This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3196790

Evaluating the Robustness of Trigger Set-Based Watermarks Embedded in Deep Neural Networks

Suyoung Lee, Wonho Song, Suman Jana, *Member, IEEE,*
Meeyoung Cha, *Member, IEEE,* Sooel Son, *Member, IEEE*

Abstract—Trigger set-based watermarking schemes have gained emerging attention as they provide a means to prove ownership for deep neural network model owners. In this paper, we argue that state-of-the-art trigger set-based watermarking algorithms do not achieve their designed goal of proving ownership. We posit that this impaired capability stems from two common experimental flaws that the existing research practice has committed when evaluating the robustness of watermarking algorithms: (1) incomplete adversarial evaluation and (2) overlooked adaptive attacks. We conduct a comprehensive adversarial evaluation of 11 representative watermarking schemes against six of the existing attacks and demonstrate that each of these watermarking schemes lacks robustness against at least two non-adaptive attacks. We also propose novel adaptive attacks that harness the adversary's knowledge of the underlying watermarking algorithm of a target model. We demonstrate that the proposed attacks effectively break all of the 11 watermarking schemes, consequently allowing adversaries to obscure the ownership of any watermarked model. We encourage follow-up studies to consider our guidelines when evaluating the robustness of their watermarking schemes via conducting comprehensive adversarial evaluation that includes our adaptive attacks to demonstrate a meaningful upper bound of watermark robustness.

Index Terms—deep neural networks, watermark removal attacks, backdoor attacks, watermark robustness, trigger set-based watermarks

1 Introduction

The recent advent of deep neural networks (DNNs) has accelerated the development and application of diverse DNN models across various domains, including image search [19], [52], security [55], [56], and self-driving vehicles [49]. As machine learning technology evolves, the structures of state-of-the-art DNN models have become more complicated. This trend renders corporations with fewer computational resources unable to train state-of-the-art DNN models from scratch. For instance, the ImageNet [42] dataset holds 14M images; training a high-performing DNN model such as a ResNet-50 [22], which consists of over 25M parameters, takes up to several weeks with a machine equipped a Tesla M40 GPU. Moreover, it is difficult to obtain a large number of high-quality training instances pertaining to privacy-sensitive information, thus rendering it infeasible for corporations with limited data access to produce a superb model.

An adversary may attempt to steal such a superb model and host another service that imitates the service provided by the original model. This adversary poses a grave threat to the model owner, who has invested resources and time to develop a high-performing model. DNN model theft thus infringes on the intellectual property (IP) of the model owner and discloses the owner's business secrets. Accordingly, corporations seek a mechanism that proves the ownership of their DNN models to protect their IPs and business secrets.

Previous studies have proposed novel methods that validate ownership of a given DNN model, thus protecting the owner's IP. Similar to watermarking algorithms devised to protect the IP of multimedia content, such as images and videos [28], [46], previous studies have proposed new ways of embedding watermarks into a given DNN model as well as algorithms that verify ownership [2], [10], [16], [21], [24], [31], [35], [36], [41], [51], [58], [59]. The proposed watermarking algorithms are categorized into two types based on their methods of embedding watermarks: feature-based and trigger set-based methods.

Feature-based schemes [10], [41], [51] require white-box access to a model's internal weight parameters. On the other hand, trigger set-based watermarking methods [2], [16], [21], [24], [31], [35], [36], [58], [59] have gained attention due to their comparative merits of requiring black-box access for ownership verification. Trigger set-based schemes harness the common query interface of a suspect model. Specifically, these watermarking methods leverage carefully created images, called *key images*. A model owner assigns an arbitrary label, called a *target label*, to the key images and generates a *trigger set* that consists of an arbitrary number of key image and target label pairs. The owner then trains a model on this trigger set as well as on the normal training data. When verifying ownership, the model owner queries the model in doubt with the key images and checks whether the model returns the target label; this enables the owner to verify the ownership by using remote queries. Previous trigger

- *Suyoung Lee, Wonho Song, Meeyoung Cha, and Sooel Son are with School of Computing, KAIST, Daejeon 34141, South Korea.*
- *Suman Jana is with Department of Computer Science, Columbia University, New York, NY 10027, USA.*

set-based methods [2], [16], [21], [31], [35], [36], [58], [59] have in common that they use key images and target labels but differ in how they generate key images or select target labels.

Contributions. In this paper, we argue that trigger set-based watermarking methods today [2], [16], [21], [24], [31], [35], [36], [58], [59] do not achieve their goal of enabling model owners to prove their ownership of watermarked DNN models. We believe that previous studies have not evaluated the robustness of their watermarking algorithms to the fullest extent, thereby failing to demonstrate their readiness for real-world deployment.

There exist two different strategies for evaluating the robustness of a DNN model: (1) proving a theoretical lower bound with approximation [3], [23] and (2) demonstrating an upper bound via adversarial evaluation with strong attacks. Previous watermarking studies [2], [16], [21], [24], [31], [35], [36], [58], [59] have taken the latter approach, demonstrating their robustness against selected attacks. However, we observed two common flaws in previous studies when evaluating the robustness of their trigger set-based watermarking algorithms: (1) performing incomplete adversarial evaluation and (2) overlooking an adaptive adversary.

Incomplete adversarial evaluation. Because various attacks have been introduced across diverse studies in various contexts, we first consolidate and reorganize six existing attacks. We then categorize the existing attacks into two types, each of which corresponds to either of the adversary's two strategies: (1) claiming ownership by the adversary or (2) obscuring the owner's ownership.

We observed that no previous watermarking studies have considered the complete set of the existing strong attacks in their adversarial evaluation; previous studies have not demonstrated their robustness against at least one critical attack. Furthermore, we contend that adversarial evaluation of attacks employing the adversary's first strategy (claiming ownership by the adversary) is unnecessary. This strategy permits for a target model to contain watermarks from its original owner as well as the adversary, which demands additional proof to prove the adversary's ownership (§5.3.1). Therefore, there is no motive for the adversary to employ this strategy alone unless she combines the two aforementioned strategies by obscuring the original watermarks and then injecting her own watermarks.

To this end, we perform our own adversarial evaluation against 11 of the representative trigger set-based watermarking schemes while taking into account the aforementioned problems. We demonstrate that they are weak against at least two of these attacks. In particular, all of the 11 evaluated schemes were vulnerable to evasion [31], [34], [36] and ownership piracy attacks [2].

Overlooked adaptive adversary. Previous studies focused on evaluating their watermark robustness against selected existing attacks. Meanwhile, a vast volume of recent research on establishing the robustness of DNN models has considered adaptive adversaries [4], [6], [13], [25], [40], [45].

To this end, we propose three novel attacks that a strong adaptive adversary is able to conduct. Under the assumption that this adversary knows the underlying watermarking algorithm of a target model, we demonstrate that the proposed adaptive attacks effectively break all existing watermarking schemes, enabling the adversary to obscure the ownership of a target model, regardless of its underlying watermarking scheme. Therefore, our proposed attacks contribute to demonstrating a new upper bound of watermark robustness.

Overall, our experimental results demonstrate that trigger set-based watermarking schemes today are far from ready for real-world deployment. We recommend that future research evaluate their watermarking methods against at least all existing strong attacks, including our adaptive attacks, and consider our guidelines when demonstrating their watermark robustness via adversarial evaluation (§8).

To enable follow-on research to evaluate its watermarking schemes, all of our attack algorithms and their implementation will be available at https://github.com/WSP-LAB/wm-eval-zoo.

2 Background

2.1 DNN Ownership Verification

Since Uchida *et al.* [51] proposed the first approach to embedding watermarks into neural networks, various watermarking techniques have been proposed. In terms of their watermark embedding methodology, these watermarking techniques have been categorized into two types: trigger set-based and feature-based methods. Trigger set-based methods utilize additional training samples as watermarks for DNNs [2], [16], [21], [24], [31], [35], [36], [58], [59]. Feature-based methods embed watermarks by modifying model features [10], [41], [51].

Zhang *et al.* [58] proposed a representative trigger set-based method. They trained a model to learn predefined key pairs, each consisting of a key image and its target label. Specifically, they assigned a *false* label with respect to the ground-truth function to the key image. The gist of their approach is that a model without the watermark is highly likely to emit a ground-truth label rather than the predefined false label for a given key image. Therefore, the owner can prove the ownership afterward by querying the model with the key images and checking whether the model outputs the predefined false label. In this scheme, the key images and their predefined false labels become a *trigger set*.

Other trigger set-based watermarking techniques employ more or less similar approaches, but Adi *et al.* [2] further integrated this scheme with cryptographic primitives to secure embedded watermarks. Recently, Jia *et al.* [24] proposed to train a model in the direction of tightly coupling the trigger set with a regular training set so that the trained model becomes robust against model stealing attacks.

2.2 Target Watermark Schemes

Our goal is to evaluate the robustness of state-of-the-art trigger set-based watermark schemes. Thus, we chose 11 representative watermarking algorithms, published at top venues over the past five years [2], [16], [21], [24], [31], [35], [36], [41], [58]. They share a common scheme that uses trigger sets for verifying ownership.

Algorithm 1 summarizes how a trigger set-based watermark algorithm embeds the ownership proof of an owner

Algorithm 1: Embedding a trigger set into a DNN.

```
Input : A regular training set (D_train).
        A set of source images (I_src).
Output: A watermarked model (M_wm).
1 function EmbedWatermark(D_train, I_src)
2     I_key ← GenerateKeyImgs(I_src)
3     L_target ← AssignTargetLabels(I_key)
4     D_trigger ← (I_key, L_target)
5     M_wm ← TrainModel(D_train, D_trigger)
6     return M_wm
```

O into a DNN model. O provides a training set $\mathcal{D}_{train}$ and a set of source images $\mathcal{I}_{src}$ to the `EmbedWatermark` function. Given $\mathcal{I}_{src}$, `GenerateKeyImgs` generates a set of key images $\mathcal{I}_{key}$ (Line 2). Note that these key images are intentionally designed to have a different underlying distribution than that of $\mathcal{D}_{train}$. Owing to the over-parameterization of DNN models, they are capable of intentionally learning key images along with $\mathcal{D}_{train}$ [14], [57]. The `AssignKeyLabels` function assigns a target label $\mathcal{L}_{target}$ to each key image (Line 3). We call generated key images together with their assigned target labels as a *trigger set* $\mathcal{D}_{trigger}$. Finally, the `TrainModel` function trains a model with both $\mathcal{D}_{train}$ and $\mathcal{D}_{trigger}$ to embed watermarks (Line 5). This step is analogous to backdoor attacks [11], [20] *per se* but different in that this step is used to claim ownership of a DNN model, instead of emplacing backdoors.

When O claims her ownership, she conducts the following verification phase: O queries a model in doubt with the key images. If the model is indeed the owner's genuine model, the model will output the predefined target labels trained in the training phase. In Supplemental Material 1, we describe each of the 11 selected watermarking algorithms. Throughout the paper, we denote each algorithm as follows: $WM_{content}$ [58], WM_{noise} [58], $WM_{unrelated}$ [58], WM_{mark} [21], $WM_{abstract}$ [2], WM_{adv} [35], $WM_{passport}$ [16], $WM_{encoder}$ [31], WM_{exp} [36], DeepSigns [41], and $WM_{entangled}$ [24].

