WAFFLE: Watermarking in Federated Learning

Buse G. A. Tekgul*, Yuxi Xia*, Samuel Marchal†, and N. Asokan ‡*
*Department of Computer Science, Aalto University, 02150, Espoo, Finland
Email: buse.atlitekgul@aalto.fi, yuxi.xia.work@gmail.com
†F-Secure Corporation, 00180, Helsinki, Finland
Email: samuel.marchal@aalto.fi
‡ University of Waterloo, Waterloo, ON N2L 3G1, Canada
Email: asokan@acm.org

***Abstract*—*Federated learning* is a distributed learning technique where machine learning models are trained on client devices in which the local training data resides. The training is coordinated via a central server which is, typically, controlled by the intended owner of the resulting model. By avoiding the need to transport the training data to the central server, federated learning improves privacy and efficiency. But it raises the risk of *model theft* by clients because the resulting model is available on every client device. Even if the application software used for local training may attempt to prevent direct access to the model, a malicious client may bypass any such restrictions by reverse engineering the application software. *Watermarking* is a well-known deterrence method against model theft by providing the means for model owners to demonstrate ownership of their models. Several recent deep neural network (DNN) watermarking techniques use *backdooring*: training the models with additional mislabeled data. Backdooring requires full access to the training data and control of the training process. This is feasible when a single party trains the model in a centralized manner, but not in a federated learning setting where the training process and training data are distributed among several client devices. In this paper, we present WAFFLE, the first approach to watermark DNN models trained using federated learning. It introduces a re-training step at the server after each aggregation of local models into the global model. We show that WAFFLE efficiently embeds a resilient watermark into models incurring only negligible degradation in test accuracy (-0.17%), and does not require access to training data. We also introduce a novel technique to generate the backdoor used as a watermark. It outperforms prior techniques, imposing no communication, and low computational ($+3.2\%$) overhead[1].**

***Index Terms*—Federated learning, ownership demonstration, watermarking, deep learning**

I. INTRODUCTION

Distributed machine learning has gained considerable attention in many big data processing applications such as recommendation systems, smart assistants, and health-care, due to its efficient parallelization and scalability properties. Federated learning [1] is an instance of privacy-preserving distributed machine learning that allows decentralized training of deep neural networks (DNNs) by many parties holding local data samples. Major companies have already utilized federated learning in various large-scale applications such as improving the Google Keyboard (Gboard) suggestions [2], [3] or Apple's voice recognition models [4]. In these applications, clients only need to download a set of tools or a specific software (e.g., Tensorflow Lite[2]) into their mobile devices, and execute the training with the on-device, cached data. Hence, instead of using publicly available or synthetic datasets that often do not represent the real-world distribution, model owners can deploy high-quality models with no access to the possibly sensitive training data.

The most common federated learning setting in commercial, global-scale applications is *client-server* involving three parties: *model owner*, *aggregator* (hosted on a server), and a large number of *clients* training a common *global model* [5]. Federated learning consists of iterative *aggregation rounds* where (1) the aggregator sends the global model parameters to clients, (2) each client replaces parameters of its local model using the global model, re-trains it using the on-device data, and sends updated parameters back to the aggregator, which (3) combines them into a new global model. The final global model is delivered to the model owner when the training is completed. Since each client's training data stays in place, federated learning preserves privacy of clients' data and avoids incurring substantial costs for transferring training data from clients to a centralized trainer.

Despite its advantages, federated learning brings forth the issue of preserving ownership. In large-scale, client-server federated learning applications, there are multiple clients who are data-owners and *can use the model* in their local devices, but there is *only one model owner.* However, a side-effect of the training process is that each client can gain complete access to the global model in every round, including the final one by reverse engineering the application software [6] or on-device dynamic analysis [7]. Although the model owner can deploy further mechanisms for hiding model parameters during the local training (e.g. homomorphic encryption), this might lead to high latency, bandwidth costs, and additional requirements for benign clients [8]. It is therefore useful to have a means of demonstrating the intended ownership in

This work was supported in part by Intel (in the context of the Private-AI Institute) and SAPPAN, a project funded by the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement no. 833418. We also thank Sebastian Szyller for interesting discussions, and Aalto Science-IT project for computational resources.

[1]The research report version of this paper is also available in https://arxiv.org/abs/2008.07298, and the code for reproducing our work can be found at https://github.com/ssg-research/WAFFLE

[2]ML for Mobile and Edge Devices, https://www.tensorflow.org/lite/

2575-8462/21/$31.00 ©2021 IEEE
DOI 10.1109/SRDS53918.2021.00038

case a malicious client uses the global model residing on their devices in unauthorized ways such as monetizing the model or making it available to their own customers.

Recently, different watermarking techniques [9]–[13] have been proposed to demonstrate ownership of DNN models. In a typical backdoor-based watermarking procedure, the model owner first designs a secret *watermark* which consists of mislabeled input output pairs. Then, the model owner trains the model with both the training dataset and the watermark in order to embed the watermark into the model. This watermark can be subsequently used to demonstrate ownership. However, existing DNN watermarking solutions cannot be directly applied in federated learning for two reasons. First, model owners cannot use techniques for generating watermarks using training data because they lack access to the training data (e.g., Gboard [3]). Second, training is performed in parallel by several clients, some of them being potentially malicious. Although some "trusted clients" can join the watermark embedding phase, the training setup and aggregation rules have to be changed to handle them differently. Thus, watermark embedding cannot be implemented easily by clients.

Our goal is to design a procedure for effectively embedding watermarks into DNN models trained via client-server federated learning, without decreasing the accuracy of the resulting global model while minimizing the computational and communication overhead imposed on the distributed training process. We claim the following contributions:

1) **Problem definition**: We identify the problem of demonstrating ownership of models trained via client-server federated learning and define requirements for a solution addressing this problem (Section III-B).
2) **Watermarking procedure for federated learning**: We introduce WAFFLE: the first solution for addressing ownership problem in client-server federated learning (Section IV). WAFFLE leverages capabilities of the aggregator to embed a backdoor-based watermark [9] by re-training the global model with the watermark during each aggregation round. We show on MNIST and CIFAR10 using two DNN architectures that WAFFLE embeds watermarks without degrading the model performance (Section VI-C).
3) **Data-independent watermark generation method**: We introduce WAFFLEPATTERN, a novel data-independent method to generate watermarks for DNN image classification models (Section IV-C). It generates images having random but class-consistent patterns on a random background, which is suitable for federated learning. Compared to prior watermark generation methods [9], [13], [14], WAFFLEPATTERN retains model performance better (Section VI-C) and imposes lower communication and computational overhead (Section VI-D). We also show that WAFFLE with WAFFLEPATTERN is resilient to watermark removal techniques including fine-tuning, pruning and reverse-engineering (Section VI-B) if no more than 10% of the clients collude to defeat the watermarking procedure.

Our procedure focuses on embedding effective watermarks into models trained with federated learning. We do not discuss the demonstration of ownership using our watermarks because secure schemes [9], [11] for registering and using backdoor-based watermarks to reliably demonstrate ownership of DNN models already exist and these schemes can be used with our watermarks. We also leave discussions on the legal validity of DNN model watermarks as out of scope.

