership of suspect model M_s can be claimed for more than one client, then the watermark collisions happen between clients.

In this section, we introduce the Decodable Unique Watermark (DUW) framework that can simultaneously address the four requirements of accountable FL summarized in Section 1: R1 (accurate IP tracking), R2 (confident verification), R3 (model utility), R4 (robustness).

3.1 Decodable Unique Watermarking

In DUW, all the watermarking is conducted on the server side and therefore no computational overhead is introduced to clients. Before broadcasting the global model to each local client, the server will inject a unique watermark for each client. The watermark is unknown to clients but known to the server (see Fig. 1 server watermark injection). Our unique decodable watermarks consist of the following two steps for encoding and decoding the client-unique keys.

Step 1: Client-unique trigger encoding. Due to the data confidentiality of FL, the server has no access to any data from any of the clients. Therefore for watermark injection, the server needs to collect or synthesize some OoD data for trigger set generation.

To accurately track the malicious client, we have to distinguish between watermarks for different clients. High similarity between trigger sets of different clients is likely to cause watermark collisions among the clients, which makes it difficult to identify which client leaked the model.

To solve this problem, inspired by [24], we propose to use a pre-trained encoder $E : \mathcal{X} \to \mathcal{X}$ governed by θ_E to generate unique trigger sets for each client. This backdoor-based method provides a successful injection of watermarks with close to 100% WSR, which ensures the confident verification (R2). We design a unique key corresponding to each client ID as a one-hot binary string to differentiate clients. For instance, for the k-th client, the k-th entry of the key string s_k is 1, and the other entries are 0. We set the length of the key as d, where $d \geq K$. For each client, the key can then be embedded into the sample-wise triggers of the OoD samples by feeding the unique key and OoD data to the pre-trained encoder. The output of the encoder makes up the trigger sets. The trigger set for the k-th client is defined in Eq. (5).

$$\mathcal{D}_{T_k} = \{(x', t_k) | x' \sim E_{x \in \mathcal{D}_{OoD}}(x, s_k; \theta_E)\}, \tag{5}$$

where $\mathcal{D}_{OoD}$ is a randomly chosen OoD dataset, and t_k is the target label for client k. To this end, different trigger sets for different clients will differ by their unique keys, and watermark collision can be alleviated (R1). Note that our trigger sets will be the same as verification datasets.

Step 2: Decoding triggers to client keys. The main intuition is that the same target label of the trigger sets may still lead to watermark collisions even if the keys are different, which is also found

Algorithm 1 Injection of Decodable Unique Watermarking (DUW)

Input: Clients datasets$\{\mathcal{D}_k\}_{k=1}^K$, OoD dataset $\mathcal{D}_{\text{OoD}}$, secret key $\{s_k\}_{k=1}^K$, pre-trained encoder E, pre-defined decoder D, global parameters θ_g, local parameters $\{\theta_k\}_{k=1}^K$, learning rate α,β, local training steps T, watermark injection steps T_w.
Step 1: Client-unique trigger encoding.
for $k = 1,\ldots, K$ **do**
  Generate trigger set for client k: $\mathcal{D}_{T_k} = \{(x', s_k)|x' \sim E_{x\in\mathcal{D}_{OoD}}(x, s_k; \theta_E)\}$
end for
Step 2: Decoding triggers to client keys.
repeat
  Server selects active clients $\mathcal{A}$ uniformly at random
  for all client $k \in \mathcal{A}$ **do**
    Server initializes watermarked model for client k as: $\theta_k \leftarrow \theta_g$.
    for $t = 1, \ldots, T_w$ **do**
      Server replaces model classifier h with decoder D.
      Server injects watermark to model, and update θ_k^f as:
      $\theta_k^f \leftarrow \theta_k^f - \beta\nabla_{\theta_k^f} J'(\theta_k^f)$. ▷ Optimize Eq. (7)
    end for
    Server broadcasts θ_k to the corresponding client k.
    for $t = 1, \ldots, T$ **do**
      Client local training: $\theta_k \leftarrow \theta_k - \alpha\nabla_{\theta_k} J_k(\theta_k)$. ▷ Optimize Eq. (1)
    end for
    Client k sends $\boldsymbol{\theta}_k$ back to the server.
  end for
  Server updates $\theta_g \leftarrow \frac{1}{|\mathcal{A}|}\sum_{k\in\mathcal{A}}\theta_k$.
until training stop

