Formal Foundations of Trigger-Based Watermarking

Stefano Calzavara
Università Ca' Foscari Venezia
stefano.calzavara@unive.it

Lorenzo Cazzaro
University of Luxembourg
lorenzo.cazzaro@uni.lu

Claudio Lucchese, Salvatore Orlando
Università Ca' Foscari Venezia
{claudio.lucchese,orlando}@unive.it

***Abstract*—Trigger-based watermarking aims to protect the intellectual property of machine learning models by embedding abnormal behavior that is activated only on specific trigger inputs, while preserving performance on standard test data. Although widely studied, current approaches to trigger-based watermarking remain largely informal and empirical. In this work, we introduce a formal framework for reasoning about the security of trigger-based watermarking, and we propose rigorous security definitions to evaluate existing schemes. Our analysis reveals that many schemes rely on underspecified parameters and flawed design assumptions. Notably, we demonstrate that different security goals in watermarking can be in tension with one another, and achieving a balance among them requires careful design. Our principled analysis offers a way to resolve this tension in practice. Experiments on existing schemes and public datasets corroborate the relevance of our theoretical findings.**

***Index Terms*—watermarking, machine learning, security**

I. Introduction

Effective machine learning models are valuable assets, because model training can be computationally expensive, possibly requiring weeks of work even on dedicated hardware. Moreover, the creation of high-quality datasets can be a major challenge by itself, requiring several person-months across data collection, cleaning and labeling. As a matter of fact, once an effective machine learning model has been trained, protecting its intellectual property is of paramount importance to the legitimate model owner. The research community proposed *model watermarking* as an effective solution to support model ownership claims [1], [2], [3]. Within the rich field of model watermarking, *trigger-based watermarking* [4], [5], [6], [7], [8] has emerged as a particularly popular approach due to its black-box nature, i.e., the model owner does not have to access model internals, but can make model ownership claims by checking the model output alone. The key intuition of trigger-based watermarking is straightforward: the model is trained so as to exhibit a peculiar behavior over a subset of the data, called the *trigger set*. For example, an image classifier may be trained over a trigger set including a specific visual pattern, e.g., a blurry area of pixels in a corner, designed to induce prediction errors. In this case, any image containing this pattern may be incorrectly classified as the image of a cat, thus evoking an unexpected behavior specific to the watermarked model, which allows the model owner to claim their rights.

The research community proposed a wide array of trigger-based watermarking schemes, differing in terms of the nature of the watermark (a visual pattern, a secret signature, etc.), the details of watermark embedding (training set modifications, design of custom training algorithms, etc.) and underlying intuitions. At a high level, however, watermark verification stays the same: the legitimate model owner supplies the model with the trigger set and verifies compliance with respect to the peculiar behavior that the watermarked model is required to evoke. Unfortunately, understanding the security properties of different trigger-based watermarking schemes is challenging, because most work in the field is empirical in nature. There is no unified framework defining formal security properties, nor any effective way to reason about such properties. This is problematic, because recent work showed that trigger-based watermarking can be attacked [9], [10], [11]. In this work, we fill this gap by performing the first formal security analysis of trigger-based watermarking, which allows us to identify the foundations underpinning this line of work and enable principled reasoning about its security guarantees.

Contributions: We make the following contributions:

1) We present a formal framework that uniformly captures existing trigger-based watermarking schemes from the literature. Based on our framework, we introduce formal security definitions capturing important requirements of trigger-based watermarking and ensuring protection against relevant attacks (Section III).
2) We use our formal framework to reason about the security of specific design choices adopted by existing trigger-based watermarking schemes. By doing this, we prove the insecurity of past empirical solutions and we identify principled techniques to securely deploy trigger-based watermarking (Section IV).
3) We use selected trigger-based watermarking schemes from the literature to train watermarked models over public datasets and we apply our framework to show that the empirical parameter choice advocated in prior work is inappropriate, because it leaves room for attacks. We then show that the use of our principled techniques enables a sound parameter choice, which may enforce security against different attackers (Section V).

II. Background

We first review some important prerequisite concepts.