II. BACKGROUND

A. Federated Learning

The client-server federated learning is composed of three main parties: 1) A large number of clients $C = \{c_j\}_{j=1}^{K}$ who are data owners and keep their datasets D_{c_j} private, 2) a model owner O providing a randomly initialized global model w_G at the beginning of federated learning and getting it trained at the end, and 3) a secure aggregator Agg [15] located between O and C. We focus on training a DNN model, which is a function $F(x, w) : \mathbb{R}^n \rightarrow \mathbb{R}^m$ that we simply refer to as its parameters w, e.g., w_G. O cannot obtain any information about D_{c_j} due to the secure aggregation protocol [15] implemented by Agg. In this work, we focus on federated learning using the FederatedAveraging (FedAvg) algorithm [1], a widely used aggregation rule. In FedAvg, clients train their models using stochastic gradient descent (SGD). Before federated learning starts, w_G is initialized by O. In aggregation round t,

1) Agg sends $w_{G(t)}$ to a subset of clients $C_{sub} = \{c_i\}_{i=1}^{L}$, where $(L \ll K)$.
2) Each c_i updates $w_{c_i(t-1)}$ with $w_{G(t)}$, re-trains the updated $w_{c_i(t)}$ by a pre-determined number of local passes over its local dataset D_{c_i}, and sends the re-trained local model $w_{c_i(t)}$ to Agg.
3) Agg averages all local models into a new $w_{G(t+1)}$.

B. Watermarking DNN Models by Backdooring

Several works [9], [10], [12]–[14], [16] have proven the feasibility of embedding watermarks into DNNs for demonstration of ownership. In general, a watermark set WM_w consists of samples $\{x, B(x)\}$ designed by the owner of the DNN model w. The model owner embeds WM_w into w by optimizing w on both the training set and WM_w such that $B(x) = w^+(x)$ and $B(x) \neq w(x)$ for almost all $x \in \mathrm{WM}_w$, where $w^+(x)$ is the watermarked model. If the model owner suspects that another model w_{adv} is possibly derived from w^+, WM_w with a pre-defined verification algorithm VERIFY is used to demonstrate ownership if $\textsc{Verify}(w_{adv}, \mathrm{WM}_w)$ returns $True$.

In this paper, we focus on watermarking via backdooring [9], [10], [13]. A *backdoor* [17] consists of a *trigger set* of samples with incorrect labels. In computer vision, a trigger set usually contains specific patterns that can be added to an image [10] or a set of images unrelated to the actual task [9], [10], both of which serve a similar purpose. The backdoor is injected into a DNN by training it with both the correct training data and the backdoor. At inference time, the backdoored DNN performs normally on clean inputs but returns the expected

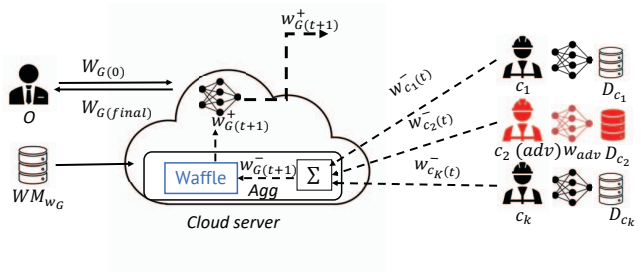


Fig. 1. Architecture and adversary model for watermarking in federated learning. Parties under the control of an adversary are highlighted in red.

incorrect label when a sample from the trigger set is given as an input. To reliably demonstrate that a model w_{adv} is derived from w^+, i.e., to obtain $\text{VERIFY}(w_{adv}, \text{WM}_w) \rightarrow True$, WM_w must be secret and registered in a timestamped public bulletin [9], [11]. In addition, the accuracy of w_{adv} on WM_w must be over a threshold T_{acc}: $Acc(w_{adv}, \text{WM}_w) \geq T_{acc}$. The value for T_{acc} is computed using the size of the watermark set $|\text{WM}_w|$ and the number of classes m for w [9], [11].

III. Ownership Demonstration in Federated Learning

A. Adversary Model

The ability to demonstrate ownership of machine learning models has emerged as an important concern because models can be monetized and represent a business advantage [11], [18], [19]. Figure 1 shows our adversary model. We consider malicious clients to be our primary adversaries. Following prior work [15], we assume that Agg is the only trusted party in federated learning. Agg resides in a central server, and is defined as an incorruptible third party [15], [20] between C and O. In some applications, the cloud server can belong to O (e.g., Gboard [3], Apple's Siri [4]) but Agg is still considered a *secure* party: it implements a secure aggregation protocol [15] and uses cryptographic primitives while routing messages between the server and clients. In unauthenticated or unencrypted network models like federated learning, Agg provides the strongest possible security for protecting data confidentiality [5], [15], [20].

1) Adversaries' goal: Adversaries aim to obtain a w_{adv} having the same performance as w_G on a test dataset D_{test}, so that $Acc(w_{adv}, D_{test}) \approx Acc(w_G, D_{test})$ while evading watermark detection, i.e., $\text{VERIFY}(w_{adv}, \text{WM}_{w_G}) \rightarrow False$, where WM_{w_G} is the watermark generated by O. We assume that an adversary $adv = c_i$ has the following capabilities:

- has access to its own relevant training data D_{adv}, but does not have access to other clients' training data. Each D_{c_i} is independent, identically distributed (IID), balanced and limited in size such that every c_i is incentivized to participate in federated learning. adv having a large D_{adv} would rather train its own model instead of stealing w_G.
- gains white-box access to $w_{adv(t)}$, and can save $w_{adv(t)}$ at any aggregation round t.
- can apply any post processing technique (e.g., fine-tuning, pruning, reverse engineering) to $w_{adv(t)}$ in order to remove watermarks.

2) Assumptions: We assume that every c_i is incentivized to maximize the accuracy of w_G. c_i does not backdoor [21], poison or embed its own watermark sets in w_G. These attacks might decrease the performance of w_G and w_{adv}, which is contrary to the goal of every c_i, including adv, who aims to obtain a model w_G/w_{adv} having the best performance possible. Furthermore, clients does not change the training scheme and use the same hyper-parameters provided by O (e.g., number of local passes E_c, loss function ℓ_c, learning rate η_c, batch size $|b_c|$), since they are optimized to maximize the accuracy of w_G.

B. Requirements

We define the following requirements for designing an effective watermarking scheme (**W1:3**) as well as performance conditions (**P1:3**) which must be met in federated learning. Performance conditions should be also taken into account for building other type of solutions for ownership demonstration of federated learning models.

W1 **Demonstration of ownership:** At any aggregation round t, $\text{VERIFY}(w_{adv(t)}, \text{WM}_{w_G})$ should return $True$, which requires $Acc(w_{adv(t)}, \text{WM}_{w_G}) \geq T_{acc}$.

W2 **Robustness:** WM_{w_G} embedded in w_{adv} should be resilient against attacks that try to remove it without destroying the performance of w_{adv}. This requirement is satisfied when **W1** still holds or $Acc(w_{adv}, D_{test}) \ll Acc(w^+_{adv}, D_{test})$ after applying a watermark removal attack to w_{adv}.

W3 **Data independence:** Watermarking procedure should not require training data knowledge, and O cannot use clients' training data in order to generate their own watermark set WM_{w_G}.

P1 **Model utility:** Watermarking should not degrade the performance of the local model, i.e., $Acc(w^+_{adv}, D_{test}) \approx Acc(w_{adv}, D_{test})$.

P2 **Low communication overhead:** Embedding WM_{w_G} should not increase the total number of aggregation rounds or the amount of the data exchanged to reach the convergence of w_G.

P3 **Low computation overhead:** Implementing watermark embedding with WM_{w_G} inside any party should incur minimal additional computation.

IV. Watermarking in Federated Learning

A. Current Challenges

Watermarking can help model owners to prove ownership of their DNN models. However, state-of-the-art techniques [9], [10], [12], [13] cannot be directly implemented in client-server federated learning, since they assume full control over the training process as in centralized machine learning. Moreover, clients might be malicious; therefore, they should be excluded from any involvement in watermarking and should not have

access to the watermark set. O has two options for embedding the watermark using existing techniques:

- "Pre-embedding" into $w_{G(0)}$ before training starts.
- "Post-embedding" into $w_{G(t)}$ before deploying the model.