3 Adversary Model

We introduce an attack scenario in which an adversary infringes on the IP of a model owner with an exfiltrated DNN model, along with the notations that we use throughout the paper. We then describe the prior knowledge of an adversary regarding the exfiltrated model.

3.1 Attack Scenario

We assume two parties in the attack scenario: a model owner $\mathcal{O}$ and an adversary $\mathcal{A}$. $\mathcal{O}$ embeds watermarks into a neural network model M_{org} by training M_{org} with a trigger set, thus producing the watermarked model M_{wm}. $\mathcal{O}$ then hosts a service by leveraging M_{wm}. On the other hand, $\mathcal{A}$ decides to steal M_{wm} because training a precise model from scratch requires a lot of computational resources as well as training instances. For instance, $\mathcal{A}$ can steal M_{wm} by compromising $\mathcal{O}$'s machine learning service server or getting help from an insider. Enumerating the feasible ways of $\mathcal{A}$ obtaining M_{wm} is beyond the scope of this paper.

After stealing M_{wm}, $\mathcal{A}$ hosts a similar service as $\mathcal{O}$ using a model M_{adv} derived from M_{wm}. Note that the end goal of $\mathcal{A}$ is to either (1) obscure $\mathcal{O}$'s ownership of M_{adv} or (2) claim the ownership of M_{adv}. Therefore, $\mathcal{A}$ may have built M_{adv} by transforming M_{wm} to achieve these goals. That is, M_{wm} and M_{adv} are not necessarily the same. We further elaborate on attack scenarios with these goals in §5.1.

Finally, once $\mathcal{O}$ suspects that M_{adv} is derived from M_{wm}, $\mathcal{O}$ will attempt to prove their ownership of M_{adv}. However, if $\mathcal{O}$ watermarked M_{wm} with a feature-based scheme, $\mathcal{O}$ must have white-box access to M_{adv} to verify the ownership. Considering that $\mathcal{A}$ certainly wants to hide the true ownership of M_{adv}, $\mathcal{A}$ will not provide white-box access to M_{adv} unless M_{adv} is under litigation. Thus, in this paper, we focus on trigger set-based watermark schemes, which only require black-box access for ownership verification.

3.2 Adversarial Knowledge

We assume two adversaries according to their adversarial knowledge: (1) a non-adaptive adversary and (2) an adaptive adversary. A non-adaptive adversary knows that the stolen target model M_{wm} has been watermarked but does not know which specific watermarking algorithm was used. On the other hand, an adaptive adversary knows the exact watermarking algorithm that $\mathcal{O}$ harnessed to protect the model among various trigger set-based methods. Specifically, the adaptive adversary only knows the internal working of `GenerateKeyImgs` in Algorithm 1. She does not know the source images ($\mathcal{I}_{src}$) for `GenerateKeyImgs`. She also has no access to the original trigger set ($\mathcal{D}_{trigger}$) as well as the training dataset ($\mathcal{D}_{train}$).

Note that both adversaries share the same knowledge except about the watermarking algorithm. As both adversaries stole M_{wm} from $\mathcal{O}$, they can observe the model inputs, outputs, and structure. Additionally, we assume that they have access to 50% of a testing set, which is required to launch attacks against M_{wm}. Note that this data accessible by the adversaries is completely disjointed from the original training set, assuming the least privilege granted to them. Previous studies [2], [31], [41], [58] assume similar capabilities for the adversary to conduct different attacks. We further considered adversaries who have access to fewer data in Supplemental Material 4.

4 Motivation

We argue that today's evaluation practice of demonstrating watermark robustness exhibits two common shortcomings: incomplete adversarial evaluation (§4.1) and overlooked adaptive attacks (§4.2).

4.1 Incomplete Adversarial Evaluation

We observe that previous studies on trigger set-based watermarks have evaluated the robustness of their methods using arbitrary choices of the existing attacks, thus demonstrating an upper bound on their robustness only to the selected attacks. Due to the nature of adversarial evaluation, the existence of one effective attack denotes the failure to protect the IP of $\mathcal{O}$, effectively breaking a target watermarking scheme. Therefore, it is paramount to account for all existing attacks to demonstrate meaningful robustness.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3196790

TABLE 1: Summary of adversarial evaluations performed by previous studies.

Attack	$WM_{content}$	WM_{noise}	$WM_{unrelated}$	WM_{mark}	$WM_{abstract}$	WM_{adv}	$WM_{passport}$	$WM_{encoder}$	WM_{exp}	DeepSigns	$WM_{entangled}$
Fine-tuning	✓	✓	✓	✗	✓	✗	✓	✓	✗	✓	✓
Model Stealing	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓
Parameter Pruning	✓	✓	✓	✗	✗	✓	✓	✗	✓	✓	✓
Evasion	✗	✗	✗	✗	✗	✗	✗	✓	✓	✗	✓
Ownership Piracy	✗	✗	✗	✗	✓	✓	✗	✗	✗	✓	✓
Ambiguity	✓	✓	✓	✗	✗	✗	✓	✗	✗	✗	✗
# of Evaluated Attacks	3	3	3	0	2	2	3	2	2	3	5

Table 1 summarizes the evaluations performed by the previous watermark research in terms of applicable existing attacks. Note from the table that no previous studies evaluated their approaches against a complete set of attacks. Among the six attacks, 10 out of 11 prior watermark studies only considered at most three attacks and missed other attacks in their evaluations. Moreover, model stealing attacks have never been evaluated in any previous studies.

We emphasize that all six attacks examined herein have existed since each watermarking algorithm was first proposed. In other words, ever since each watermarking algorithm was first proposed, their robustness against several existing state-of-the-art attacks has remained unexplored. Therefore, it is still questionable whether state-of-the-art watermarking algorithms can successfully work as a defense mechanism against various real-world threats.

Furthermore, incomplete adversarial evaluation becomes problematic when comparing the robustness of different watermarking algorithms. Because the previous studies evaluated watermarking algorithms against arbitrarily chosen attacks, they have failed to demonstrate which algorithms are more robust than others in general. Even though one algorithm is robust against a given attack, it can be broken by another attack against which other algorithms are known to be secure. We believe that this incomplete evaluation practice stems from the lack of prior systematic studies that enumerate all the applicable attacks. Thus, in this paper, we summarize these attacks (§5.1).

4.2 Overlooked Adaptive Attacks

A vast volume of recent research on securing machine learning models has striven to demonstrate a meaningful upper bound of its robustness [4], [6], [13], [25], [40], [45]. To this end, they have focused on strong adaptive adversaries who know the adopted defense algorithms for securing the model. Nevertheless, the previous studies on watermarking algorithms have not yet taken into account adaptive attacks in their adversarial evaluation. Therefore, to challenge the robustness of watermarking algorithms to the fullest extent, we propose novel adaptive attacks in the context of DNN watermarking.

Note that the existing attacks in Table 1 are non-adaptive attacks. In addition to these attacks, we consider adaptive attacks against M_{wm}. The adaptive adversary mounts the same attacks as non-adaptive adversaries. She leverages her prior knowledge of the underlying watermarking algorithm and adapts these attacks, thus mounting strong attacks.

5 Attack Algorithms

We now introduce state-of-the-art attacks that non-adaptive and adaptive adversaries (§3) can conduct. We consolidate six of the existing attacks spread across various studies in the literature and systematically categorize them from the perspective of the goal that the adversaries aim to achieve (§5.1). We then briefly describe each existing attack (§5.2–§5.3). Finally, we present novel attacks that the adaptive adversary is able to conduct via leveraging the knowledge of a target watermarking algorithm (§5.4).

5.1 Attack Overview

An adversary $\mathcal{A}$ can devise two different scenarios to conceal the fact that $\mathcal{A}$ stole M_{wm} from $\mathcal{O}$; $\mathcal{A}$ can decide to either obscure $\mathcal{O}$'s ownership or claim her ownership.

Obscuring $\mathcal{O}$'s ownership. The goal of $\mathcal{A}$ in this scenario is to thwart $\mathcal{O}$'s ownership verification by modifying M_{wm}, such as by training a counterfeit model or detecting key images. As $\mathcal{O}$ fails to verify their ownership in this scenario, $\mathcal{A}$ can successfully obscure $\mathcal{O}$'s ownership and insist that M_{wm} is not watermarked. To achieve this goal, $\mathcal{A}$ can launch fine-tuning, model stealing, evasion, or parameter pruning attacks.

Claiming ownership by $\mathcal{A}$. Another scenario that $\mathcal{A}$ can consider is to claim the ownership of M_{wm} by implanting a new trigger set into M_{wm} or generating a set of fake key images that can trigger the target labels. Note that $\mathcal{A}$ does not aim to damage $\mathcal{O}$'s ownership and $\mathcal{O}$'s watermark may persist. Therefore, both $\mathcal{O}$ and $\mathcal{A}$ can claim the ownership based on the respective trigger set, which results in conflicting ownership arguments. Since it is infeasible to decide which one is fraudulently claiming ownership solely based on their key images and target labels, previous studies [2], [16], [24], [35], [41], [58] have considered this to be a plausible strategy. To realize this scenario, $\mathcal{A}$ is able to conduct one of the following two attacks: ownership piracy or ambiguity attacks.

5.2 Obscuring $\mathcal{O}$'s Ownership

Fine-tuning attack. To remove the original watermark, $\mathcal{A}$ can fine-tune M_{wm} with a new training set [**?**], [2], [16], [24], [31], [41], [58]. Specifically, $\mathcal{A}$ trains M_{wm} with a new small set that shares an underlying distribution with the original training set, thus preventing M_{wm} from losing its original functionality. At the same time, $\mathcal{A}$ does not include any data that are distant from the underlying distribution in the new training set in the expectation that M_{wm} will forget $\mathcal{O}$'s key images.

Model stealing attack. $\mathcal{A}$ in model stealing attacks [24], [37], [50] aims to copy the functionality of M_{wm} into a new model, except for the capability of remembering the trigger set. To this end, $\mathcal{A}$ labels arbitrary images by querying M_{wm}. Using the constructed training set, $\mathcal{A}$ trains a model from scratch. The new model may forget $\mathcal{O}$'s key images

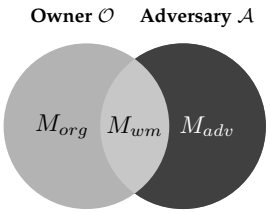

Fig. 1: The difference between models accessible to $\mathcal{A}$ who aims to claim her ownership and $\mathcal{O}$.

because the distribution represented by the arbitrary images is highly likely not to include $\mathcal{O}$'s trigger set.

Parameter pruning attack. As an attempt to make M_{wm} forget a trained trigger set, $\mathcal{A}$ in parameter pruning attack scenarios [16], [24], [35], [36], [41], [58] prunes certain parameters of M_{wm}. The original goal of model pruning is to reduce the number of redundant parameters in DNNs. However, recall that model watermarking is possible due to the over-parameterization of DNNs. $\mathcal{A}$ expects M_{wm} to lose the capability of remembering the key images after the pruning of some trained parameters, thus causing $\mathcal{O}$'s ownership claim to fail.