A. Supervised Learning

Let $\mathcal{X}$ be a d-dimensional vector space of *features* from the set $\{1, \ldots, d\}$. An *instance* $\vec{x} \in \mathcal{X}$ is a d-dimensional vector $\langle x_1, x_2, \ldots, x_d \rangle$ representing an object in the feature space $\mathcal{X}$. Each instance $\vec{x} \in \mathcal{X}$ is assigned its correct class label

$y \in \mathcal{Y}$ by a *target* function $\tau : \mathcal{X} \to \mathcal{Y}$. The target function is generally unknown, except for its value on specific sets of instances (*datasets*) for which the correct class is known, e.g., as the result of manual labeling by human experts.

A supervised learning algorithm for a classification task automatically learns a *classifier* (or *model*) $M : \mathcal{X} \to \mathcal{Y}$ from a *training set* of correctly labeled instances $\mathcal{D}_{train} = \{\langle \vec{x}_i, \tau(\vec{x}_i)\rangle\}_i$, with the goal of approximating the target function τ. The performance of classifiers is normally estimated on a *test set* of correctly labeled instances $\mathcal{D}_{test} = \{\langle \vec{z}_i, \tau(\vec{z}_i)\rangle\}_i$, disjoint from the training set, yet drawn from the same data distribution (noted $\mathcal{D}_{test} \sim \mathcal{D}_{train}$). For example, the standard *accuracy* measure $a(M, \mathcal{D}_{test})$ counts the percentage of test instances where the classifier M returns a correct prediction.

B. Trigger-Based Watermarking

Watermarking schemes for intellectual property protection allow a legitimate model owner to prove the ownership of a machine learning model unduly operated by an attacker, e.g., because the attacker managed to steal the model from the owner's machine [1]. We focus on *trigger-based* watermarking schemes, where watermark creation forces the model to exhibit an abnormal and unique behavior on a set of instances called the *trigger set*. Watermark verification is *black-box*, i.e., the legitimate model owner may prove ownership by means of queries to the model and has no access to the model internals.

Formally, a trigger set is a collection of *incorrectly* labeled instances $\mathcal{D}_{trigger} = \{\langle \vec{x}_i, y_i\rangle \mid y_i \neq \tau(\vec{x}_i)\}_i$, where the watermarked model is required to show high accuracy. In other words, "correct" predictions on the trigger set correspond to specific types of prediction errors. High accuracy on the trigger set is abnormal and specific to the watermarked model, because machine learning models are trained to optimize accuracy, hence, for any instance $\langle \vec{x}_i, y_i\rangle \in \mathcal{D}_{trigger}$, a non-watermarked model is much more likely to return the correct label $\tau(\vec{x}_i)$ than the incorrect label $y_i \neq \tau(\vec{x}_i)$.

Trigger-based watermarking schemes are defined in terms of three main algorithms:

1) The trigger set creation algorithm $\textsc{Trigger}(\mathcal{D}_{train}, \omega)$ takes as input a training set $\mathcal{D}_{train}$ and a *watermark* ω to produce a trigger set $\mathcal{D}_{trigger}$. The watermark uniquely identifies the model owner by defining the abnormal behavior that the model should expose on $\mathcal{D}_{trigger}$ to provide a proof of ownership, e.g., the model incorrectly classifies cats as dogs. Different watermarking schemes rely on different forms of ω, define different trigger set construction algorithms, and may use $\mathcal{D}_{train}$ differently.
2) The *embedding* algorithm $\textsc{Embed}(\mathcal{D}_{train}, \omega)$ takes as input a training set $\mathcal{D}_{train}$ and a watermark ω to learn a watermarked model $M : \mathcal{X} \to \mathcal{Y}$. Concretely, the algorithm first invokes $\textsc{Trigger}(\mathcal{D}_{train}, \omega)$ to construct a trigger set $\mathcal{D}_{trigger}$ and then trains the model M over $\mathcal{D}_{train} \cup \mathcal{D}_{trigger}$ (or a subset of it). Specific details of model training may vary across different watermarking schemes, but the high-level intuition of the embedding algorithm stays the same. The algorithm finally returns the watermarked model M.
3) The *verification* algorithm $\textsc{Verify}(M, \mathcal{D}_{test}, \mathcal{D}_{trigger})$ takes as input a model M, a test set $\mathcal{D}_{test}$ (possibly empty) and a trigger set $\mathcal{D}_{trigger}$ to determine whether M contains the watermark or not. Note that the trigger set used for verification purposes does not necessarily coincide with the trigger set used in the watermark embedding phase, e.g., $\mathcal{D}_{trigger}$ may be generated using instances from $\mathcal{D}_{test}$. Intuitively, a watermarked model should classify instances from $\mathcal{D}_{test}$ to their correct labels and instances from $\mathcal{D}_{trigger}$ to their expected labels as required by the watermarking scheme, thus achieving high accuracy on both $\mathcal{D}_{test}$ and $\mathcal{D}_{trigger}$.