Pre-embedding achieves a watermark accuracy of 100% during the first aggregation round, while Post-embedding achieves that at the last aggregation round t. However, both techniques have limitations. Pre-embedded watermarks are easily removed from the global model after several aggregation rounds. Post-embedding demonstrates a reliable proof of ownership when the model is deployed, but adv can use updates from the global model, $w_{G(t-1)}$, one aggregation round before the training is completed. $w_{adv(t-1)} = w_{G(t-1)}$ contains no watermark and $Acc(w_{adv(t-1)}, D_{test}) \approx Acc(w_{adv(t)}, D_{test})$. Finally, post-embedded watermarks are not resilient to removal attacks such as fine-tuning and pruning as shown in [9]. **Both techniques fail to satisfy W1 and W2 and are not feasible for watermarking federated learning models.**

B. WAFFLE *Procedure*

Considering the capabilities of Agg presented in Section III-A, we add a new responsibility for it: *watermarking*. In WAFFLE, O shares its watermark set WM_{w_G} only with Agg. Agg re-trains w_G to guarantee that the watermark is embedded and VERIFY returns $True$ for watermarked models: $w^+_{G(t)}$ and $w^+_{c_i(t)}$. WAFFLE makes **no modification to client operations or the secure aggregation.** Inside Agg, we introduce two new functions: PRETRAIN and RETRAIN. PRETRAIN is a one time operation performed before federated learning starts and RETRAIN is a recurrent operation performed at each aggregation round t. The pseudo-code for PRETRAIN and RETRAIN is given in the extended version of this work [22].

在初始阶段和每轮fedavg之后重新基于waffle pattern训练模型

PRETRAIN gets a randomly initialized $w_{G(0)}$ and returns a watermarked version $w^+_{G(0)}$, $Acc(w^+_G(0), \mathrm{WM}_{w_G}) = 100\%$. RETRAIN first implements the secure aggregation: averaging re-trained $w^-_{c_i(t)}$[3] received from C_{sub} and updating $w^-_{G(t+1)}$ with this average. Then, it re-trains $w^-_{G(t+1)}$ using WM_{w_G} until $Acc(w^-_{G(t+1)}, \mathrm{WM}_{w_G})$ reaches threshold value th. We also define the maximum number of re-training rounds $E_r = 100$ in RETRAIN to not slow down model upload speed. Therefore, RETRAIN terminates when the number of re-training rounds reaches E_r, even if $Acc(w^+_{G(t+1)}, \mathrm{WM}_{w_G}) < th$.

WAFFLE starts with Agg executing PRETRAIN. Then Agg executes RETRAIN and sends $w^+_{G(t+1)}$ to C_{sub} in each t. By starting federated learning with a global model already converged to the watermark, we can decrease the number of re-training rounds required to reach high $Acc(w^+_{G(t)}, \mathrm{WM}_{w_G})$ satisfying **P3**. WAFFLE ensures high enough $Acc(w^+_{G(t)}, \mathrm{WM}_{w_G})$ to enable demonstration of ownership at anytime, satisfying **W1**. WAFFLE also ensures that $w^+_{G(t)}$ and $w^+_{adv(t)}$ converge for both the watermark and the actual task, satisfying **P1**. In addition, WAFFLE does not require any client's training data to operate, since WAFFLE re-trains $w^+_{G(t)}$ using only the watermark but no other data samples. Therefore, it satisfies the requirement **W3**.

WAFFLE is executed after the aggregation step of federated learning, and **is independent of the aggregation method**. Therefore, any other robust aggregation method such as Krum [23], trimmed mean or median [24] can be easily combined with WAFFLE. While WAFFLE can use any existing watermark set, we introduce WAFFLEPATTERN, a novel way to generate (training) data-independent watermarks.

C. WAFFLEPATTERN

在随机噪声的图片中添加特定的水印pattern

Inspired by the prior work [9], [13] and assuming that O has no access to the training data in large-scale federated learning applications, we propose WAFFLEPATTERN: adding specific patterns to images containing only noise. While creating WAFFLEPATTERN, we first generate different images using Gaussian noise. Then, each image is embedded with a certain pattern and labeled with a class that is related to the original task. Each class has a different pattern and every pattern is unique in terms of color, shape, orientation and position. We assume that O knows the dimensions of the model's inputs and outputs, since O selects the model architecture according to these specifications. Therefore, this information is enough to construct WAFFLEPATTERN. Figure 2b illustrates samples of WAFFLEPATTERN generated for colored images.

WAFFLEPATTERN is **independent of training data** and suitable for federated learning. This property satisfies the requirement **W3**. Additionally, WAFFLEPATTERN is **easy to learn**. Using the same pattern for each class helps w_G to converge and overfit to WAFFLEPATTERN, since samples of each class include features that are easy to learn and memorize. Therefore, this property helps satisfying **P2** and **P3**.

V. EXPERIMENTAL SETUP

A. Datasets and Models

Considering that reaching a high accuracy in federated learning is very challenging, we replicated image classification tasks in [1] to provide baseline models with a sufficiently high test accuracy. We choose MNIST [25] and CIFAR10 [26] as datasets since prior work [1], [21], [27] has obtained good models with these datasets using FedAvg. For MNIST, we use a 5-layer convolutional neural network [18]. For CIFAR10, we use VGG16 [28][4] which is an off-the-shelf complex DNN model trained over the ImageNet dataset [29]. For the federated learning setup, we use 100 clients participating the training and select 10 clients randomly as C_{sub} in each t. We repeat our experiments for different number of local passes $E_c = \{1, 5, 10, 20\}$ used by each client. We distribute the training data over clients in IID fashion. Each client receives 600 and 500 training images for MNIST and CIFAR10, respectively. We measure the test accuracy using the test subset of datasets, assuming a similar setup as in [2]: the model is evaluated at different client devices not involved

[3] w^- refers to a re-trained version of a previously watermarked model w^+.

[4] VGG16 has no batch normalization layer that uses training data statistics. Therefore, it is a suitable model for federated learning.

(a) Clean (b) WAFFLEPATTERN (c) Embedded C. (d) unRelate (e) unStruct

Fig. 2. Original images (a) in CIFAR10 dataset for classes airplane, automobile and bird as well as sample of watermark sets WAFFLEPATTERN (b), Embedded Content [13] (c), unRelate [9], [13] (d) and unStruct [14] (e) constructed for these three classes.

in the federated learning process. Details about the training configuration settings can be found in the extended version of this work [22].

We compare WAFFLE to other methods presented in IV-A and to baseline models that do not use a watermarking scheme. Baseline models are constructed using FedAvg and we use the tuple of $\{E_c, E_a\}$ to define different baseline models. We achieve 99% test accuracy on MNIST in all 4 baseline models. For CIFAR10, we achieve 85% test accuracy similar to [1].

B. Watermark Sets

For generating WAFFLEPATTERN we choose 100 for the size of the watermark set since it is sufficient to provide a high confidence $> 1-2^{-64}$ during demonstration of ownership [9], [11] while not degrading the overall performance of the model. In both MNIST and CIFAR10, each class contains 10 watermark samples. In order to construct WAFFLEPATTERN, we first generate 100 images with Gaussian noise, then randomly create 10 different patterns and label each pattern with a different class. Finally, each noisy image is combined with a random pattern. Figure 2b shows samples of WAFFLEPATTERN generated for the CIFAR10 task.