Evasion attack. $\mathcal{A}$ conducting evasion attacks [24], [31], [34], [36] may attempt to detect key images on the fly when $\mathcal{O}$ queries M_{wm}. Recall that key images do not belong to the underlying distribution of regular images. Thus, $\mathcal{A}$ can distinguish key images by checking the distribution of a given image. Once $\mathcal{A}$ finds a suspicious image, she can evade the verification process by returning a random label.

5.3 Claiming Ownership by $\mathcal{A}$

Ownership piracy attack. In ownership piracy attacks [2], [24], [35], [41], $\mathcal{A}$ attempts to implant her own new trigger set into M_{wm} to claim the ownership. Specifically, $\mathcal{A}$ prepares a new trigger set that is different from the original and then retrains M_{wm} with the new trigger set. After training, M_{adv} will classify $\mathcal{A}$'s key images as their target labels, and $\mathcal{A}$ can fraudulently claim the ownership of M_{adv}, which leads to conflicting ownership arguments.

Ambiguity attack. To claim ownership, in an ambiguity attack scenario [16], [17], [58], $\mathcal{A}$ generates a set of counterfeit key images that can trigger the target labels. Similar to model inversion attacks [17], $\mathcal{A}$ gradually updates regular images by leveraging gradient descent so that M_{wm} classifies the updated images as their predefined labels. The core difference of this attack compared to ownership piracy attacks is that the adversary in this scenario does not modify M_{wm} but creates counterfeit key images by leveraging M_{wm}.

Assume a scenario where $\mathcal{A}$ launches an ambiguity attack against M_{wm} trained on CIFAR-10 and watermarked using $WM_{content}$. $\mathcal{A}$ can add quasi-imperceptible perturbations to "apple" images taken from CIFAR-100 such that M_{wm} classifies each image as an "airplane." In this scenario, $\mathcal{A}$ can verify the ownership based on $WM_{unrelated}$ using the perturbed images as key images.

5.3.1 Shortcomings of Evaluation

Recall from §5.1 that ownership piracy and ambiguity attacks inevitably cause a stalemate between $\mathcal{A}$ and $\mathcal{O}$ with conflicting ownership arguments based on their respective watermarks. In this regard, previous studies [2], [16], [24], [35], [41], [58] have demonstrated the degree to which their watermarking algorithms can withstand these attacks. However, we claim that there exists a straightforward solution to manifest the true owner in these attack scenarios; thus, their evaluation should have been performed assuming a different scenario.

We note that there exists a clear difference between the capabilities of $\mathcal{O}$ and $\mathcal{A}$, as shown in Figure 1. Because $\mathcal{A}$ steals the model after $\mathcal{O}$ watermarks M_{org}, $\mathcal{A}$ cannot access M_{org} which does not have any watermarks. Accordingly, in court, a judge may request that both $\mathcal{O}$ and $\mathcal{A}$ provide a functional model without any watermarks. Then, $\mathcal{O}$ can prove the ownership by providing M_{org}, which $\mathcal{A}$ cannot provide. $\mathcal{A}$ will lose this ownership dispute game due to the inability to present a functional model that remembers none of the key images and achieves a test accuracy comparable to M_{adv} at the same time.

We emphasize that $\mathcal{A}$ in ownership piracy or ambiguity attack scenarios does not possess the aforementioned functional model without key images. One may argue that $\mathcal{A}$ can present this model to the court by conducting an attack that removes $\mathcal{O}$'s watermark. However, if $\mathcal{A}$ was able to remove $\mathcal{O}$'s watermark, the ownership dispute would not have occurred in the first place because the attacker would have used the watermark-removed model for hosting the service.

This verification leveraging the adversary's inability of presenting a watermark-free model is analogous to that in the traditional image and video watermarking research [1], [15]. To prevent the threat of an adversary claiming ownership by means of blending her watermarks on top of the owner's watermarked image, it is common in the verification to ask the adversary to present the original image without any watermarks.

Therefore, we propose the following more plausible scenario in which $\mathcal{A}$ aims to obscure O's ownership and claim her ownership at the same time. To achieve both goals together, we insist that $\mathcal{A}$ should first mount an attack that removes $\mathcal{O}$'s watermark and then launch attacks devised to claim $\mathcal{A}$'s ownership against the watermark-removed model, thus constructing a model that only remembers $\mathcal{A}$'s trigger set. Unfortunately, no previous studies have considered these attacks together. On the contrary, we considered this new scenario by performing ownership piracy and ambiguity attacks against target models after removing $\mathcal{O}$'s watermark (§7.4).

5.4 Adaptive Attacks

We argue that the robustness of watermarked models should not be undermined by the adversary's prior knowledge of target watermarking algorithms. Considering that any insiders are able to leak the algorithms, solely depending on the security by obscurity is not a desirable goal that follow-up watermarking studies should pursue. Carlini *et al.* have also emphasized the necessity of evaluations against adaptive attacks for demonstrating adversarial robustness [5].

To this end, we propose novel adaptive attacks in which the adversary can adapt their attacks to a given watermarking scheme. In adaptive attacks, the adversary aims to obscure $\mathcal{O}$'s ownership by modifying M_{wm} to remove $\mathcal{O}$'s trigger set. For this, the adaptive adversary removes $\mathcal{O}$'s trigger set by employing the same fine-tuning, model stealing, and pruning attacks (§5.2). The key difference is that this adversary engineers a new trigger set that plays a role similar to $\mathcal{O}$'s trigger set against M_{wm} and leverages this new trigger set when launching the aforementioned three watermark removal attacks. In the following, we explain how the adversary can adaptively create the new trigger set based on M_{wm}'s watermarking algorithm.

We propose a general framework that the adaptive adversary leverages to create a new trigger set. Since the adversary seeks to generate new key images that serve as $\mathcal{O}$'s key images, the new key images should have an underlying distribution similar to that of the original key images. At the same time, the new key images should be able to trigger attacker-specified target labels. To achieve these two goals, we propose to train an autoencoder such that (1) the output images have a distribution similar to images that the watermarking scheme of M_{wm} generates and (2) M_{wm} classifies each output image as a target label. Note that the adaptive adversary can train such an autoencoder by leveraging her knowledge about the target watermarking scheme and white-box access to the stolen target model. Specifically, given a source image x and a target label y_t, the adversary trains the autoencoder to minimize the following loss function.

$$\begin{aligned} x' &= AutoEncoder(x) \\ L(x, y_t) &= L_{ae}(x, x') + \lambda \cdot L_f(y_t, f(x')) \end{aligned} \tag{1}$$

In Equation 1, the loss function has two terms: L_{ae} and L_f. These terms are designed to achieve the autoencoder's two training objectives, respectively. L_{ae} refers to a relationship between the input and output images that the adversary can adaptively define based on a target watermarking scheme, and L_f refers to the classification error of a target model.

To perform strong attacks, it is important to choose well-suited source images x and a loss function L_{ae} so that the autoencoder is able to learn how a target watermarking scheme performs the `GenerateKeyImgs` function in Algorithm 1 with high fidelity. For instance, consider $WM_{abstract}$ [2] as a target watermarking scheme. In this case, the adversary can use arbitrary abstract images collected from the Internet as source images and choose the mean squared error loss function as L_{ae} so that the output images x' become abstract images that can trigger target labels when given to M_{wm}. We describe the source images and loss functions that we chose to model each of our target watermarking schemes in Supplemental Material 2. Note that we have devised fine-tuning, model stealing, and parameter pruning adaptive attacks for each watermarking scheme, yielding 30 attack variants (3 attacks $\times$ 11 schemes).

Besides the source images x and the loss function L_{ae}, the adaptive adversary also needs to specify the target label y_t to train this autoencoder but has no prior knowledge about the target labels of the original key images. Therefore, the adversary repeatedly trains this autoencoder for each class while assuming the current class as a target label. Then, the adversary collects trigger set pairs (x', y_t) from all trained autoencoders and leverages all the collected pairs when initiating the watermark removal attacks. The adversary expects these trigger set pairs to effectively contribute to removing the original trigger set of a target model.

6 Implementation

We implemented the target watermarking algorithms and attacks using TensorFlow 2.7.0. However, publicly available code for $WM_{passport}$ and $WM_{entangled}$ are written in PyTorch 1.10.1 and TensorFlow 1.14.0, respectively. Since it requires a huge engineering effort to migrate them to TensorFlow 2.7.0, we used the corresponding frameworks to implement the attacks targeting these two schemes. The remaining nine target algorithms were implemented by referring to their papers and code if available.

7 Evaluation

In this section, we evaluate the robustness of the 11 trigger set-based watermarks. We first explain the datasets and DNN models that we used (§7.1) and demonstrate how we successfully implanted watermarks into the DNN models using the target watermark schemes in our experimental settings (§7.2). We then conduct the adversarial evaluation of each attack that we have discussed so far (§7.3.1–§7.4.2).

7.1 Datasets and Target Models

Dataset. We use the MNIST, GTSRB, CIFAR-10, TinyImageNet, and CIFAR-100 datasets. All the prior studies have only evaluated their algorithms using at most four datasets. We use these five widely adopted datasets of various sizes for extensive evaluation.

DNN models. For MNIST and TinyImageNet, we prepared LeNet-5 models [29] and EfficientNetV2S models [48]. For the remaining datasets, we implemented ResNet-56 models [22]. However, we employed ResNet-18 for all five datasets to evaluate $WM_{passport}$ and $WM_{entangled}$ in the same setup as provided by the authors (recall §6). Note that these models have been widely adopted in previous studies [2], [16], [21], [31], [36]. Since these three models show outstanding performance, they are highly likely to be deployed in real-world cases, rendering them good target models for watermark implantation.

7.2 Embedding Watermarks into the DNN Models

To build M_{wm}, we watermarked the DNN models trained on the five datasets by leveraging each algorithm, yielding a total of 55 target DNN models (5 datasets $\times$ 11 schemes). Note that each M_{wm} should maintain its classification accuracy and emit the predefined target labels for given key images.

Table 2 shows the recall rate of watermark key images and accuracy for the test instances on M_{wm}. The second to the sixth columns summarize the trigger set recall of M_{wm} across datasets, the fraction of the watermark key images that are correctly classified as their target labels. Most M_{wm} correctly remember their trigger sets and classify key

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3196790

TABLE 2: Performance of the target models M_{wm} on four datasets: MNIST (MN), GTSRB (GT), CIFAR-10 (C10), Tiny-ImageNet (TI), and CIFAR-100 (C100). Numbers in parentheses denote the degree to which test accuracy dropped compared to a model without watermarks.