Note that, although quite general, this abstraction of trigger-based watermarking does not cover all the existing schemes in the literature. For example, it does not support schemes where model ownership is proved by leveraging explanation or interpretability signals rather than model outputs [12].

III. Formal Framework

We here propose our formal framework for the security evaluation of trigger-based watermarking schemes, which is based on a generalized version of existing watermarking verification algorithms proposed in prior work.

A. Modeling Watermark Verification

Our formal model specifically focuses on the inner workings of the watermark verification algorithm, which are difficult to appreciate based on the informal treatment of previous work. Most watermarking schemes just query a model M over instances of $\mathcal{D}_{trigger}$ and assume the watermark to be there if and only if the number of correct predictions exceeds an empirically set threshold [8], [6], [13]. The use of $\mathcal{D}_{test}$ in addition to $\mathcal{D}_{trigger}$ is specific to the watermarking scheme by Guo and Potkonjak [4]. In their scheme, the watermarked model is required to show high accuracy on both $\mathcal{D}_{test}$ and $\mathcal{D}_{trigger}$, but the paper never clarifies the rationale and security impact of this design choice. After all, it is the abnormal behavior on the trigger set that enables watermarking and a high accuracy on $\mathcal{D}_{test}$ is not specific to watermarked models.

We here motivate a possible use of $\mathcal{D}_{test}$ and we discuss how we reflect it in our formal model of the watermark verification algorithm, based on the following rationale underlying trigger-based watermarking schemes:

1) If M has high accuracy on both $\mathcal{D}_{test}$ and $\mathcal{D}_{trigger}$, then M is likely to be the watermarked model, hence the verification process must return **True**.
2) If M has high accuracy on $\mathcal{D}_{test}$, but low accuracy on $\mathcal{D}_{trigger}$, then M is unlikely to be the watermarked model, hence the verification process must return **False**.
3) If M has low accuracy on $\mathcal{D}_{test}$, we are in a controversial case, because both watermarked and non-watermarked models are expected to show high accuracy on test instances. In this case, either somebody has unexpectedly claimed ownership of a poor-performing

non-watermarked model, or the attacker has actively altered the predictions of a watermarked model to evade verification. Hence, we assume that verification is inconclusive and we require the verification algorithm to return the special value $\perp$.

Our treatment differs from previous watermarking schemes, because we are not aware of any work proposing the use of three-valued logic for watermark verification. However, our interpretation is consistent with the watermarking scheme of Guo and Potkonjak [4], because a low accuracy on $\mathcal{D}_{test}$ leads to a failure in watermark verification as they propose. The choice of using $\perp$ rather than **False** for this type of failures clarifies that such cases are unexpected and fundamentally different from cases where the model under verification shows high accuracy on $\mathcal{D}_{test}$, but low accuracy on $\mathcal{D}_{trigger}$, meaning that the watermark is likely not present. This choice enables subtler forms of reasoning in our formal definitions.

There is another delicate aspect to deal with to design a general model of watermark verification. Most watermarking schemes do not assume that the model owner commits to a specific choice of $\mathcal{D}_{test}$ and $\mathcal{D}_{trigger}$ for verification purposes, but any other datasets from the same data distribution may be used for watermark verification. For example, some watermarking schemes are based on specific perturbations of test instances leading to abnormal model predictions, e.g., the application of a visual pattern to images [14]. The application of these perturbations allows the construction of arbitrary new trigger sets enabling watermark verification with high probability.