in our experiment section. Thus, we propose to project the output dimension of the original model M to a higher dimension, larger than the client number K, to allow each client to have a unique target label. To achieve this goal, we first set the target label t_k in Eq. (5) to be the same as the input key s_k corresponding to each client, and then use a decoder $D : \mathcal{Z} \to \mathcal{Y}$ parameterized by θ_D to replace the classifier h in the FL training model M. The decoder D only has one linear layer, whose input dimension is the same as the input dimension of h, and its output dimension is the length of the key. To avoid watermark collision between clients induced by the target label, we make the decoder weights orthogonal with each other during the random initialization so that the watermark injection tasks for each client can be independent (R1). The weights of the decoder are frozen once initialized to preserve the independence of different tasks for different clients during watermark injection. We formulate the injection optimization for client k as:

$$\min_{\theta_k^f} J(\theta_k^f) := \frac{1}{|\mathcal{D}_{T_k}|}\sum\nolimits_{(x',s_k)\in\mathcal{D}_{T_k}} \ell(D(f(x'; \theta_k^f); \theta_D), s_k), \tag{6}$$

where ℓ is the cross-entropy loss. Note that the classifier h will be plugged back into model M before the server broadcast the watermarked model to each client. Compared with traditional backdoor-based watermarking Eq. (3), no training samples from clients are needed for the watermark injection, which ensures the data confidentiality of FL.

Robustness. Our framework also brings in robustness against fine-tuning-based watermark removal (R4). The main intuition is that replacing classifier h with decoder D also differs the watermark injection task space from the original classification task space. Since the malicious clients have no access to the decoder and can only conduct attacks on model M, the attacks have more impact on the classification task instead of our watermark injection task, which makes our decodable watermark to be more resilient against watermark removal attacks.

3.2 Injection Optimization with Preserved Utility

While increasing the size of the key pool, watermark injection in the OoD region may lead to a significant drop in the standard FL accuracy (R3) because of the overload of irrelevant knowledge.

An ideal solution is to bundle the injection with training in-distribution (ID) data, which however is impractical for a data-free server. Meanwhile, lacking ID data to maintain the standard task accuracy, the distinct information between the increasing watermark sets and the task sets could cause the fade-out of the task knowledge. We attribute such knowledge vanishing to the divergence in the parameter space between the watermarked and the original models. Thus, we propose to augment the injection objective Eq. (6) with a l_2 regularization on the parameters:

$$\min_{\theta_k} J'(\theta_k^f) := J(\theta_k^f) + \frac{\beta}{2}\|\theta_k^f - \theta_g^f\|^2, \tag{7}$$

where θ_g^f is the original parameter of the global model. The regularization term of Eq. (7) is used to restrict the distance between the watermarked model and the non-watermarked one so that the utility of the model can be better preserved (R3). To this end, the overall watermark injection algorithm is summarized in Algorithm 1.

3.3 Verification

During verification, we not only verify whether the suspect model $M_s = (f_s, h_s)$ is a copy of our model M, but also track who is the leaker among all the clients. Before verifying, we first use our decoder D to replace the classifier h_s in the suspect model M_s, then the suspect model can be restructured as $M_s = (f_s, D)$. According to Definition 3.1, we test the suspect model M_s on all the verification datasets $\mathcal{D}_T = \{\mathcal{D}_{T_1}, \ldots, \mathcal{D}_{T_k}, \ldots, \mathcal{D}_{T_K}\}$ of all the clients to track the malicious clients, and report WSR_k on the k-th verification datasets correspondingly. The client whose verification dataset achieves the highest WSR leaked the model (see Fig. 1 server verification). The tracking mechanism can be defined as follows:

$$\textit{Track}(M_s, \mathcal{D}_T) = \arg\max_k \text{WSR}_k.$$

Suppose the malicious client is k_m, which is the ground truth of the suspect model. If WSR_{k_m} is larger than a threshold σ, and WSR_k for other verification datasets is smaller than σ, then the ownership of the model can be verified, and no watermark collision happens. If $\textit{Track}(M_s, \mathcal{D}_T) = k_m$, then the malicious client is identified correctly.