We also compare the performance of WAFFLEPATTERN to other state-of-the-art watermark generation methods [9], [13], [14]. Figure 2 shows samples of these watermarks generated for the CIFAR10 task. **Embedded Content** [13] takes a subset of the training data and modifies samples of this subset by adding a meaningful content (e.g., logo, text, a specifically designed pattern) into them. While original samples are labeled correctly, modified samples have incorrect labels that are pre-defined by the model owner. We construct Embedded Content by randomly selecting 10 images from the training data for each class, 100 images in total. We modify these images by adding a specifically designed pattern to them and assign incorrect labels to this modified set. We emphasize that Embedded Content requires training data knowledge, so it is not a suitable method for watermarking large-scale federated learning models. Nevertheless, we compare WAFFLEPATTERN to Embedded Content, since it is proved to be a robust watermark [13] against post-processing techniques. **unRelate** [9], [13] is either designed as unstructured abstract images [9] or as structured images from another data distribution [13], both of which are unrelated to the original task. These images are labeled with classes from the original task. For example, if the task is face recognition, the model owner might use different handwriting images to construct the watermark. We construct unRelate by randomly sampling 100 images from the ImageNet dataset which are unrelated to CIFAR10 and MNIST [13]. **unStruct** [14] generates a random watermark set and adds it into different layers of DNN models. In order to imitate unStruct, we produce watermark samples with purely Gaussian noise as in [30]. In both unRelate and unStruct, 10 classes are assigned to randomly selected 10 images, 100 in total.

VI. EVALUATION

We evaluate the performance of WAFFLE and WAFFLEPATTERN using the requirements defined in Section III-B. We compare the performance of models watermarked using WAFFLE to Pre-embedded models and baseline models without watermark (see Section V). We also analyze how well WAFFLEPATTERN performs by comparing it to Embedded Content [13], unRelate [9], [13] and unStruct [14] (see Figure 2).

A. *W1* Demonstration of Ownership

W1 requires VERIFY to return $True$ at any t. We explained in Section IV that WM_{w_G} is secret, only known by the model owner and Agg. It can be registered in a public bulletin as proposed in [9]. Thus, **W1** is satisfied if $Acc(w^+_{G(t)}, \mathrm{WM}_{w_G}) \geq T_{acc}$. Using the formula defined in [11], we compute that a reliable demonstration of ownership with confidence $> 1-2^{-64}$, given a watermark set of size 100 for a 10-classes classifier (MNIST and CIFAR10 models), is provided by $T_{acc} = 47\%$.

Table I shows that Pre-embedding achieves on average $Acc(w^+_{G(E_a)}, \mathrm{WM}_{w_G}) = 27\%$ for MNIST and 15% for CIFAR10. Watermarks embedded via Pre-embedding are not resilient to re-training. In contrast, WAFFLE achieves high watermark accuracy ($\sim 99.0\%$) at the last aggregation round in both tasks and VERIFY returns $True$ before w_G starts to improve (after 10 aggregation rounds on average). We can successfully embed all four types of watermark sets long before the global model converges. Therefore, in the remaining

TABLE I

AVERAGE WATERMARK ACCURACY (OVER 4 DIFFERENT WATERMARKS) AT THE FINAL AGGREGATION ROUND E_a. WAFFLE SATISFIES **W1** WHILE PRE-EMBEDDING DOES NOT: $Acc(w^{+}_{G(E_a)}, \mathrm{WM}_{w_G}) < 47\%$.

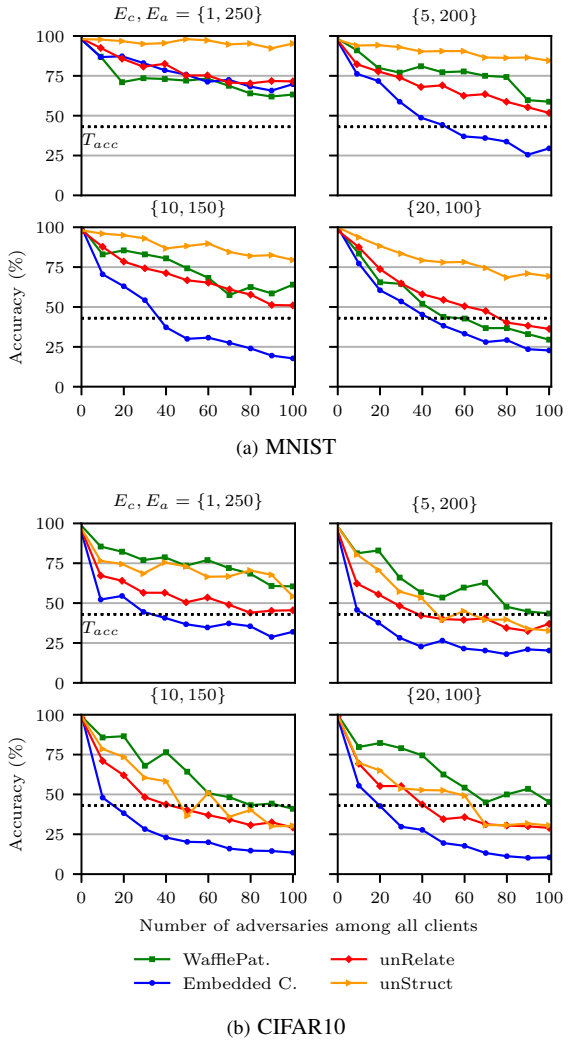
	$Acc(w^{+}_{G(E_a)}, \mathrm{WM}_{w_G})$ for **MNIST**		$Acc(w^{+}_{G(E_a)}, \mathrm{WM}_{w_G})$ for **CIFAR10**	
$\{E_c, E_a\}$	Pre-embedding	WAFFLE	Pre-embedding	WAFFLE
$\{1, 250\}$	24.00	99.00	15.00	99.00
$\{5, 200\}$	30.00	99.00	14.00	99.50
$\{10, 150\}$	22.75	98.50	15.00	99.00
$\{20, 100\}$	31.00	98.75	16.00	99.75

experiments, we only evaluate WAFFLE with different watermark sets.

B. *W2 Robustness*

W2 states that embedded watermarks should be robust against post-processing watermark removal techniques. For evaluating the robustness, we use three state-of-the-art defenses against backdooring. Fine-tuning [31] and pruning [32], [33] are generic watermark removal techniques that do not require knowledge of the trigger pattern or the watermarking method. Both techniques utilize some clean dataset (e.g., subset of a training set) and re-train the model in order to remove the watermark. Neural Cleanse [34] is a technique to detect, reverse-engineer and subsequently remove potential backdoors from DNN models, and it can be used for watermark removal. To evaluate these techniques, we set the acceptable utility drop of 5 percentage point (pp): the watermark must be removed while keeping the test accuracy degradation less than 5pp.

1) Fine-tuning attack: We tested fine-tuning attack using an increasing number of malicious clients combining their local datasets. Figure 3 shows that WAFFLEPATTERN is the most resilient watermark to fine-tuning for CIFAR10 in all experiments while it is the second most resilient watermark for MNIST when $\{E_c, E_a\} = \{5, 200\}$ and $\{10, 150\}$. Table II provides a detailed evaluation of resilience according to the number of fine-tuning epochs run by adv. Table II shows that



(a) MNIST

(b) CIFAR10

Fig. 3. Comparison of watermark accuracy for MNIST and CIFAR10 at different $\{E_c, E_a\}$ tuples when fine-tuning is implemented by an increasing number of malicious clients.

TABLE II

TEST AND WATERMARK (WM) ACCURACY (%) FOR DIFFERENT NUMBERS OF FINE-TUNING EPOCHS ON WATERMARKED MNIST AND CIFAR10 MODELS USING WAFFLEPATTERN. 1 ADVERSARY OUT OF 100 CLIENTS.