	Trigger Set Recall (%)					Test Acc. (%)				
	MN	GT	C10	TI	C100	MN	GT	C10	TI	C100
Content	99.90	100	100	100	100	98.85 (-0.21)	94.75 (-0.28)	93.09 (0.12)	77.16 (-0.74)	71.71 (-0.66)
Noise	100	100	100	100	100	99.04 (-0.02)	94.89 (-0.13)	93.20 (0.23)	78.55 (0.65)	72.77 (0.40)
Unrelated	99.94	100	100	100	100	99.02 (-0.04)	94.32 (-0.70)	92.91 (-0.06)	78.49 (0.59)	72.51 (0.14)
Mark	98.72	99.72	99.64	99.90	94.79	98.94 (-0.12)	96.56 (1.53)	92.42 (-0.55)	73.96 (-3.94)	70.85 (-1.52)
Abstract	100	100	100	100	100	99.02 (-0.04)	95.08 (0.06)	92.93 (-0.04)	78.08 (0.18)	72.46 (0.09)
Adv	100	100	100	99.00	100	99.21 (0.15)	97.21 (2.18)	91.79 (-1.18)	77.76 (-0.14)	71.78 (-0.59)
Passport	84.00	100	82.00	100	93.00	99.12 (-0.23)	94.29 (0.95)	88.63 (-2.72)	60.35 (-3.12)	63.17 (-4.87)
Encoder	100	96.94	99.20	99.80	99.20	98.98 (-0.08)	93.13 (-1.90)	92.67 (-0.30)	77.15 (-0.75)	72.19 (-0.18)
Exp	100	100	100	100	100	99.07 (0.01)	94.54 (-0.49)	92.62 (-0.35)	77.30 (-0.60)	71.53 (-0.84)
DeepSigns	100	100	100	100	100	99.09 (0.03)	95.79 (0.76)	91.69 (-1.28)	77.21 (-0.69)	70.27 (-2.10)
Entangled	100	81.25	15.54	60.94	62.76	98.84 (0.58)	95.26 (1.22)	93.10 (3.08)	56.81 (3.84)	73.45 (6.33)

samples with a high recall of over 99%. The seventh to the eleventh columns describe the test accuracy for M_{wm} as well as the magnitude of drops in test accuracy in comparison to the corresponding models without any watermark. We observe that most M_{wm} preserve their test accuracy after watermarking, showing only a slight drop of within 4%.

However, the models watermarked using $WM_{passport}$ and $WM_{entangled}$ yielded relatively low trigger set recall levels compared to the other target models. Note that we used the original $WM_{passport}$ and $WM_{entangled}$ implementation, resulting in no differences between our implementation and that of the authors. Since $WM_{passport}$ utilizes abstract images as its key images, we believe that this result stems from using a different set of abstract key images. For the $WM_{entangled}$ models, our results accord with the trigger set recall levels reported by Jia *et al.* [24]. We believe that the initial trigger set recalls of the $WM_{entangled}$ models are already too low for ownership claims. In the remaining sections, we evaluate the presented attacks using these watermarked models.

7.3 Obscuring $\mathcal{O}$'s Ownership

An adversary seeking to obscure $\mathcal{O}$'s ownership attempts to thwart $\mathcal{O}$'s ownership verification process. For this, the adversary can employ fine-tuning, model stealing, evasion, or parameter pruning attacks against M_{wm}, thus generating M_{adv} with a low $\mathcal{O}$'s trigger set recall. At the same time, the test accuracy of M_{adv} should not drop significantly as the adversary needs to host a functional service by leveraging M_{adv}. We now evaluate each attack in this category assuming both non-adaptive and adaptive adversaries.

TABLE 3: Trigger set recall (%) of M_{adv} after fine-tuning attacks.

	Non-adaptive Attack					Adaptive Attack				
	MN	GT	C10	TI	C100	MN	GT	C10	TI	C100
Content	57.02	0.36	24.54	0.00	13.20	39.74	53.96	26.92	0.00	75.20
Noise	5.93	84.46	99.14	0.40	93.80	0.36	5.63	3.78	0.40	10.20
Unrelated	99.34	100	99.10	99.40	92.80	32.76	99.77	17.26	15.80	0.00
Mark	40.28	8.95	3.86	5.64	2.29	19.77	25.87	8.02	1.25	1.46
Abstract	51.00	51.00	60.00	100	26.00	45.00	83.00	54.00	100	23.00
Adv	35.00	79.00	24.00	66.00	13.00	14.00	8.00	12.00	6.00	2.00
Passport	14.00	43.00	14.00	74.00	3.00	13.00	43.00	17.00	75.00	3.00
Encoder	20.00	4.08	20.00	7.00	8.00	17.00	7.14	20.60	1.60	5.60
Exp	6.00	0.00	1.00	5.00	1.00	7.00	0.00	2.00	9.00	0.00
DeepSigns	11.00	1.00	8.00	1.00	0.00	11.00	1.00	12.00	0.00	2.00
Entangled	99.81	27.34	4.57	2.68	40.89	97.42	33.59	1.48	4.69	23.18

7.3.1 Fine-tuning Attack

Non-adaptive attack. A non-adaptive adversary tunes M_{wm} on a dataset that does not include any key images, thus constructing another model M_{adv}. As $\mathcal{A}$ has access to 50% of a test set, we leveraged this set to fine-tune M_{wm}; however, using this set alone might decrease the test accuracy of M_{adv}. Therefore, we also used an extra set of images when simulating fine-tuning attacks. Similar to the method proposed by Chen *et al.* [12], we collected arbitrary images and labeled each of them with the output of M_{wm}. For M_{wm} trained on MNIST, we collected all images from the Fashion-MNIST dataset [54]. We took images from CIFAR-100 for fine-tuning the GTSRB, CIFAR-10, and TinyImageNet models. For M_{wm} trained on CIFAR-100, we collected images from CIFAR-10.

Adaptive attack. In addition to these training instances, the adaptive adversary harnesses the autoencoder-generated key images to make M_{wm} unlearn $\mathcal{O}$'s trigger set. Recall from §5.4 that this adversary collects x', which is designed to resemble $\mathcal{O}$'s key images that trigger y_t. Therefore, the adversary assigns a random label other than y_t to x' and provides this pair as a training instance for fine-tuning attacks, expecting that M_{adv} will interpret $\mathcal{O}$'s key images as the adversary-chosen random classes. For training each autoencoder, it takes 3–12 minutes for each class, depending on the dataset.

When fine-tuning M_{wm}, we optimized M_{wm} using Adam [27] and trained M_{wm} for 10 epochs. We fixed the learning rates at 0.01, 0.0001, and 0.0005 for MNIST, TinyImageNet, and the other datasets, respectively, except for one case: for M_{wm} trained on the CIFAR datasets and watermarked using $WM_{passport}$, we used a fixed learning rate of 0.0001. We selected these learning rates after exploratory experiments.

Table 3 presents the trigger set recall of M_{adv}, which is the resulting model after conducting fine-tuning attacks on M_{wm}. Note that it is challenging to set a minimum trigger set recall sufficient to prove $\mathcal{O}$'s ownership. Thus, in the table, we colored the cells of vulnerable watermarking schemes that rendered a trigger set recall lower than the threshold varying from 10% to 80%. The gradations represent the extent to which the model is vulnerable to the attacks. We excluded M_{adv} that exhibited over a 5%

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3196790

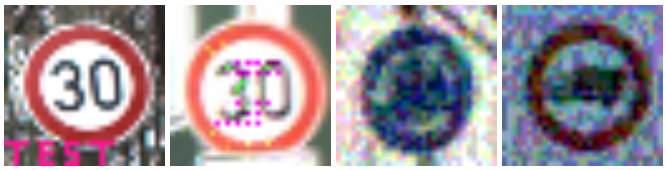

Fig. 2: Pairs of $\mathcal{O}$'s key image and an autoencoder-generated key image. The left and the right halves show examples of $WM_{content}$ and WM_{noise}, respectively.

drop in test accuracy because these models do not suffice for the adversary's goal of hosting functional services. In Supplemental Material 5, we include an expanded version of Table 3 that displays the test accuracies of M_{adv} as well as their trigger set recalls.

The left half of the table shows the results for the non-adaptive attacks. When we set 10% as the minimum trigger set recall to prove ownership, the non-adaptive fine-tuning attacks only worked against the 19 target models. However, the number of vulnerable target models jumped to 36 in total when the minimum requirement was set to 80%. Interestingly, all models watermarked using WM_{mark} and DeepSigns were vulnerable to this fine-tuning attack. On the other hand, all models watermarked with $WM_{unrelated}$ were robust, demonstrating trigger set recalls of over 92% for all the datasets.

The right half of the table summarizes the results for the adaptive attacks. Note in the table that the adaptive attacks further destroyed schemes that were robust to the non-adaptive attacks. For instance, WM_{noise} and $WM_{unrelated}$ models exhibited significant trigger set recall drops. On the other hand, the GTSRB model with $WM_{unrelated}$ was robust to the adaptive attack. Recall that we collected an extra set of images when conducting fine-tuning attacks to preserve the test accuracy. We observed that this extra set hindered unlearning the trigger set of the GTSRB model with $WM_{unrelated}$. Specifically, we found that if we exclude this set when launching the attacks, we can successfully decrease the trigger set recall down to 0% without loss of test accuracy. Considering that the adversary can either include or exclude this extra set when launching fine-tuning attacks, we conclude that $WM_{unrelated}$ is also vulnerable to the adaptive fine-tuning attack.

We also observed that adaptive attacks are worse than non-adaptive attacks in several cases. For instance, the non-adaptive and adaptive fine-tuning attacks against the GTSRB model with $WM_{content}$ reduce the trigger set recall to 0.36% and 53.96%, respectively. We carefully analyzed these cases and founded that the performance of autoencoders used for conducting adaptive attacks varies considerably based on the target watermarking algorithms. Figure 2 shows examples of $\mathcal{O}$'s key images and autoencoder-generated images. Note from the figure that the autoencoders trained to simulate WM_{noise} is capable of generating an image that has a distribution similar to that of $\mathcal{O}$'s key images. On the other hand, the autoencoders reported poor performance against a watermarking scheme embedding contents, which requires more sophisticated trigger generation. We thus conclude that this performance difference has affected the success of adaptive attacks in removing the embedded trigger sets.

TABLE 4: Trigger set recall (%) of M_{adv} after model stealing attacks.

	Non-adaptive Attack					Adaptive Attack				
	MN	GT	C10	TI	C100	MN	GT	C10	TI	C100
Content	82.94	0.00	2.04	1.00	0.80	28.37	0.05	1.18	1.80	0.40
Noise	0.21	59.19	3.60	2.20	45.60	0.09	6.76	4.04	0.20	87.60
Unrelated	99.76	100	95.26	54.60	0.00	34.94	100	9.02	20.20	0.00
Mark	11.57	5.10	2.32	0.66	0.90	7.66	7.27	6.81	0.81	1.46
Abstract	41.00	35.00	24.00	65.00	2.00	39.00	48.00	27.00	66.00	2.00
Adv	23.00	70.00	8.00	31.00	11.00	17.00	0.00	16.00	17.00	1.00
Passport	7.00	34.00	19.00	60.00	2.00	7.00	37.00	16.00	57.00	1.00
Encoder	9.67	2.55	12.60	0.80	1.80	9.33	2.30	13.20	1.00	1.40
Exp	1.00	0.00	2.00	0.00	0.00	4.00	0.00	2.00	0.00	2.00
DeepSigns	10.00	3.00	6.00	0.00	1.00	6.00	2.00	11.00	0.00	0.00
Entangled	99.93	75.78	52.02	8.04	35.68	96.20	25.00	37.62	6.70	21.35

We emphasize that none of the previous studies conducted the adaptive attack even though they are mostly vulnerable to this attack. Furthermore, although eight out of the 11 watermarking algorithms had already been evaluated against fine-tuning attacks in previous studies [2], [16], [24], [31], [41], [58], our analysis reveals that many of them are still vulnerable. This implies that fine-tuning attacks that previous studies have conducted were too weak to construct a meaningful upper bound of their watermarking algorithms. Therefore, we recommend that follow-up studies evaluate their schemes against fine-tuning attacks with sufficiently strong settings and demonstrate the extent to which their watermarks can withstand attacks without being removed. We further investigate various attack settings that can affect the strength of fine-tuning attacks in Supplemental Material 3.