4 Experiments

In this section, we empirically show how our proposed DUW can fulfill the requirements (R1-R4) for tracking infringers as described in Section 1.

Datasets. To simulate *class non-iid* FL setting, we use two image datasets CIFAR-10, CIFAR-100 [17], which contain 32×32 images with 10 and 100 classes, respectively. CIFAR-10 data is uniformly split into 100 clients, and 3 random classes are assigned to each client. CIFAR-100 data is split into 100 clients with Dirichlet distribution. For CIFAR-10 and CIFAR-100, the OoD dataset we used for OoD injection is a subset of ImageNet-DS [5] with randomly chosen 500 samples downsampled to the same image size as CIFAR-10 and CIFAR-100. To simulate the *feature non-iid* FL setting, a multi-domain FL benchmark, Digits [22, 13], is adopted. The dataset is composed of 28×28 images for recognizing 10 digit classes, which task was widely used in the community [3, 26]. The Digits dataset includes five different domains: MNIST [20], SVHN [28], USPS [14], SynthDigits [9], and MNIST-M [9]. We leave out USPS as the OoD dataset for watermark injection (a subset of 500 samples is chosen), and use the rest of the four domains for the standard FL training. Each domain of digits is split into 10 different clients, thus, 40 clients will participate in the FL training.

Training setup. A preactivated ResNet (PreResNet18) [12] is used for CIFAR-10, a preactivated ResNet (PreResNet50) [12] is used for CIFAR-100, and a CNN defined in [23] is used for Digits. For all three datasets, we leave out 10% of the training set as the validation dataset to select the best FL model. The total training round is 300 for CIFAR-10 and CIFAR-100, and is 150 for Digits.

Watermark injection. The early training stage of FL is not worth protecting since the standard accuracy is very low, we start watermark injection at round 20 for CIFAR-10 and Digits, and at round 40 for CIFAR-100. The standard accuracy before our watermark injection is 0.8520, 0.4023, and 0.2941 for Digits, CIFAR-10, and CIFAR-100, respectively.

Dataset	Acc	ΔAcc	WSR	WSR_Gap	TAcc
Digits	0.8855	0.0234	0.9909	0.9895	1.0000
CIFAR-10	0.5583	0.0003	1.0000	0.9998	1.0000
CIFAR-100	0.5745	0.0063	1.0000	0.9998	1.0000

Table 1: Benchmark results.

Dataset	Acc	ΔAcc	WSR	ΔWSR	TAcc
Digits	0.9712	-0.0258	0.9924	0.0030	1.0000
CIFAR-10	0.7933	0.1521	1.0000	0.0000	1.0000
CIFAR-100	0.4580	0.0290	0.9930	0.0070	1.0000

Table 2: DUW is robust against fine-tuning.

Dataset	Acc	ΔAcc	WSR	ΔWSR	TAcc
Digits	0.8811	0.0643	0.9780	0.0174	1.0000
CIFAR-10	0.5176	0.0010	0.6638	0.3362	1.0000
CIFAR-100	0.4190	0.0680	0.8828	0.1172	1.0000

Table 3: DUW is robust against model extraction.

Evaluation metrics. For watermark verification, we use watermark success rate (**WSR**) which is the accuracy of the trigger set for evaluation. To measure whether we track the malicious client (leaker) correctly, we define tracking accuracy (**TAcc**) as the rate of the clients we track correctly. To further evaluate the ability of our method for distinguishing between different watermarks for different clients, we also report the difference between the highest WSR and second best WSR as **WSR_Gap** to show the significance of verification and IP tracking. With a significant WSR_Gap, no watermark collision will happen. To evaluate the utility of the model, we report the standard FL accuracy (**Acc**) for each client's individual test sets, whose classes match their training sets. We also report the accuracy degradation (**ΔAcc**) of the watermarked model compared with the non-watermarked one. Note that, to simulate the scenario where malicious clients leak their local model after local training, we test the average WSR, TAcc and WSR_Gap for the local model of each client instead of the global model. Acc and ΔAcc are evaluated on the best FL model selected using the validation datasets.