MNIST

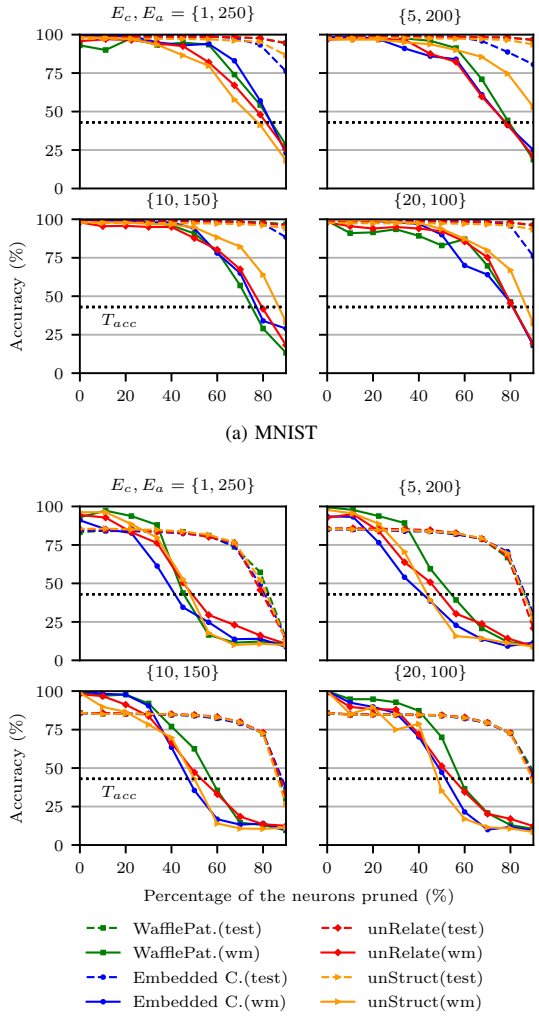
$\{E_c, E_a\}$	$\{1, 250\}$		$\{5, 200\}$		$\{10, 150\}$		$\{20, 100\}$	
epoch	Test	WM	Test	WM	Test	WM	Test	WM
0	98.8	99.0	99.0	99.0	98.9	100.0	98.8	99.0
20	98.6	97.8	98.9	98.8	98.7	98.8	98.5	88.2
40	98.6	97.2	98.9	98.5	98.7	98.8	98.5	88.0
60	98.6	95.5	98.9	98.2	98.7	98.2	98.5	88.0
80	98.6	95.2	98.9	98.2	98.7	98.0	98.5	88.0
100	98.6	95.0	98.9	98.0	98.7	98.0	98.5	88.0

CIFAR10

$\{E_c, E_a\}$	$\{1, 250\}$		$\{5, 200\}$		$\{10, 150\}$		$\{20, 100\}$	
epoch	Test	WM	Test	WM	Test	WM	Test	WM
0	85.7	100.0	85.6	100.0	85.8	99.0	85.6	100.0
20	85.4	94.2	85.5	99.8	85.5	96.8	84.4	95.2
40	85.4	94.0	85.5	99.8	85.5	96.2	84.5	94.8
60	85.4	93.0	85.5	99.8	85.6	96.0	84.6	95.0
80	85.4	93.0	85.5	99.8	85.6	96.0	84.6	95.0
100	85.4	93.0	85.5	99.8	85.6	96.2	84.6	94.8

WAFFLEPATTERN is resilient to fine-tuning even for a large number of fine-tuning epochs.

2) Pruning attack: We implemented parameter pruning proposed in [32] followed by fine-tuning by one adv using different pruning rates. We chose the pruning technique in [32] that first removes a number of connections with magnitudes close to zero from the dense model and obtains a more sparse model. Then, it re-trains this sparse model to recover the test accuracy. Figure 4 shows that WAFFLEPATTERN is the most resilient watermark for CIFAR10, since the watermark accuracy decreases below T_{acc} when a higher percentage of neurons removed from CIFAR10 models compared to other watermark types. Although it is not the most resilient in MNIST, Figure 4 shows that if the adversary removes more than 70% of the neurons from MNIST models (respectively 50% in CIFAR10 models), test accuracy starts to decline. Therefore, even though adv can evade demonstration of



(a) MNIST

(b) CIFAR10

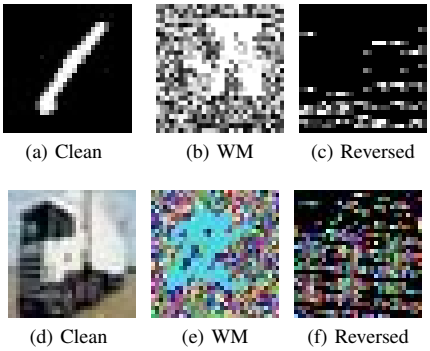
Fig. 4. Comparison of the test and watermark (wm) accuracy for MNIST and CIFAR10 at different $\{E_c, E_a\}$ tuples when the pruning attack is implemented by one adversary.

ownership, i.e., $\text{VERIFY}(w_{adv}, \text{WM}_G) \rightarrow False$, the pruned model shows a drop in test accuracy larger than 5pp compared to the original w_{adv}. We also measured that WAFFLEPATTERN satisfies **W2** if the proportion of malicious clients is smaller than 40%.

3) Neural Cleanse attack: Neural Cleanse first finds potential small trigger patterns needed to misclassify all inputs as a specific label and repeats this step for all classes. Then, it outputs an anomaly index based on this analysis. If the anomaly index is above 2.0, the model is considered backdoored. In the case of a suspected backdoor, Neural Cleanse also marks any class that is likely to be infected. We found that adversaries cannot detect the presence of watermarks in MNIST (Table III). On the other hand, adv can recognize that the model is watermarked in CIFAR10. However, it can only mark up to 3 possibly infected classes, while all ten classes are

TABLE III
AVERAGE ANOMALY INDEX FOR WATERMARKED MODELS WATERMARKED USING DIFFERENT WATERMARK SETS. RESULTS ARE AVERAGED OVER WATERMARKED MODELS WITH DIFFERENT $\{E_c, E_a\}$ TUPLES AND VARIOUS NUMBER OF ADVERSARIES.

Dataset	WAFFLEPATTERN	Embedded C.	unRelate	unStruct
MNIST	1.27	1.32	1.47	1.54
CIFAR10	2.35	2.16	2.32	2.15



(a) Clean (b) WM (c) Reversed

(d) Clean (e) WM (f) Reversed

Fig. 5. Visualization of reversed triggers where Neural Cleanse is implemented by one adversary against watermarked models using WAFFLEPATTERN. We show the original images (a) MNIST class 1 and (d) CIFAR10 class 9; example watermark samples for them (b) and (e) respectively; and reversed triggers via Neural Cleanse (c) and (f) respectively.

infected by our watermarks. Moreover, the reversed triggers of marked classes look completely different from the original patterns, as shown in Figure 5.

We also implemented patching-via-unlearning [34], a mitigation technique using reversed triggers obtained via Neural Cleanse. This method requires relaxing some assumptions of our adversary model and leaking information about the triggers to adv. We assumed that adv is aware of the presence of watermarks in w_{adv} and it knows that the watermark set includes different, distinctive patterns for each class. Performing this attack, a single adv cannot remove the watermark from MNIST model and at least 40% of clients should be malicious for a successful removal. For CIFAR10, one adv might evade the verification but the performance drop for watermarked models using WAFFLEPATTERN is over 5pp after patching as shown in Table IV. If less than 10% of clients are malicious, they cannot recover the performance of w_G and WAFFLEPATTERN satisfies **W2**.