7.3.2 *Model Stealing Attack*

In model stealing attacks, an adversary does not have enough training instances to train a new model from scratch. Thus, the adversary prepares a set of arbitrary images and leverages M_{wm} to label these images. The adversary then trains M_{adv} from scratch on these instances, thereby copying M_{wm}'s functionality except for the capability of remembering the trigger set. We consider both non-adaptive and adaptive adversaries in evaluating the target models against model stealing attacks.

Non-adaptive attack. To collect training instances, we took the same approach as we did for fine-tuning attacks (§7.3.1). For training, we selected M_{adv} to have the same model structure as M_{wm}. Note that the adversary knows the exact structure of M_{wm} because M_{wm} is already in her hands. We performed model stealing attacks by training this new model from scratch with the collected dataset.

Adaptive attack. The adaptive adversary in this attack scenario also leverages the trigger set created with the autoencoders to preclude M_{adv} from learning $\mathcal{O}$'s trigger set. The adversary prepares training instances in the exact same way as the adaptive adversary in fine-tuning attacks and appends them to the training set for training M_{adv}. Because the adversary feeds x' with a random label to M_{adv} for its training, this new model cannot learn $\mathcal{O}$'s trigger set.

Table 4 summarizes the experimental results of model stealing attacks. We shaded (in red) the cells according to the same criteria as we did for Table 3. The left half of the table

TABLE 5: Trigger set recall (%) of M_{adv} after parameter pruning attacks.

	Non-adaptive Attack					Adaptive Attack				
	MN	GT	C10	TI	C100	MN	GT	C10	TI	C100
Content	99.87	100	100	99.80	100	64.95	100	100	0.00	100
Noise	100	100	100	99.00	100	97.29	0.27	100	0.00	58.20
Unrelated	99.38	100	100	70.20	100	17.22	0.81	90.98	100	4.00
Mark	97.40	99.64	99.64	41.65	94.63	69.03	97.03	96.97	51.80	69.96
Abstract	73.00	100	100	74.00	100	78.00	95.00	97.00	3.00	98.00
Adv	91.00	100	100	41.00	100	96.00	7.00	97.00	12.00	94.00
Passport	80.00	100	71.00	99.00	87.00	84.00	94.00	82.00	94.00	91.00
Encoder	96.50	96.94	99.20	80.80	98.60	99.00	91.07	98.20	0.30	92.80
Exp	92.00	100	100	4.00	100	92.00	99.00	98.00	0.00	97.00
DeepSigns	39.00	89.00	100	2.00	98.00	81.00	98.00	99.00	12.00	78.00
Entangled	100	83.59	17.81	70.09	61.98	99.95	71.88	5.33	2.68	57.81

presents the results from the non-adaptive attacks. Overall, the target models underwent a more drastic trigger set recall drop compared to fine-tuning attacks. However, the target models trained on the CIFAR-100 and TinyImageNet dataset experienced a huge test accuracy drop along with a trigger set recall drop, which indicates that our model stealing attacks are ineffective against target models with a large number of classes. As a result, assuming the minimal trigger set recall to be 10%, 16 target models were vulnerable to this attack. When we consider 80% as the minimal requirement, 26 models failed to verify $\mathcal{O}$'s ownership. Among the 11 watermarking schemes, M_{wm} with WM_{exp} and DeepSigns were the most vulnerable models, showing a trigger set recall of below 10% after the attacks.

The right half of the table shows the results of the adaptive model stealing attacks. Considering 10% as the required minimal trigger set recall, the watermarks embedded in 18 out of the 55 target models were destroyed by the adaptive attack. We also note that the trigger set recalls further decreased in most target models compared to the non-adaptive attack. Moreover, when we raise the bar to 80%, all 11 watermarking schemes were broken by this attack. Although $WM_{unrelated}$ was robust against the non-adaptive model stealing attack, it was destroyed by the adaptive attack.

Note that most target models were vulnerable to the non-adaptive and adaptive model stealing attacks. This means that the current watermark evaluation practice does not consider real-world threats properly. We stress that researchers should evaluate robustness against the complete set of attacks, including model stealing attacks, and raise the bar of watermarking schemes' robustness with aggressive evaluation that considers an adaptive adversary.

7.3.3 Parameter Pruning Attack

The non-adaptive and adaptive adversaries in parameter pruning attacks attempt to prune the parameters of M_{wm}.

Non-adaptive attack. To erase $\mathcal{O}$'s watermark, the non-adaptive adversary prunes p% of the smallest parameters in M_{wm}, thus building a new model M_{adv}.

Adaptive attack. In adaptive pruning attacks, the adversary identifies parameters that contribute to the classification of $\mathcal{O}$'s trigger set by leveraging the autoencoder-generated trigger set and then removes those parameters. Specifically, the adversary observes the differences between the neuron activations of M_{wm} when x and x' are given. Note that the neurons that render different behaviors between these images can be regarded as trigger set-related. The adversary thus prunes p% of parameters that showed the greatest differences. After pruning, M_{adv} becomes non-reactive to $\mathcal{O}$'s trigger set. When pruning parameters, we only considered parameters that belong to the fully connected layers.

Table 5 presents the trigger set recall of M_{adv} after parameter pruning attacks. We evaluated the effect of this attack with six different values of p: 5, 10, 20, 40, 60, and 80. Among the results for the six different p values, we only show the results that reported the lowest trigger set recall with a test accuracy drop of less than 5%. We colored the cells according to the same criteria that we set for Table 3. The left half shows the trigger set recall after the non-adaptive attacks. In general, we found that the watermarking schemes are robust against this attack, which accords with the experimental results of previous studies [16], [24], [35], [36], [41], [58]. Seven out of 55 target models showed a trigger set recall of less than 80%; only two models were weak against this attack when we considered 40% as the minimal recall required to prove ownership.

The right half of the table summarizes the experimental results after the adaptive pruning attacks. The adaptive pruning attacks were not as strong as other adaptive attacks; however, the adaptive attack damaged five target models that were robust to the non-adaptive attacks. Furthermore, note that the $WM_{unrelated}$ models tend to demonstrate a significant drop in the trigger set recall, although they experience non-trivial test accuracy drops as well (see Supplemental Material 5). These results suggest the necessity of our adaptive pruning attacks against the existing watermarking algorithms.

7.3.4 Evasion Attack

The goal of $\mathcal{A}$ in performing an evasion attack is to distinguish queries that have key images from normal queries. Once a key image is identified, the adversary may return random labels to drop the trigger set recall, thus obscuring $\mathcal{O}$'s ownership.

To assess $\mathcal{A}$'s capability of distinguishing key images from regular images, we trained autoencoders for each class of images with 50% of a test set. For instance, we prepared a total of 100 autoencoders for CIFAR-100. We then evaluated whether the trained autoencoders could output an image similar to the input image. Note that these autoencoders are able to reconstruct normal images well but fail with key images as the autoencoders are trained on regular images. To decide whether the autoencoders fail to reconstruct given images, we computed three metrics, i.e., L_1 norms, L_2 norms, and Jensen-Shannon divergence, between the input and output images as in the approach of [34].

Specifically, given an image, we query M_{wm} and record the output class. We then reconstruct the image with the autoencoder of the output class and compute the metrics. If all three metrics computed from the image are lower than the thresholds, we consider the given image to be a normal one. We set the thresholds such that false-positive rates are at most 0.1% on the set of images used for training the autoencoders.

TABLE 6: Trigger set detection accuracies when performing evasion attacks against the target models. A table cell in the red background represents a vulnerable model that enables $\mathcal{A}$ to detect the trigger set with an accuracy of over 85%, and the gradations represent the extent to which the model is vulnerable to evasion attacks.

	Detection Acc. (%)				
	MNIST	**GTSRB**	**C10**	**TI**	**C100**
Content	98.31	97.57	98.62	79.60	89.20
Noise	98.38	97.75	99.64	90.20	90.30
Unrelated	98.32	97.91	49.73	39.40	86.70
Mark	98.29	93.18	93.88	75.20	86.38
Abstract	99.50	87.00	67.50	60.00	81.00
Adv	99.50	97.50	100	93.50	91.00
Passport	99.50	89.00	60.50	50.00	78.50
Encoder	94.75	85.59	89.40	79.35	87.10
Exp	62.50	85.50	78.00	55.50	82.00
DeepSigns	100	97.00	100	93.50	93.00
Entangled	98.55	92.19	52.47	84.49	87.76

Table 6 summarizes the detection accuracies of evasion attacks. We balanced the number of key images and regular images when measuring the detection accuracy so that the baseline detection accuracy is 50%. These regular images were taken from the training set so that they would not overlap with the images used to train the autoencoders. A high detection accuracy implies that the adversary can successfully reduce the trigger set recall without losing test accuracy.

As shown in the table, detection accuracies against the TinyImageNet models are lower than those against the other models. Since we train an autoencoder for each class, the number of training instances to train each autoencoder becomes extremely limited (e.g., 25 images) when attacking the TinyImageNet models. Nevertheless, note in the table that 38 target models out of 55 can successfully evade the verification process as they reported at least 85% detection accuracies. This is not surprising as only three out of the 11 previous studies have considered evasion attacks in their adversarial evaluation. Among the three previous studies that considered evasion attacks, WM_{exp} is robust against this attack, as shown in the table. This is because it takes key images from exactly the same distribution as the regular images used for training WM_{exp}. However, $WM_{encoder}$ was vulnerable to evasion attacks in our settings, even though a previous study [31] demonstrated its robustness against this attack scenario. That is, the previous study took a naive approach to conduct evasion attacks so that it failed to demonstrate a meaningful upper bound on its robustness against evasion attacks (see Supplemental Material 3).

7.4 Claiming Ownership by $\mathcal{A}$

The goal of a non-adaptive adversary claiming her ownership is to cause a stalemate in the ownership dispute game against $\mathcal{O}$. To simulate this adversary, all prior research has considered a scenario where an adversary conducts single ownership piracy or ambiguity attacks. However, there exists an obvious solution to identify the authentic owner; thus the adversary necessarily loses in this game (§5.3.1).

TABLE 7: Trigger set recalls of M_{adv} after ownership piracy attacks. Numbers in parentheses denote the differences of trigger set recalls between $\mathcal{A}$ and $\mathcal{O}$.