4.1 IP Tracking Benchmark

We evaluate our method using the IP tracking benchmark with various metrics as shown in Table 1. Our ownership verification is confident with all WSRs over 99% (R2). The model utility is also preserved with accuracy degradation 2.34%, 0.03%, and 0.63%, respectively for Digits, CIFAR-10 and CIFAR-100 (R3). TAcc for all benchmark datasets is 100% which indicates accurate IP tracking (R1). All WSR_Gap is over 98%, which means the WSRs for all other benign client's verification datasets are close to 0%. In this way, the malicious client can be tracked accurately with high confidence, no collisions will occur within our tracking mechanism (R1).

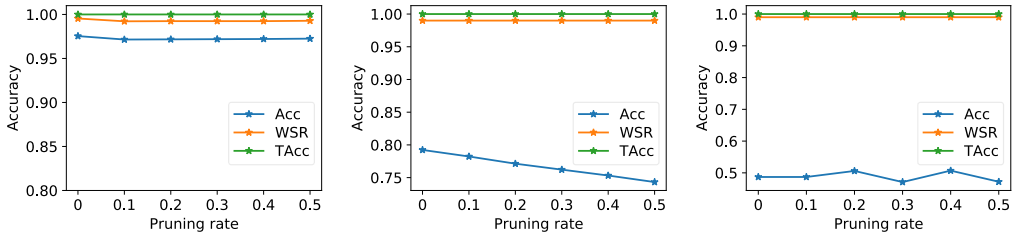
4.2 Robustness

Malicious clients can conduct watermark removal attacks before leaking the FL model to make it harder for us to verify the model copyright, and track the IP infringers accurately. In this section, we show the robustness of the watermarks under various watermark removal attacks (R4). Specifically, we evaluate our method against 1) **fine-tuning** [2]: Fine-tune the model using their own local data; 2) **pruning** [25]: prune the model parameters which have the smallest absolute value according to a certain pruning rate, and then fine-tune the model on their local data; 3) **model extraction attack**: first query the victim model for the label of an auxiliary dataset, and then re-train the victim model on the annotated dataset. We take knockoff [29] as an example of the model extraction attack; 4) **parameter perturbations**: add random noise to local model parameters [10].

10 of the clients are selected as the malicious clients, and the metrics in this section are average values for 10 malicious clients. All the watermark removal attacks are conducted for 50 epochs with a learning rate 10^{-5}. All the attacks are conducted for the local model of the last round.

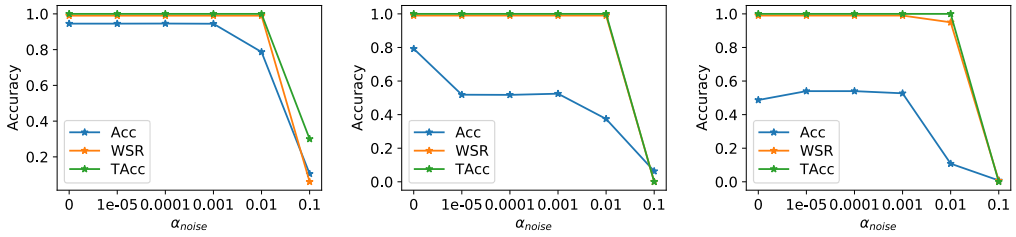
Robustness against fine-tuning attack. We report the robustness of our proposed DUW against fine-tuning in Table 2. ΔAcc and ΔWSR in this table indicate the accuracy and WSR drop compared with accuracy and WSR before the attack. According to the results, after 50 epochs of fine-tuning, the attacker can only decrease the WSR by less than 1%, and the TAcc is even not affected. Fine-tuning with their limited local training samples can also cause a standard accuracy degradation. Therefore, fine-tuning can neither remove our watermark nor affect our IP tracking, even if sacrifices their standard accuracy.

Robustness against pruning attack. We investigate whether pruning can remove our proposed DUW in Fig. 2 by varying the pruning rate from 0 to 0.5. With the increase in the pruning ratio, both TAcc and WSR will not be affected. For CIFAR-10, Acc for the standard classification task will



(a) Digits. (b) CIFAR-10. (c) CIFAR-100.