C. *P1* Model Utility

According to **P1**, watermarking should not degrade the test accuracy of the converged model, i.e., $w^+_{G(t=E_a)}$ and $Acc(w^+_{G(t=E_a)}, D_{test}) \approx Acc(w_{adv(t=E_a)}, D_{test})$. For evaluating utility, we measured the test accuracy of watermarked federated learning models using four different watermark sets, and compared it to non-watermarked baseline models. Table V presents the test accuracy of baseline and watermarked models using different $\{E_c, E_a\}$ combinations. Results show that WAFFLEPATTERN reaches a sufficient test accuracy similar to baseline models ($\leq 0.2pp$ for MNIST and $\leq 0.7pp$ for

TABLE IV
ROBUSTNESS AGAINST NEURALCLEANSE PATCHING VIA UNLEARNING. TEST AND WATERMARK (WM) ACCURACY (%) FOR CIFAR10 MODELS WATERMARKED USING WAFFLEPATTERN.

$\{E_c, E_a\}$	$\{1, 250\}$		$\{5, 200\}$		$\{10, 150\}$		$\{20, 100\}$	
# of adv.	Test	WM	Test	WM	Test	WM	Test	WM
0 (Baseline)	85.7	100.0	85.6	100.0	85.8	99.0	85.6	100.0
1	74.0	27.2	75.9	25.2	75.6	28.0	76.2	35.0
2	78.8	29.5	73.4	31.8	72.6	26.0	74.4	26.5
5	78.1	29.8	80.5	26.0	78.5	31.2	78.8	24.8
10	80.2	27.5	80.8	34.5	80.9	20.0	78.4	38.8
20	79.3	36.5	81.6	30.2	81.5	25.2	80.8	36.5
30	80.6	17.8	82.4	26.5	81.0	34.8	82.0	37.5
40	83.3	30.8	81.8	27.8	81.1	40.2	81.5	30.0

TABLE V
THE TEST ACCURACY (%) (AT $t = E_a$) OF WATERMARKED MODELS USING DIFFERENT WATERMARK SETS.

$\{E_c, E_a\}$	Watermark pattern				
MNIST	Baseline	WAFFLEPATTERN	Embedded C.	unRelate	unStruct
$\{1, 250\}$	98.97	98.88	99.05	98.92	97.59
$\{5, 200\}$	98.91	98.94	98.98	98.79	98.13
$\{10, 150\}$	99.11	99.06	98.97	99.06	97.97
$\{20, 100\}$	99.02	98.85	98.97	98.79	97.77
CIFAR10	Baseline	WAFFLEPATTERN	Embedded C.	unRelate	unStruct
$\{1, 250\}$	86.27	85.70	85.19	85.81	86.53
$\{5, 200\}$	86.24	85.61	86.21	86.25	85.99
$\{10, 150\}$	85.90	85.89	85.69	85.76	85.91
$\{20, 100\}$	85.85	85.67	85.47	85.74	85.72

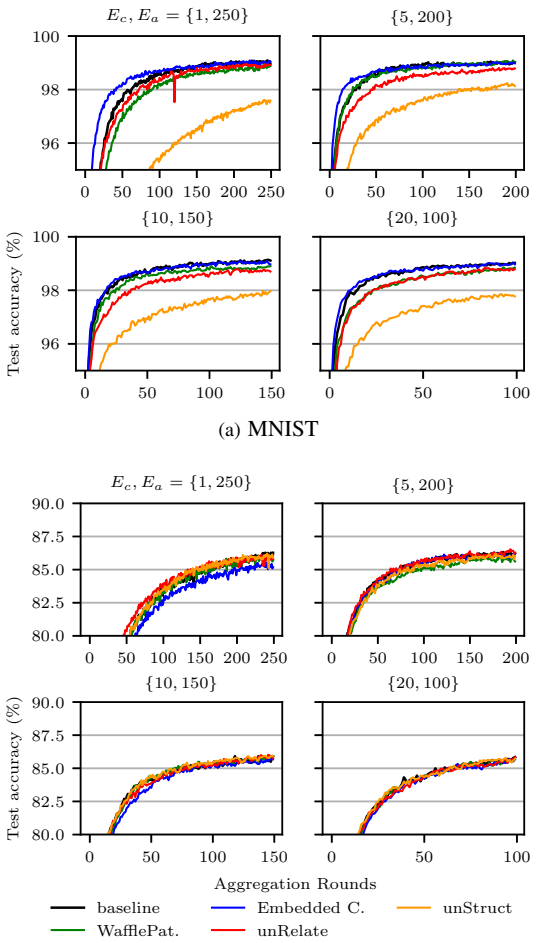
CIFAR10.) and does not degrade the test accuracy as much as unStruct in MNIST or Embedded Content in CIFAR10. WAFFLEPATTERN satisfies **P1**.

D. *P2-P3 Communication and computational overhead*

P2 and **P3** state that both the watermarking procedure and the watermark set should not increase the communication overhead and incur minimal additional computation. WAFFLE increases computation by re-training the global model in each t. Different watermark sets may be easier/harder to learn and incur additional communication (aggregation rounds) for the model to converge for both the watermark set and its main task: satisfying both **W1** and **P1**.

We evaluated the communication overhead by measuring the increase in test accuracy of watermarked models according to E_a. Reaching a high test accuracy in a smaller E_a means that the model can be trained with minimal communications. Figure 6 compares the test accuracy progression of non-watermarked baseline models and watermarked models using different watermark sets. In MNIST, the test accuracy quickly converges in baseline models and all watermarked models except for unStruct. In CIFAR10, Embedded Content converges slower than other watermark sets and requires more aggregation rounds for obtaining a performance similar to baseline models, except $\{E_c, E_a\} = \{1, 250\}$. In both cases, WAFFLEPATTERN satisfies **P2**.

We also calculated the computational overhead by dividing the total number of retraining rounds to the total number of local passes performed by clients. Table VI gives the computational overhead in WAFFLE using different watermarks. While unStruct has the lowest computational overhead, it requires twice as much aggregation rounds compared to other models to reach 99% test accuracy in MNIST. WAFFLEPATTERN usually needs fewer retraining rounds than other watermark sets, since it contains similar features that can be learned easily for each class. Therefore, WAFFLEPATTERN satisfies **P3**.



(a) MNIST

(b) CIFAR10

Fig. 6. Progression of the test accuracy for MNIST(a) and CIFAR10(b). Baseline models and watermarked models using different watermark sets are shown in the figure.

TABLE VI
AVERAGE COMPUTATIONAL OVERHEAD (%) INCURRED BY THE RETRAINING ROUNDS IN WATERMARKED MODELS USING DIFFERENT WATERMARK SETS.

Dataset	WAFFLEPATTERN	Embedded C.	unRelate	unStruct
MNIST	3.06	2.02	10.39	0.91
CIFAR10	2.97	5.72	6.10	1.47

E. *Evasion of Verification*

In addition to methods to recover and remove watermarks, *adv* can also attack the verification mechanisms [9], [11] used to demonstrate model ownership. Even though verification is not in the scope of our paper, we discuss possible attacks.

adv may try to *evade verification* by detecting queries for watermark samples as out-of-distribution (OOD) samples.

TABLE VII
EVASION OF VERIFICATION IN BOTH IID AND NON-IID SETTINGS. THE LOWEST FALSE POSITIVE RATE (FPR) AND TRUE POSITIVE RATE (TPR) CALCULATED AT THAT FPR IS REPORTED FOR WATERMARKED CIFAR10 MODELS USING WAFFLEPATTERN.