	$\mathcal{A}$'s Recall (%)					$\mathcal{O}$'s Recall (%)				
	MN	**GT**	**C10**	**TI**	**C100**	**MN**	**GT**	**C10**	**TI**	**C100**
Content	86.00 (36.24)	99.00 (98.96)	100 (99.72)	100 (99.00)	100 (99.80)	49.76	0.05	0.28	1.00	0.20
Noise	92.00 (91.91)	99.00 (74.95)	100 (97.20)	95.83 (95.63)	100 (78.00)	0.09	24.05	2.80	0.20	22.00
Unrelated	96.00 (86.72)	98.00 (37.28)	94.00 (43.48)	100.00 (77.60)	99.00 (99.00)	9.28	60.72	50.52	22.40	0.00
Mark	87.00 (75.46)	99.00 (93.38)	98.00 (94.94)	22.46 (21.44)	100 (99.32)	11.54	5.62	3.06	1.03	0.68
Abstract	89.00 (61.00)	99.00 (67.00)	98.00 (81.00)	100 (39.00)	100 (100)	28.00	32.00	17.00	61.00	0.00
Adv	75.00 (41.00)	91.00 (79.00)	98.00 (92.00)	100 (70.00)	100 (93.00)	34.00	12.00	6.00	30.00	7.00
Passport	94.00 (89.00)	0.00 (-6.00)	0.00 (-10.00)	4.00 (-44.00)	15.00 (14.00)	5.00	6.00	10.00	48.00	1.00
Encoder	98.00 (88.00)	100 (97.70)	100 (88.20)	100 (99.50)	100 (99.00)	10.00	2.30	11.80	0.50	1.00
Exp	70.00 (70.00)	98.00 (98.00)	98.00 (96.00)	100 (100)	100 (100)	0.00	0.00	2.00	0.00	0.00
DeepSigns	93.00 (88.00)	100 (98.00)	99.00 (92.00)	100 (99.00)	100 (100)	5.00	2.00	7.00	1.00	0.00
Entangled	100 (0.00)	100 (67.97)	99.93 (87.86)	95.31 (93.97)	98.05 (98.05)	100	32.03	12.08	1.34	0.00

With this in mind, we propose a new attack scenario that incorporates watermark removal attacks within ownership claiming attacks. Specifically, we consider a novel scenario where the adversary first removes $\mathcal{O}$'s watermark and then implants $\mathcal{A}$'s watermark, thus claiming the ownership of a new model that only holds $\mathcal{A}$'s watermark. Among the watermark removal attacks, we chose models constructed via model stealing attacks as a base for ownership piracy and ambiguity attacks due to model stealing attacks' outstanding performance in removing watermarks (recall §7.3.2).

Recall that the non-adaptive adversary in these attack scenarios claims ownership based on her own trigger set. In other words, the adversary needs to choose one watermarking algorithm to prepare her trigger set. For this, we assumed that $\mathcal{A}$ prepares her trigger set using $WM_{unrelated}$. Hence, $WM_{unrelated}$ becomes the basis of $\mathcal{A}$'s fraudulent ownership claim of the resulting model M_{adv}.

7.4.1 Ownership Piracy Attack

To perform piracy attacks, the adversary follows the same procedures and settings as fine-tuning attacks. The only difference is that $\mathcal{A}$ also appends her trigger set to the dataset of a fine-tuning attacker. With this dataset, $\mathcal{A}$ fine-tunes a watermark-removed model to embed her trigger set.

Table 7 presents the trigger set recalls of $\mathcal{A}$ and $\mathcal{O}$ after the attack. We compare these two trigger set recalls because $\mathcal{A}$ in this attack insists that M_{adv} only contains $\mathcal{A}$'s trigger set and has never been trained on $\mathcal{O}$'s trigger set. That is, $\mathcal{A}$ aims to demonstrate that $\mathcal{A}$'s trigger set recall is high, whereas $\mathcal{O}$'s trigger set recall is low. We thus show the differences between the trigger set recall rates of $\mathcal{A}$ and $\mathcal{O}$ in parentheses. We shaded (in red) the target models that showed a difference greater than a threshold varying from 20% to 80%. The gradations illustrate the magnitude of each trigger recall difference. As we did for all other attacks, we excluded target models with a test accuracy drop of over 5%.

TABLE 8: Trigger set recalls of M_{wm} after ambiguity attacks. Numbers in parentheses denote the differences of trigger set recalls between $\mathcal{A}$ and $\mathcal{O}$.

	$\mathcal{A}$'s Recall (%)					$\mathcal{O}$'s Recall (%)				
	MN	GT	C10	TI	C100	MN	GT	C10	TI	C100
Content	100 (17.06)	98.00 (98.00)	100 (97.96)	100 (99.00)	100 (99.20)	82.94	0.00	2.04	1.00	0.80
Noise	100 (99.79)	98.00 (38.81)	100 (96.40)	100 (97.80)	100 (54.40)	0.21	59.19	3.60	2.20	45.60
Unrelated	98.00 (-1.76)	100 (0.00)	100 (4.74)	99.00 (44.40)	97.00 (97.00)	99.76	100	95.26	54.60	0.00
Mark	100 (88.43)	100 (94.90)	100 (97.68)	100 (99.34)	94.00 (93.10)	11.57	5.10	2.32	0.66	0.90
Abstract	100 (59.00)	100 (65.00)	100 (76.00)	100 (35.00)	100 (98.00)	41.00	35.00	24.00	65.00	2.00
Adv	100 (77.00)	100 (30.00)	100 (92.00)	99.00 (68.00)	100 (89.00)	23.00	70.00	8.00	31.00	11.00
Passport	100 (93.00)	48.00 (14.00)	0.00 (-19.00)	0.00 (-60.00)	41.00 (39.00)	7.00	34.00	19.00	60.00	2.00
Encoder	100 (90.33)	99.00 (96.45)	100 (87.40)	98.00 (97.20)	100 (98.20)	9.67	2.55	12.60	0.80	1.80
Exp	48.00 (47.00)	0.00 (0.00)	100 (98.00)	97.00 (97.00)	0.00 (0.00)	1.00	0.00	2.00	0.00	0.00
DeepSigns	100 (90.00)	100 (97.00)	100 (94.00)	100 (100)	98.00 (97.00)	10.00	3.00	6.00	0.00	1.00
Entangled	100 (0.07)	99.00 (23.22)	97.00 (44.98)	98.00 (89.96)	99.00 (63.32)	99.93	75.78	52.02	8.04	35.68

Since we removed $\mathcal{O}$'s watermark before embedding $\mathcal{A}$'s watermark, $\mathcal{A}$'s trigger set acquired a dominant position over that of $\mathcal{O}$ from 24 target models. Furthermore, from 21 target models, $\mathcal{A}$'s trigger set recall surpassed $\mathcal{O}$'s by at least 60%. These results imply that $\mathcal{A}$ can successfully take the ownership of those target models, claiming that those models only contain $\mathcal{A}$'s trigger set. Considering these results, we suggest future researchers prove their algorithms' robustness against piracy attacks based on our new scenario.

7.4.2 Ambiguity Attack

Unlike the previous adversary, an adversary performing ambiguity attacks does not implant $\mathcal{A}$'s trigger set into the watermark-removed model. Instead, $\mathcal{A}$ generates key images that can trigger the adversary-chosen target labels when given to the watermark-removed model. Consequently, $\mathcal{A}$ can claim that the resulting model only remembers $\mathcal{A}$'s trigger set. Specifically, $\mathcal{A}$ perturbs seed images by leveraging gradient descent in order to divert the classification of the perturbed images towards the adversary-chosen target label [17].

Table 8 summarizes the ambiguity attack results. We applied the same criteria as piracy attacks to color each cell. Since $\mathcal{A}$ does not modify the watermark-removed model, $\mathcal{O}$'s trigger set recalls after this attack are the same as those shown in Table 4. Among 55 target models, $\mathcal{A}$'s trigger set recalls from 23 target models were greater than $\mathcal{O}$'s by at least 20%. The second and forth images in Figure 3 show examples of key images generated by ambiguity attacks targeting the WM_{noise}-MNIST and $WM_{content}$-GTSRB models, respectively. We note that the L_2 norm of perturbations added to the seed images is less than 0.004 on average, which implies that the added perturbations are quasi-imperceptible. We thus conclude that $\mathcal{A}$ can successfully claim her ownership of those models with the created trigger set based on $WM_{unrelated}$. As we demonstrated the effectiveness of this attack, we argue that future research should also evaluate its algorithm against ambiguity attacks.

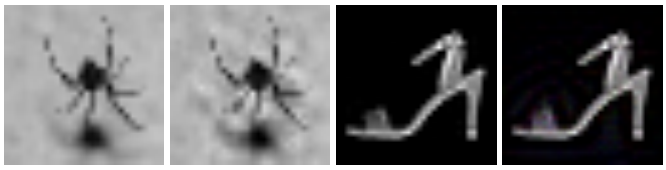

Fig. 3: $\mathcal{A}$'s key images (second and forth images) generated from seed images (first and third images) using ambiguity attacks.

8 Lessons

We have so far conducted six different attacks along with three adaptive attacks to evaluate the robustness of the 11 watermarking algorithms. Table 9 shows the number of attacks that succeeded in each target model watermarked using the given algorithm. When tallying up the totals, we only included results that reported a test accuracy drop of at most 5%. We set a trigger set recall of 80% as the minimum threshold for claiming ownership after conducting fine-tuning, model stealing, and parameter pruning attacks. For evasion attacks, we considered 85% as the minimal detection accuracy necessary to evade the verification step.

Note in the table that every watermarking algorithm is broken by at least two presented adaptive attacks and two non-adaptive attacks. When considering both adaptive and non-adaptive attacks, all schemes do not demonstrate their robustness against at least five attacks. Furthermore, six out of the 11 watermarking algorithms [2], [16], [24], [31], [41], [58] were vulnerable to attacks against that the authors had already evaluated. While all the evaluated watermarking algorithms were broken, $WM_{unrelated}$ was the most robust algorithm among them.

These results highlight that all the existing trigger set-based watermarking algorithms are not ready for real-world deployment. We believe that the demonstrated failure to establish watermark robustness stems from current research practice regarding how adversarial evaluation is conducted. We further discuss several factors that make robust watermarks (§8.1) and suggestions for adversarial evaluation (§8.2).

8.1 Robustness of Watermarking Algorithms

We analyze what makes a particular watermarking algorithm more resilient to adversarial attacks than the others.

Distances between key images and decision boundaries. Note that watermark removal attacks aim to slightly distort the decision boundaries so that the resulting model predicts the key images as a label other than the target label. In other words, when the key images are distant from the decision boundaries, the target model becomes resilient to watermark removal attacks. We thus measured the distances between key images and the decision boundaries. Specifically, we conducted the PGD attacks [33] against the target model and utilized the perturbation size required to modify the prediction of key images as a distance metric. We selected two most robust (i.e., WM_{noise} and $WM_{unrelated}$) and two most vulnerable (i.e., DeepSigns and WM_{mark})

TABLE 9: Summary of the attack results. ✓ denotes that the attack succeeded against a target model watermarked with the corresponding algorithm, whereas ✗ indicates that the attack failed. For each watermarking scheme, the successful attacks are presented in the order of MNIST, GTSRB, CIFAR-10, TinyImageNet, and CIFAR-100 models.