Figure 2: DUW is robust against pruning.



(a) Digits. (b) CIFAR-10. (c) CIFAR-100.

Figure 3: DUW is robust against parameter perturbation.

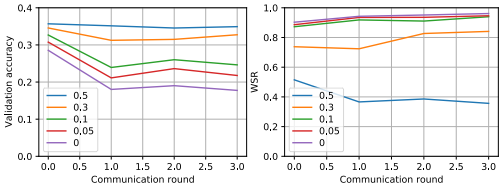
drop 5%. Pruning is not an effective attack on our watermark, and it will even cause an accuracy degradation for the classification task.

Robustness against model extraction attack. To verify the robustness of our proposed DUW against model extraction attack, we take knockoff [29] as an example, and STL10 [6] cropped to the same size as the training data is used as the auxiliary dataset for this attack. According to the results for three benchmark datasets in Table 3, after knockoff attack, WSR for all three datasets is still over 65%, and our tracking mechanism is still not affected with TAcc remains to be 100%. Therefore, our DUW is resilient to model extraction attacks.

Robustness against parameter perturbations attack. Malicious clients can also add random noise to model parameters to remove watermarks, since Garg *et al.* [10] found that backdoor-based watermarks are usually not resilient to parameter perturbations. Adding random noise to the local model parameters can also increase the chance of blurring the difference between different watermarked models. We enable each malicious client to blend Gaussian noise to the parameters of their local model, and set the parameter of the local model as $\theta_i = \theta_i + \theta_i * \alpha_{\text{noise}}$, where $\alpha_{\text{noise}} = \{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$. We investigate whether parameter perturbation will remove the watermark and affect our tracking mechanism in Fig. 3. According to the results, when α_{noise} is smaller than 10^{-2}, WSR, Acc, and TAcc will not be affected. When $\alpha_{\text{noise}} = 10^{-2}$, Acc will drop more than 10%, TAcc remains unchanged, and WSR is still over 90%. When $\alpha_{\text{noise}} = 10^{-1}$, Acc will drop to a random guess, thus, although the watermark has been removed, the model has no utility. Therefore, parameter perturbation is not an effective attack for removing our watermark and affecting our tracking mechanism.

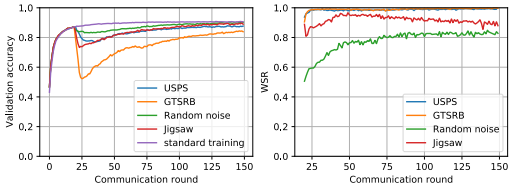
4.3 Qualitative Study

Effects of decoder. To investigate the effects of the decoder on avoiding watermark collision, we compare the results of w/ and w/o decoder. When the decoder is removed, the task dimension of the watermark injection will be the same as the FL classification, thus, we also have to change the original target label (the original target label is the same as the input key) of the trigger set to the FL classification task dimension. To achieve this goal, we set the target label of w/o decoder case as (client_ID $\%$ class_number). For instance, client 0, 10, 20 will have target label 0, and client 1, 11, 21 will have target label 1. We report the results of w/ and w/o decoder on CIFAR-10 after 1 round of watermark injection at round 20 in Table 4. According to the results, when we have 100 clients in total, w/o decoder can only achieve a TAcc of 6%, while w/ decoder can increase TAcc to 100%. We also find that clients with the same target label are more likely to conflict with each other, which makes those clients difficult to be identified, even if their trigger sets are different. Utilizing a decoder



(a) Validation accuracy. (b) WSR.

Figure 4: Validation accuracy and WSR for different values of β for 4 communication rounds.



(a) Validation accuracy. (b) WSR.

Figure 5: Validation accuracy and WSR for different OoD datasets by communication rounds.

Method	Acc	ΔAcc	WSR	TAcc
w/ decoder	0.3287	0.0736	**0.8778**	**1.0000**
w/o decoder	0.3235	0.0788	0.8099	0.0600

Table 4: Effects of decoder: the decoder can improve TAcc to avoid watermark collision. ΔAcc in this table is the accuracy degradation compared with the previous round.