	IID setting		non-IID setting	
# of adv.	TPR	(lowest) FPR	TPR	(lowest) FPR
1	64.0	0.8	89.5	53.0
2	78.7	1.3	80.8	39.3
5	88.0	1.6	92.2	22.9
10	94.7	2.5	90.8	19.7
20	90.0	1.1	91.8	7.0
30	96.5	1.6	88.0	15.3
40	81.0	1.0	91.8	6.8
50	80.0	0.6	84.0	4.8

We implemented and tested this attack using the threshold-based detector model introduced in [16] for CIFAR10. This method is shown [16] to be strong enough to evade verification against backdoor-based watermarking methods [9], [10] with a negligible false positive rate (FPR). As a possible watermark set, we use a subset of the TinyImageNet[5], which is similar to unRelate. adv trains the detector with both its training data which represents in-distribution data, and TinyImageNet subset representing OOD data. We investigate two different scenarios: 1) Each client including adversaries, has a balanced, IID dataset as defined in the adversary model, and 2) a more realistic scenario where clients as well as adversaries have non-IID, unbalanced datasets [1]. In both scenarios, CIFAR10 models are watermarked with WAFFLEPATTERN where $\{E_{c_1}, E_{a_1}\} = \{1, 250\}$, $\{E_{c_2}, E_{a_2}\} = \{1, 450\}$ and w^2_{adv} has 81.9% test accuracy so the two models have similar performance. Table VII reports both the lowest FPR calculated over multiple thresholds and the true positive rate (TPR, the ratio of watermark samples correctly identified as OOD to the watermark set) at that FPR. As can be seen from the table, WAFFLE watermark verification could be evaded if adv has high quality IID data. However, WAFFLE is resilient to evasion in a non-IID setting. In the non-IID scenario, verification can be evaded with 5% FPR only if more than 50% clients are adversaries and share their datasets for training the detector. Moreover, since adv having only a limited training dataset might choose poor OOD data, this affects the detection performance and increases FPR to a degree such that the resulting prediction model is unusable [35]. Therefore, evasion of verification is not a concern for real-world client-server federated learning with non-IID data.

VII. DISCUSSION AND TAKEAWAYS

In contrast to existing Pre- and Post-embedding techniques, WAFFLE meets **W1** enabling to reliably demonstrate ownership of w_G at any aggregation round. Figure 7 summarizes the overall effectiveness of different watermark types, and it is evident that WAFFLEPATTERN gives the best trade-off (big area and never the worst one) considering all six requirements.

[5]https://tiny-imagenet.herokuapp.com

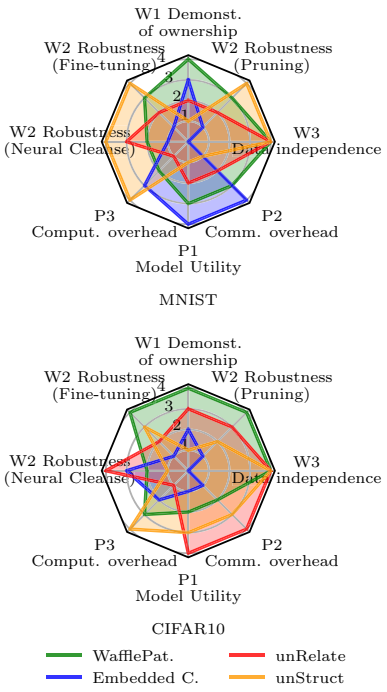


Fig. 7. Trade-off between meeting the requirements in Section III-B for different watermark sets. All watermarks are ranked between 1-4 (the higher, the better) based on the results obtained from all experiments.

WAFFLE is also resilient to almost all watermark removal attacks if less than 40% clients are malicious, as summarized in Table VIII. While Neural Cleanse requires only 10% malicious clients to be successful against the CIFAR10 model, adversaries would have to share their datasets and collaborate with each other to recover the triggers. The collaboration between adversaries in a distributed attack is different from a sybil attack in our case. Creating multiple identities does not improve removal and evasion attacks since the performance of these attacks is directly related to the quantity and the diversity of data held by the adversary. Collaboration is impractical since adversaries need to reveal their highly-sensitive dataset to other untrustworthy parties. We also empirically evaluated that watermark removal attacks fail to decrease watermark accuracy or recover test accuracy, if performed in a federated learning setup without sharing their datasets. On the other hand, if an adversary holds more than 10% of the whole training data, it could successfully remove watermarks from w_{adv} without decreasing the model performance. This scenario is not typical in large-scale distributed learning settings, where there are a very large number of clients, each holding a tiny fraction of the overall training data. We conclude that under reasonable assumptions, WAFFLE is resilient to distributed attacks of several malicious federated learning clients.

Adversaries can also try to cast model ownership into doubt by embedding its own watermark into the model: *ownership piracy*. This issue can be addressed by registering any wa-

TABLE VIII
MINIMUM RATIO OF MALICIOUS CLIENTS REQUIRED TO EVADE WAFFLE.

Attack type	MNIST	CIFAR10
Fine-tuning [31]	50%	50%
Pruning [32]	40%	40%
Neural Cleanse, patching-via-unlearning [34]	40%	10%
Evasion of verification with OOD detection [16]	-	50%

termark into a time-stamped bulletin (e.g., blockchain) for it to be valid [9], [11]. We can enforce our watermark to be registered together with some private artifacts of the model before federated learning starts as a proof of authenticity. Such a private artifact can be the architecture of the model (number of layers, number of neurons, type of activation functions, etc.) since all these parameters can provide a large entropy and they are unknown to any federated learning client before they receive the initial model.

Finally, a model stealing attack can be performed against w_{adv} by adv itself to create a surrogate model without watermark or embed its own watermark [11], [36]. Model stealing typically requires a large amount of data (more than specified by our adversary model) and it causes accuracy drops larger than 5pp [37]. Thus, we consider these attacks impractical in our adversary model.

We assume that each client's training data D_{c_i} is IID (Section III-A), and based this assumption on state-of-the-art papers [1], [23], [38]. Although IID assumption leads to a better setting to study the impact of model utility requirement **P1**, in real world federated learning applications, D_{c_i}'s are typically unbalanced and non-IID. Therefore, we evaluated the performance of WAFFLE and WAFFLEPATTERN in non-IID scenarios, and concluded that we met requirements **W1** and **P1** with a slight increase in communication and computational overhead. Detailed experimental results can be found in the extended version of this work [22].

While we focused on a single owner O, WAFFLE can also be used if several clients would all be owners of the trained model. In this case, Agg can generate and distribute different subsets of the watermark set to each client for individual demonstration of ownership. There are challenges to be addressed in the case of collective ownership since the size of the watermark set increases linearly with the number of owners and there might be a decrease in utility. We consider extending WAFFLE to this case as future work. We will also explore how malicious clients can try to recover and degrade watermarks *during* the training phase of the federated learning and provide detection/mitigation techniques against these adversaries.

VIII. RELATED WORK

Watermarking DNNs for ownership verification was first proposed in [12] by using backdooring techniques. Deepmarks [39] presents a collusion-secure watermarking method that encodes watermarks into the probability density function of weights using a specific regularization loss during the training phase. However, these techniques require direct access to model weights (white-box access) for ownership verification. DeepSigns [14] is the first watermarking method applicable with both white-box and black-box access, and it embeds watermarks to the activation maps of selected layers. DeepIPR [40] proposes a passport-based DNN ownership verification scheme that tries to embed watermarks into a special passport layer of DNNs. Although DeepIPR is robust against watermark removal attacks, it is expensive and imposes a significant computational cost. A zero-bit watermarking algorithm [41] embeds watermarks into models, that are stolen via black-box methods, by leveraging adversarial perturbations. Although this approach is feasible, it heavily depends on adversarial examples and their transferability property across different models. Authors in [16] propose a blind-watermark based approach that inserts a specifically designed logo into the original sample via encoder such that the resulting sample is almost indistinguishable from the original one. All these proposals require full control of the training process and cannot be applied in federated learning.