Attack (Adv.)	$WM_{content}$	WM_{noise}	$WM_{unrelated}$	WM_{mark}	$WM_{abstract}$	WM_{adv}	$WM_{passport}$	$WM_{encoder}$	WM_{exp}	DeepSigns	$WM_{entangled}$
Fine-tuning (non-adap.)	✓✓✓✓✗	✓✗✗✓✗	✗✗✗✗✗	✓✓✓✓✓	✓✓✓✗✗	✓✓✓✓✗	✓✓✗✗✗	✓✓✓✓✗	✓✓✓✗✗	✓✓✓✓✓	✗✓✓✓✓
Fine-tuning (adap.)	✓✓✓✓✗	✓✓✓✓✗	✓✗✓✓✗	✓✓✓✓✗	✓✗✓✗✗	✓✓✓✓✓	✓✓✗✗✗	✓✓✓✓✗	✓✓✓✗✗	✓✓✓✓✓	✗✓✓✓✗
Stealing (non-adap.)	✗✓✓✗✗	✓✓✓✗✗	✗✗✗✗✗	✓✓✓✗✗	✓✓✓✗✗	✓✓✓✗✗	✓✓✗✗✗	✓✓✓✗✗	✓✓✓✗✗	✓✓✓✗✗	✗✓✗✗✗
Stealing (adap.)	✓✓✓✗✗	✓✓✓✗✗	✓✗✓✗✗	✓✓✓✗✗	✓✓✓✗✗	✓✓✓✗✗	✓✓✗✗✗	✓✓✓✗✗	✓✓✓✗✗	✓✓✓✗✗	✗✓✗✗✗
Pruning (non-adap.)	✗✗✗✗✗	✗✗✗✗✗	✗✗✗✗✗	✗✗✗✗✗	✓✗✗✗✗	✗✗✗✗✗	✓✗✓✗✗	✗✗✗✗✗	✗✗✗✗✗	✓✗✗✗✗	✗✗✓✓✓
Pruning (adap.)	✓✗✗✗✗	✗✓✗✗✗	✓✗✗✗✗	✓✗✗✗✗	✓✗✗✗✗	✗✗✗✗✗	✗✗✗✗✗	✗✗✗✗✗	✗✗✗✗✗	✗✗✗✗✗	✗✓✓✗✗
Evasion	✓✓✓✗✓	✓✓✓✓✓	✓✓✗✗✓	✓✓✓✗✓	✓✓✗✗✓	✓✓✓✓✓	✓✓✗✗✗	✓✓✓✗✓	✗✓✗✗✗	✓✓✓✓✓	✓✓✗✗✓
Ownership Piracy	✓✓✗✗✗	✓✓✗✗✗	✓✓✗✗✗	✓✓✓✗✗	✓✓✗✗✗	✓✓✓✗✗	✓✗✗✗✗	✓✓✗✗✗	✓✓✗✗✗	✓✓✓✗✗	✗✗✗✗✗
Ambiguity	✗✓✓✗✗	✓✓✓✗✗	✗✗✗✗✗	✓✓✓✗✗	✓✓✓✗✗	✓✓✓✗✗	✓✗✗✗✗	✓✓✓✗✗	✓✗✓✗✗	✓✓✓✗✗	✗✗✗✗✗
# of Succeeded Attacks	6 7 6 2 1	7 7 5 3 1	5 2 2 2 1	8 7 7 2 2	9 6 5 0 0	7 7 7 3 2	8 5 1 0 0	7 7 6 2 1	6 6 5 0 0	8 7 7 3 3	1 6 4 2 3
Maximum # per Scheme	7	7	5	8	9	7	8	7	6	8	6

watermarking schemes against fine-tuning attacks for this evaluation. When measured over the CIFAR-10 models, we observed that the distances of WM_{noise} and $WM_{unrelated}$ are greater than those of DeepSigns and WM_{mark}.

Effect of target labels. As shown in Table 9, WM_{noise} and $WM_{unrelated}$ are more robust than the other algorithms against non-adaptive fine-tuning attacks. Note that the key difference between these two schemes and the others is that they allocate a single class to all key images, whereas the remaining schemes assign different labels to each key image. That is, the consistent labeling of these two schemes helps M_{wm} generalize on $\mathcal{O}$'s trigger set, thus making it difficult for $\mathcal{A}$ to remove $\mathcal{O}$'s watermark.

Effect of key images. We observed that WM_{exp} is the only robust algorithm against evasion attacks. Note that WM_{exp} employs images selected from the same distribution as normal training instances for key images, while the other watermarking algorithms use out-of-distribution images.

Considering these factors that affect watermark robustness, we propose the following recommendations for improving watermark robustness. First, it is better to assign a single target label to all key images rather than random labels. Second, it is better to select key images from the same distribution as a regular training set rather than from a different distribution.

8.2 Suggestions for Adversarial Evaluation

From our evaluations, we draw the following takeaways that future research on designing a secure watermarking algorithm should consider. We encourage researchers to evaluate their defenses following our suggestions discussed herein, thus demonstrating a meaningful upper bound on their robustness.

Apply the complete attack set. We found out that all the previous works were broken by already existing attacks. They could have known this result if they have conducted a complete set of existing state-of-the-art attacks to evaluate their algorithms. In this regard, we suggest future research conduct at least a complete set of state-of-the-art attacks at the time of suggesting a new approach.

Recently, several watermark removal attacks [12], [30], [43], [53] that have better performance compared to the attacks examined herein have been recently proposed. For instance, Chen *et al.* [12] adopted the elastic weight consolidation algorithm to further improve the fine-tuning attacks. We thus recommend researchers to consider these state-of-the-art attacks when evaluating their watermarking schemes.

Use adaptive attacks. All the state-of-the-art watermarking algorithms were vulnerable to the proposed adaptive attacks. We believe that our adaptive attacks serve as a better baseline for demonstrating the robustness of a target watermarking scheme. We recommend future research consider the proposed adaptive attacks when conducting fine-tuning, model stealing, and pruning attacks.

Focus on attacks that obscure $\mathcal{O}$'s ownership. Recall from §5.3.1 that an attack scenario in which the adversary conducts a single attack that claims her ownership is futile. Therefore, when evaluating attacks that aim to claim $\mathcal{A}$'s ownership, one should first launch attacks that remove

$\mathcal{O}$'s watermark and then initiate the attacks to claim $\mathcal{A}$'s ownership.

Search for effective attack hyperparameters. Surprisingly, five out of the 11 evaluated watermarking algorithms were broken by attacks that the previous studies already evaluated (recall §7.3.1 and §7.3.4). To avoid providing a misleading upper bound on robustness, follow-on research must conduct strong attacks by carefully exploring hyperparameters and adopting state-of-the-art attacks.

Consider diverse datasets. Overall, the models trained on the MNIST dataset tend to be vulnerable, as shown in Table 9. Interestingly, the watermarked models trained on CIFAR-100 and TinyImageNet were robust against the presented attacks in general. This is because the conducted attacks have always contributed to decreasing a test accuracy over 5% (see Supplemental Material 5), which means that the presented attacks on the CIFAR-100 and TinyImageNet models easily undermine the models' performance. In other words, we observed that test accuracies are prone to drop significantly after watermark removal attacks when the number of classes in a dataset increases. Note that it is well-known that various DNN defense algorithms showed different levels of robustness depending on the dataset [6]. We thus suggest considering more datasets than the MNIST and CIFAR datasets when evaluating watermarking algorithms.

9 Related Work

Backdoor attacks. There have been several studies on backdoor attacks against DNN models [11], [20]. In this type of attack, a user sends a training set to the adversarial trainer to outsource the training process. The adversary then trains a model with the received normal data as well as images containing a backdoor trigger, e.g., a sticker with a flower. The goal of the adversary here is to lead the model to misclassify when the backdoor-triggering input is provided.

To mitigate backdoor attacks, researchers have proposed several mitigation methodologies [9], [32], [53]. DeepInspect [9] reverse-engineers the backdoor trigger using a conditional generative model and then fine-tunes the target model by harnessing the generated backdoor-triggering images and their correct labels. Wang *et al.* [53] suggested another method that remedies the target model by removing neurons that contribute to misclassifying backdoor-triggering images.

Note that these defenses are similar to our adaptive fine-tuning attacks and adaptive pruning attacks *per se*. However, their approaches are not directly applicable to reverse-engineering key images of various trigger set-based DNN watermarking algorithms because they only focus on backdoor-triggering inputs created by adding a backdoor trigger to the source images. On the other hand, we demonstrated how an adaptive adversary generates key images against diverse watermarking schemes.

Adversarial example attacks. DNN models are known to misclassify adversarial examples created by adding quasi-imperceptible perturbations to normal examples [18], [47]. Since this finding, there has been a vast volume of research on adversarial examples. To mitigate this threat, Papernot *et al.* [39] proposed defensive distillation to smooth the network gradients exploited for generating adversarial samples. On the other hand, MagNet [34] detects such examples at the testing phase; it detects and reforms adversarial examples by leveraging autoencoders trained on regular images. However, these defenses were later broken by other strong attacks [6], [7], [8]. Adversarial training [44], which improves the robustness of DNN models by training adversarial examples with correct labels, is the current state-of-the-art defense against adversarial example attacks [33]. In our study, we selected WM_{adv} that utilizes adversarial examples as our target watermarking scheme and employed the approach of MagNet [34] for evasion attacks.

Model stealing attacks. The goal of model stealing attacks, also known as model extraction attacks, is to copy the classification performance of remote target models [50]. Papernot *et al.* [38] trained a counterfeit model as a stepping stone for creating adversarial examples of remote target models. Orekondy *et al.* [37] demonstrated that model stealing is still possible against complex DNN models even though the adversary does not have enough training sets and does not know the model structure. They showed that arbitrary images downloaded from the Internet and arbitrary models are enough to forge the target model. PRADA [26] detects model stealing attempts by analyzing incoming queries. However, this defense is inapplicable to DNN watermarking algorithms. Note that the adversary does not have to send remote queries because the target model is already in the hands of the adversary. We leveraged this attack for removing $\mathcal{O}$'s watermark in the target model. In our settings, we prepared the training set for model stealing attacks in §7.3.2 following the approach of [37].

10 Conclusion

We investigate the current practice of demonstrating watermark robustness via adversarial evaluation in the previous studies. We point out two common flaws in their evaluations: (1) incomplete adversarial evaluation and (2) overlooked adaptive attacks. Taking into account these shortcomings, we evaluate the 10 trigger set-based watermarking schemes and demonstrate that every proposed watermarking scheme is vulnerable to at least five presented attacks, which significantly undermines their intended goal of proving ownership. We conclude these failures stem from today's flawed practice in conducting adversarial evaluation. We encourage future studies on new watermarking algorithms to consider our guidelines presented herein to demonstrate a meaningful upper bound of robustness against the complete set of the existing attacks, including the proposed adaptive attacks.

Acknowledgments

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2020-0-00153, Penetration Security Testing of ML Model Vulnerabilities and Defense).

REFERENCES

[1] André Adelsbach, Stefan Katzenbeisser, and Helmut Veith. Watermarking schemes provably secure against copy and ambiguity attacks. In *Proceedings of the ACM Workshop on Digital Rights Management*, pages 111–119, 2003.

[2] Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *Proceedings of the USENIX Security Symposium*, pages 1615–1631, 2018.

[3] Osbert Bastani, Yani Ioannou, Leonidas Lampropoulos, Dimitrios Vytiniotis, Aditya V. Nori, and Antonio Criminisi. Measuring neural net robustness with constraints. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 2621–2629, 2016.