Dataset	Acc	ΔAcc	WSR	WSR_Gap	TAcc
USPS	0.8855	0.0234	0.9909	0.9895	1.0000
GTSRB	0.8716	0.0373	0.9972	0.9962	1.0000
Random noise	0.9007	0.0082	0.8422	0.8143	1.0000
Jigsaw	0.9013	0.0076	0.8789	0.8601	1.0000

Table 5: Effects of different OoD datasets: a trade-off exists between Acc and WSR, given different selections of OoD datasets.

to increase the target label space to a dimension larger than the client number allows all the clients to have their own target label. In this way, watermark collision can be avoided. Besides, WSR of w/ decoder is also higher than w/o decoder after 1 round of injection. One possible reason is that we differ the watermark injection task from the original classification task using the decoder, thus, in this case, the watermark will be more easily to be injected compared with directly injected to the original FL classification task.

Effects of l_2 regularization. To show the effects of l_2 regularization in Eq. (7), we report the validation accuracy and WSR for 4 rounds of watermark injection on Digits with different values of the hyperparameter β in Fig. 4. Validation accuracy is the standard FL accuracy evaluated on a validation dataset for every round. We see that with the increase of β, higher validation accuracy can be achieved, but correspondingly, WSR drops from over 90% to only 35.65%. Larger β increases the impact of l_2 norm, which decreases the model difference between the watermarked model and the non-watermarked one, so the validation accuracy will increase. At the same time, the updates during watermark injection also have much more restriction due to l_2 regularization, so the WSR drops to a low value. Accordingly, we select $\beta = 0.1$ for all our experiments, since $\beta = 0.1$ can increase validation accuracy by 6.88% compared with $\beta = 0$, while maintaining WSR over 90%.

Effects of different OoD datasets for watermark injection. We investigate the effects of different OoD datasets including USPS [14], GTSRB [31], random noise, and Jigsaw for watermark injection when the standard training data is Digits. All OoD images are cropped to the same size as the training images. A jigsaw image is generated from a small 4×4 random image, and then uses reflect padding mode from PyTorch to padding to the same size as the training images. The effect of these different OoD datasets is shown in Table 5 and Fig. 5. We see that all OoD datasets can achieve 100% TAcc, suggesting the selection of OoD dataset will not affect the tracking of the malicious client. There is a trade-off between the standard accuracy (Acc) and WSR: higher WSR always leads to lower Acc. Random noise and jigsaw achieve high Acc, with accuracy degradation within 1%. These two noise OoD also have a faster recovery of the standard accuracy after the accuracy drop at the watermark injection round as shown in Fig. 5. However, the WSR of random noise and Jigsaw are lower than 90%. For two real OoD datasets USPS and GTSRB, the WSR quickly reaches over 99% after 1 communication round. However, their accuracy degradation is larger than 2%.

5 Conclusion

In this paper, we target at accountable FL, and propose Decodable Unique Watermarking (DUW), that can verify the FL model's ownership and track the IP infringers in the FL system at the same time. Specifically, the server will embed a client-unique key into each client's local model before broadcasting. The IP infringer can be tracked according to the decoded keys from the suspect model. Extensive experimental results show the effectiveness of our method in accurate IP tracking, confident verification, model utility preserving, and robustness against various watermark removal attacks.

Acknowledgement

This research was supported by the National Science Foundation (IIS-2212174, IIS-1749940), National Institute of Aging (IRF1AG072449).

References

[1] Meta's powerful ai language model has leaked online — what happens now? https://www.theverge.com/2023/3/8/23629362/meta-ai-language-model-llama-leak-online-misuse. Accessed: 2023-03-08.

[2] 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)*, pages 1615–1631, 2018.

[3] S. Caldas, S. M. K. Duddu, P. Wu, T. Li, J. Konečnỳ, H. B. McMahan, V. Smith, and A. Talwalkar. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*, 2018.

[4] X. Chen, T. Chen, Z. Zhang, and Z. Wang. You are caught stealing my winning lottery ticket! making a lottery ticket claim its ownership. *Advances in Neural Information Processing Systems*, 34:1780–1791, 2021.