Federated learning is vulnerable to adversarial attacks that alters the training and inference phase of the system. Poisoning attacks are first introduced in [21], where a malicious client trains its local model on the backdoor and attempts to replace the global model with the poisoned model. Another powerful attack is model update poisoning attacks, where the adversary aims to prevent the global model from converging to a desirable state by sending poisoned model updates to the server [38]. [42] states that clients might suffer Byzantine failures, which leads to arbitrary behavior across communication rounds affecting the convergence of the global model.

Client-server federated learning also suffers from privacy leakage [43]–[45]. For example, in [43], a malicious client can learn about class representatives of other clients' training data by using generative adversarial networks. Attackers in client-server federated learning implement passive [45] and active [44] membership inference attacks in order to detect whether a sample belongs to the overall training set or a specific client. In passive attacks, the attacker could be the aggregator that can only observe individual model updates. In active attacks, malicious clients try to influence the global model in order to extract more information about other clients' training dataset. There have been a few attempts to prevent the privacy leakage applying differential privacy [46] or using trusted execution environments in clients' devices [47]. However, these methods trade-off either computational overhead or performance for privacy.

REFERENCES

[1] H. B. McMahan, E. Moore, D. Ramage, S. Hampson *et al.*, "Communication-efficient learning of deep networks from decentralized data," *arXiv preprint arXiv:1602.05629*, 2016.

[2] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," *arXiv preprint arXiv:1811.03604*, 2018.

[3] T. Yang, G. Andrew, H. Eichner, H. Sun, W. Li, N. Kong, D. Ramage, and F. Beaufays, "Applied federated learning: Improving google keyboard query suggestions," *arXiv preprint arXiv:1812.02903*, 2018.

[4] F. Granqvist, M. Seigel, R. van Dalen, Áine Cahill, S. Shum, and M. Paulik, "Improving on-device speaker verification using federated learning with privacy," 2020. [Online]. Available: https://arxiv.org/pdf/2008.02651.pdf

[5] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konecny, S. Mazzocchi, H. B. McMahan *et al.*, "Towards federated learning at scale: System design," *arXiv preprint arXiv:1902.01046*, 2019.

[6] K. Li, "Reverse engineering AI models," 2018, HITBSecConf. [Online]. Available: https://conference.hitb.org/hitbsecconf2018dxb/sessions/ai-model-security-reverse-engineering-machine-learning-models/

[7] Z. Sun, R. Sun, and L. Lu, "Mind your weight (s): A large-scale study on insufficient machine learning model protection in mobile apps," *arXiv preprint arXiv:2002.07687*, 2020.

[8] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *arXiv preprint arXiv:1912.04977*, 2019.

[9] 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 Security Symposium (Security 18)*, 2018, pp. 1615–1631.

[10] J. Guo and M. Potkonjak, "Watermarking deep neural networks for embedded systems," in *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2018, pp. 1–8.

[11] S. Szyller, B. G. Atli, S. Marchal, and N. Asokan, "DAWN: dynamic adversarial watermarking of neural networks," *arXiv preprint arXiv:1906.00830*, 2019.

[12] 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.

[13] 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*, 2018, pp. 159–172.

[14] 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.

[15] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1175–1191.

[16] 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.

[17] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang, and X. Zhang, "Trojaning attack on neural networks." in *ISOC Network and Distributed System Security Symposium*, 2018.

[18] M. Juuti, S. Szyller, S. Marchal, and N. Asokan, "PRADA: protecting against dnn model stealing attacks," in *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2019, pp. 512–527.

[19] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction apis," in *25th {USENIX} Security Symposium ({USENIX} Security 16)*, 2016, pp. 601–618.

[20] WeBank AI Group, "Federated learning white paper v1.0," https://www.fedai.org/static/flwp-en.pdf, 2018, online; accessed 29 January 2020.

[21] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," *arXiv preprint arXiv:1807.00459*, 2018.

[22] B. G. Atli, Y. Xia, S. Marchal, and N. Asokan, "WAFFLE: watermarking in federated learning," *arXiv preprint arXiv:2008.07298*, 2020.

[23] P. Blanchard, R. Guerraoui, J. Stainer *et al.*, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Advances in Neural Information Processing Systems*, 2017, pp. 119–129.

[24] D. Yin, Y. Chen, K. Ramchandran, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," *arXiv preprint arXiv:1803.01498*, 2018.

[25] Y. LeCun, C. Cortes, and C. Burges, "MNIST handwritten digit database," *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, vol. 2, 2010.

[26] A. Krizhevsky, V. Nair, and G. Hinton, "CIFAR-10 (Canadian Institute for Advanced Research)." [Online]. Available: http://www.cs.toronto.edu/~kriz/cifar.html

[27] D. Li and J. Wang, "Fedmd: Heterogenous federated learning via model distillation," *arXiv preprint arXiv:1910.03581*, 2019.

[28] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[29] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[30] R. Namba and J. Sakuma, "Robust watermarking of neural network with exponential weighting," in *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*, 2019, pp. 228–240.

[31] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Advances in neural information processing systems*, 2014, pp. 3320–3328.

[32] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Advances in neural information processing systems*, 2015, pp. 1135–1143.

[33] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[34] 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)*. IEEE, 2019, pp. 707–723.

[35] L. Song, V. Sehwag, A. N. Bhagoji, and P. Mittal, "A critical evaluation of open-world machine learning," *arXiv preprint arXiv:2007.04391*, 2020.

[36] M. Shafieinejad, J. Wang, N. Lukas, and F. Kerschbaum, "On the robustness of the backdoor-based watermarking in deep neural networks," *arXiv preprint arXiv:1906.07745*, 2019.

[37] B. G. Atli, S. Szyller, M. Juuti, S. Marchal, and N. Asokan, "Extraction of complex DNN models: Real threat or boogeyman?" *arXiv preprint arXiv:1910.05429*, 2019.

[38] E. M. E. Mhamdi, R. Guerraoui, and S. Rouault, "The hidden vulnerability of distributed learning in byzantium," in *Proceedings of the 35th International Conference on Machine Learning*, 2018, pp. 3518–3527.

[39] H. Chen, B. D. Rouhani, C. Fu, J. Zhao, and F. Koushanfar, "DeepMarks: a secure fingerprinting framework for digital rights management of deep learning models," in *Proceedings of the 2019 on International Conference on Multimedia Retrieval*. ACM, 2019, pp. 105–113.

[40] L. Fan, K. W. Ng, and C. S. Chan, "Rethinking deep neural network ownership verification: Embedding passports to defeat ambiguity attacks," in *Advances in Neural Information Processing Systems*, 2019, pp. 4716–4725.

[41] E. L. Merrer, P. Perez, and G. Trédan, "Adversarial frontier stitching for remote neural network watermarking," *arXiv preprint arXiv:1711.01894*, 2017.

[42] Y. Chen, L. Su, and J. Xu, "Distributed statistical machine learning in adversarial settings: Byzantine gradient descent," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 1, no. 2, pp. 1–25, 2017.

[43] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the GAN: information leakage from collaborative deep learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 603–618.

[44] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 691–706.

[45] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: Stand-alone and federated learning under passive and active white-box inference attacks," *arXiv preprint arXiv:1812.00910*, 2018.

[46] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," *arXiv preprint arXiv:1712.07557*, 2017.

[47] F. Mo, A. S. Shamsabadi, K. Katevas, S. Demetriou, I. Leontiadis, A. Cavallaro, and H. Haddadi, "DarkneTZ: towards model privacy at the edge using trusted execution environments," *arXiv preprint arXiv:2004.05703*, 2020.