[4] Xiaoyu Cao, Minghong Fang, Jia Liu, and Neil Zhenqiang Gong. FLTrust: Byzantine-robust federated learning via trust bootstrapping. In *Proceedings of the Network and Distributed System Security Symposium*, 2021.

[5] Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, Aleksander Madry, and Alexey Kurakin. On evaluating adversarial robustness. *CoRR*, abs/1902.06705, 2019.

[6] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the ACM Workshop on Artificial Intelligence and Security*, pages 3–14, 2017.

[7] Nicholas Carlini and David Wagner. MagNet and "efficient defenses against adversarial attacks" are not robust to adversarial examples. *CoRR*, abs/1711.08478, 2017.

[8] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 39–57, 2017.

[9] Huili Chen, Cheng Fu, Jishen Zhao, and Farinaz Koushanfar. DeepInspect: A black-box trojan detection and mitigation framework for deep neural networks. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 4658–4664, 2019.

[10] Huili Chen, Bita Darvish Rouhani, Cheng Fu, Jishen Zhao, and Farinaz Koushanfar. DeepMarks: A secure fingerprinting framework for digital rights management of deep learning models. In *Proceedings of the ACM International Conference on Multimedia Retrieval*, pages 105–113, 2019.

[11] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *CoRR*, abs/1712.05526, 2017.

[12] Xinyun Chen, Wenxiao Wang, Chris Bender, Yiming Ding, Ruoxi Jia, Bo Li, and Dawn Song. REFIT: a unified watermark removal framework for deep learning systems with limited data. In *Proceedings of the ACM Asia Conference on Computer and Communications Security*, pages 321–335, 2021.

[13] Yizheng Chen, Shiqi Wang, Dongdong She, and Suman Jana. On training robust PDF malware classifiers. In *Proceedings of the USENIX Security Symposium*, pages 2343–2360, 2020.

[14] Anna Choromanska, Mikael Henaff, Michael Mathieu, Gerard Ben Arous, and Yann LeCun. The loss surfaces of multilayer networks. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 192–204, 2015.

[15] Scott Craver, Nasir Memon, Boon-Lock Yeo, and Minerva M. Yeung. Resolving rightful ownerships with invisible watermarking techniques: Limitations, attacks, and implications. *IEEE Journal on Selected Areas in Communications*, 16(4):573–586, 1998.

[16] Lixin Fan, Kam Woh Ng, and Chee Seng Chan. Rethinking deep neural network ownership verification: Embedding passports to defeat ambiguity attacks. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 4716–4725, 2019.

[17] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the ACM Conference on Computer and Communications Security*, pages 1322–1333, 2015.

[18] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *Proceedings of the International Conference on Learning Representations*, 2015.

[19] Albert Gordo, Jon Almazán, Jerome Revaud, and Diane Larlus. Deep image retrieval: Learning global representations for image search. In *Proceedings of the European Conference on Computer Vision*, pages 241–257, 2016.

[20] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. BadNets: Identifying vulnerabilities in the machine learning model supply chain. *CoRR*, abs/1708.06733, 2019.

[21] Jia Guo and Miodrag Potkonjak. Watermarking deep neural networks for embedded systems. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 1–8, 2018.

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[23] Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. Safety verification of deep neural networks. In *Proceedings of the International Conference on Computer Aided Verification*, pages 3–29, 2017.

[24] Hengrui Jia, Christopher A. Choquette-Choo, Varun Chandrasekaran, and Nicolas Papernot. Entangled watermarks as a defense against model extraction. In *Proceedings of the USENIX Security Symposium*, pages 1937–1954, 2021.

[25] Jinyuan Jia, Ahmed Salem, Michael Backes, Yang Zhang, and Neil Zhenqiang Gong. MemGuard: Defending against black-box membership inference attacks via adversarial examples. In *Proceedings of the ACM Conference on Computer and Communications Security*, pages 259–274, 2019.

[26] Mika Juuti, Sebastian Szyller, Samuel Marchal, and N. Asokan. PRADA: Protecting against DNN model stealing attacks. In *Proceedings of the IEEE European Symposium on Security and Privacy*, pages 512–527, 2019.

[27] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*, 2015.

[28] Gerhard C. Langelaar, Iwan Setyawan, and Reginald L. Lagendijk. Watermarking digital image and video data. a state-of-the-art overview. *IEEE Signal Processing Magazine*, 17(5):20–46, 2000.

[29] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

[30] Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. Neural attention distillation: Erasing backdoor triggers from deep neural networks. In *Proceedings of the International Conference on Learning Representations*, 2021.

[31] Zheng Li, Chengyu Hu, Yang Zhang, and Shanqing Guo. How to prove your model belongs to you: A blind-watermark based framework to protect intellectual property of DNN. In *Proceedings of the Annual Computer Security Applications Conference*, pages 126–137, 2019.

[32] Yingqi Liu, Wen-Chuan Lee, Guanhong Tao, Shiqing Ma, Yousra Aafer, and Xiangyu Zhang. ABS: Scanning neural networks for back-doors by artificial brain stimulation. In *Proceedings of the ACM Conference on Computer and Communications Security*, pages 1265–1282, 2019.

[33] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *Proceedings of the International Conference on Learning Representations*, 2018.

[34] Dongyu Meng and Hao Chen. MagNet: a two-pronged defense against adversarial examples. In *Proceedings of the ACM Conference on Computer and Communications Security*, pages 135–147, 2017.

[35] Erwan Le Merrer, Patrick Pérez, and Gilles Trédan. Adversarial frontier stitching for remote neural network watermarking. *Neural Computing and Applications*, 2019.

[36] Ryota Namba and Jun Sakuma. Robust watermarking of neural network with exponential weighting. In *Proceedings of the ACM Asia Conference on Computer and Communications Security*, pages 228–240, 2019.

[37] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Knockoff nets: Stealing functionality of black-box models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4954–4963, 2019.

[38] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the ACM Asia Conference on Computer and Communications Security*, pages 506–519, 2017.

[39] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 582–597, 2016.

[40] Erwin Quiring, David Klein, Daniel Arp, Martin Johns, and Konrad Rieck. Adversarial preprocessing: Understanding and pre-

venting image-scaling attacks in machine learning. In *Proceedings of the USENIX Security Symposium*, pages 1363–1380, 2020.

[41] Bita Darvish Rouhani, Huili Chen, and Farinaz Koushanfar. DeepSigns: An end-to-end watermarking framework for ownership protection of deep neural networks. In *Proceedings of the ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 485–497, 2019.

[42] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2019.

[43] Masoumeh Shafieinejad, Nils Lukas, Jiaqi Wang, Xinda Li, and Florian Kerschbaum. On the robustness of the backdoor-based watermarking in deep neural networks. In *Proceedings of the ACM Workshop on Information Hiding and Multimedia Security*, pages 177–188, 2021.

[44] Uri Shaham, Yutaro Yamada, and Sahand Negahban. Understanding adversarial training: Increasing local stability of neural nets through robust optimization. *CoRR*, abs/1511.05432, 2016.

[45] Shawn Shan, Emily Wenger, Bolun Wang, Bo Li, Haitao Zheng, and Ben Y. Zhao. Gotta catch'em all: Using honeypots to catch adversarial attacks on neural networks. In *Proceedings of the ACM Conference on Computer and Communications Security*, pages 67–83, 2020.

[46] Mitchell D. Swanson, Mei Kobayashi, and Ahmed H. Tewfik. Multimedia data-embedding and watermarking technologies. *Proceedings of the IEEE*, 86(6):1064–1087, 1998.

[47] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *Proceedings of the International Conference on Learning Representations*, 2014.

[48] Mingxing Tan and Quoc V. Le. EfficientNetV2: Smaller models and faster training. In *Proceedings of the International Conference on Machine Learning*, pages 10096–10106, 2021.

[49] Yuchi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. DeepTest: Automated testing of deep-neural-network-driven autonomous cars. In *Proceedings of the International Conference on Software Engineering*, pages 303–314, 2018.

[50] Florian Tramér, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction APIs. In *Proceedings of the USENIX Security Symposium*, pages 601–618, 2016.

[51] Yusuke Uchida, Yuki Nagai, Shigeyuki Sakazawa, and Shin'ichi Satoh. Embedding watermarks into deep neural networks. In *Proceedings of the ACM International Conference on Multimedia Retrieval*, pages 269–277, 2017.

[52] Ji Wan, Dayong Wang, Steven C.H. Hoi, Pengcheng Wu, Jinake Zhu, Yongdong Zhang, and Jintao Li. Deep learning for content-based image retrieval: A comprehensive study. In *Proceedings of the ACM International Conference on Multimedia*, pages 157–166, 2016.

[53] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y. Zhao. Neural Cleanse: Identifying and mitigating backdoor attacks in neural networks. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 707–723, 2019.

[54] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747, 2017.

[55] Xiaojun Xu, Chang Liu, Qian Feng, Heng Yin, Le Song, and Dawn Song. Neural network-based graph embedding for cross-platform binary code similarity detection. In *Proceedings of the ACM Conference on Computer and Communications Security*, pages 363–376, 2017.

[56] Guixin Ye, Zhanyong Tang, Dingyi Fang, Zhanxing Zhu, Yansong Feng, Pengfei Xu, Xiaojiang Chen, and Zheng Wang. Yet another text captcha solver: A generative adversarial network based approach. In *Proceedings of the ACM Conference on Computer and Communications Security*, pages 332–348, 2018.

[57] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *Proceedings of the International Conference on Learning Representations*, 2017.

[58] Jialong Zhang, Zhongshu Gu, Jiyong Jang, Hui Wu, Marc Ph. Stoecklin, Heqing Huang, and Ian Molloy. Protecting intellectual property of deep neural networks with watermarking. In *Proceedings of the ACM Asia Conference on Computer and Communications Security*, pages 159–172, 2018.

[59] Jie Zhang, Dongdong Chen, Jing Liao, Han Fang, Weiming Zhang, Wenbo Zhou, Hao Cui, and Nenghai Yu. Model watermarking for image processing networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

Suyoung Lee is a Ph.D. student at KAIST. His primary research interests lie in the area of finding, analyzing, and patching software vulnerabilities. He has participated in building several tools that automatically find vulnerabilities in large software, such as web browsers and web applications.

Wonho Song is a Master's student at KAIST. His research is mainly focused on securing machine learning systems.

Suman Jana is an associate professor in the department of computer science at Columbia University. His primary research interests are at the intersection of computer security and machine learning. His research has won six best paper awards including one at the SOSP 2017 and two at the IEEE S&P 2014 and 2016. His work has led to reporting and fixing of around 250 high-impact security vulnerabilities across a wide range of software.

Meeyoung Cha is an associate professor of School of Computing at KAIST. Her research is on data science and information science with an emphasis on modeling socially-relevant information propagation processes. Her work, on misinformation, poverty mapping, fraud detection, and long-tail content, has gained more than 14,000 citations and received the best paper awards at several conferences.

Sooel Son is an associate professor of School of Computing at KAIST. He received his Ph.D. in the department of computer science at the University of Texas at Austin. He is working on various topics regarding web security and privacy. He received a best student paper award at NDSS 2013.