[5] P. Chrabaszcz, I. Loshchilov, and F. Hutter. A downsampled variant of imagenet as an alternative to the cifar datasets. *arXiv preprint arXiv:1707.08819*, 2017.

[6] A. Coates, A. Ng, and H. Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223. JMLR Workshop and Conference Proceedings, 2011.

[7] 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*, pages 485–497, 2019.

[8] L. Fan, K. W. Ng, and C. S. Chan. Rethinking deep neural network ownership verification: Embedding passports to defeat ambiguity attacks. *Advances in neural information processing systems*, 32, 2019.

[9] Y. Ganin and V. Lempitsky. Unsupervised domain adaptation by backpropagation. In *International conference on machine learning*, pages 1180–1189. PMLR, 2015.

[10] S. Garg, A. Kumar, V. Goel, and Y. Liang. Can adversarial weight perturbations inject neural backdoors. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 2029–2032, 2020.

[11] M. Goldblum, D. Tsipras, C. Xie, X. Chen, A. Schwarzschild, D. Song, A. Mądry, B. Li, and T. Goldstein. Dataset security for machine learning: Data poisoning, backdoor attacks, and defenses. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(2):1563–1580, 2022.

[12] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[13] J. Hong, H. Wang, Z. Wang, and J. Zhou. Efficient split-mix federated learning for on-demand and in-situ customization. *arXiv preprint arXiv:2203.09747*, 2022.

[14] J. J. Hull. A database for handwritten text recognition research. *IEEE Transactions on pattern analysis and machine intelligence*, 16(5):550–554, 1994.

[15] J. Konečnỳ, B. McMahan, and D. Ramage. Federated optimization: Distributed optimization beyond the datacenter. *arXiv preprint arXiv:1511.03575*, 2015.

[16] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[17] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[18] M. Kuribayashi, T. Tanaka, S. Suzuki, T. Yasui, and N. Funabiki. White-box watermarking scheme for fully-connected layers in fine-tuning model. In *Proceedings of the 2021 ACM Workshop on Information Hiding and Multimedia Security*, pages 165–170, 2021.

[19] E. Le Merrer, P. Perez, and G. Trédan. Adversarial frontier stitching for remote neural network watermarking. *Neural Computing and Applications*, 32:9233–9244, 2020.

[20] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[21] F. Li, L. Yang, S. Wang, and A. W.-C. Liew. Leveraging multi-task learning for umambiguous and flexible deep neural network watermarking. In *SafeAI@ AAAI*, 2022.

[22] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2:429–450, 2020.

[23] X. Li, M. Jiang, X. Zhang, M. Kamp, and Q. Dou. Fedbn: Federated learning on non-iid features via local batch normalization. *arXiv preprint arXiv:2102.07623*, 2021.

[24] Y. Li, Y. Li, B. Wu, L. Li, R. He, and S. Lyu. Invisible backdoor attack with sample-specific triggers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16463–16472, 2021.

[25] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell. Rethinking the value of network pruning. *arXiv preprint arXiv:1810.05270*, 2018.

[26] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[27] D. Mehta, N. Mondol, F. Farahmandi, and M. Tehranipoor. Aime: watermarking ai models by leveraging errors. In *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 304–309. IEEE, 2022.

[28] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[29] T. Orekondy, B. Schiele, and M. Fritz. Knockoff nets: Stealing functionality of black-box models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4954–4963, 2019.

[30] S. Shao, W. Yang, H. Gu, J. Lou, Z. Qin, L. Fan, Q. Yang, and K. Ren. Fedtracker: Furnishing ownership verification and traceability for federated learning model. *arXiv preprint arXiv:2211.07160*, 2022.

[31] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural networks*, 32:323–332, 2012.

[32] R. Tang, M. Du, N. Liu, F. Yang, and X. Hu. An embarrassingly simple approach for trojan attack in deep neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 218–228, 2020.

[33] 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)*, pages 310–320. IEEE, 2021.

[34] 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*, pages 269–277, 2017.

[35] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 707–723. IEEE, 2019.

[36] J. Zhang, Z. Gu, J. Jang, H. Wu, M. P. Stoecklin, H. Huang, and I. Molloy. Protecting intellectual property of deep neural networks with watermarking. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, pages 159–172, 2018.