To support these cases, we model watermark verification as a *probabilistic* algorithm sampling new datasets $\mathcal{D}^*_{test} \sim \mathcal{D}_{test}$ and $\mathcal{D}^*_{trigger} \sim \mathcal{D}_{trigger}$ for verification purposes. We model this probabilistic behavior by means of an auxiliary function $\textsc{Challenge}(M, \mathcal{D})$ that, given a watermarked model M and a dataset $\mathcal{D}$, returns a pair $\langle \mathcal{D}^*, N^* \rangle$, where $\mathcal{D}^* \sim \mathcal{D}$ is a dataset coming from a sampling process ensuring that the probability that M performs a correct prediction on $\mathcal{D}^*$ is equal to $a(M, \mathcal{D})$ and $N^* \leq |\mathcal{D}^*|$ defines the number of correct predictions that M should perform on $\mathcal{D}^*$ for a successful watermark verification. The challenge function is a modeling tool to represent that verification may be performed on datasets different from $\mathcal{D}_{test}$ and $\mathcal{D}_{trigger}$, yet drawn from the same data distributions, where the model is expected to show comparable performance.

In the end, our model of the verification algorithm $\textsc{Verify}(M, \mathcal{D}_{test}, \mathcal{D}_{trigger})$ first builds challenges $\langle \mathcal{D}^*_{test}, N \rangle$ and $\langle \mathcal{D}^*_{trigger}, N' \rangle$ from $\mathcal{D}_{test}$ and $\mathcal{D}_{trigger}$ respectively. The model M is then used to classify all the instances in $\mathcal{D}_{verify} = \mathcal{D}^*_{test} \cup \mathcal{D}^*_{trigger}$: if the number of correctly classified test instances exceeds the threshold N, watermark verification succeeds if and only if the number of trigger instances classified to their expected label as required by the watermarking scheme exceeds the threshold N'. Pseudocode of our generalized watermark verification algorithm summarizing the present discussion is shown in Algorithm 1. Appendix A shows how different trigger-based watermarking schemes can be encoded in our framework, confirming its generality.

Algorithm 1 Watermark verification algorithm

function $\textsc{Verify}(M, \mathcal{D}_{test}, \mathcal{D}_{trigger})$
  $\langle \mathcal{D}^*_{test}, N \rangle \leftarrow \textsc{Challenge}(M, \mathcal{D}_{test})$
  $\langle \mathcal{D}^*_{trigger}, N' \rangle \leftarrow \textsc{Challenge}(M, \mathcal{D}_{trigger})$
  return $\textsc{Claim}(M, \mathcal{D}^*_{test}, N, \mathcal{D}^*_{trigger}, N')$
end function

function $\textsc{Claim}(M, \mathcal{D}^*_{test}, N, \mathcal{D}^*_{trigger}, N')$
  $n \leftarrow 0$
  $n' \leftarrow 0$
  $\mathcal{D}_{verify} \leftarrow \mathcal{D}^*_{test} \cup \mathcal{D}^*_{trigger}$
  for $\langle \vec{x}, y \rangle \in \mathcal{D}_{verify}$ **do**
    if $M(\vec{x}) = y$ **then**
      if $\langle \vec{x}, y \rangle \in \mathcal{D}^*_{test}$ **then**
        $n \leftarrow n + 1$
      else
        $n' \leftarrow n' + 1$
      end if
    end if
  end for
  if $n > N$ **then**
    return $n' > N'$
  else
    return $\perp$
  end if
end function

B. Threat Model

We assume that the attacker may have illegitimate access to the original watermarked model M, e.g., because they have been able to steal it from the owner, or to some variant of M, e.g., obtained through model extraction [15], [16], [17] or fine-tuning of the original model [18], [19], [11]. However, we assume that the attacker does not know the watermark ω and the original trigger set $\mathcal{D}_{trigger}$.

We assume that the attacker unduly operates the watermarked model to offer services leveraging its classification power to third parties, e.g., in exchange of economic incentives, hence the legitimate model owner may be aware of this inappropriate use and claim model ownership by watermark verification. However, the attacker may be able to detect watermark verification attempts and may actively try to make model ownership claims fail. This conservative assumption is important to reason about the security of the watermarking scheme in the adversarial setting where the attacker unduly operates the watermarked model (or some variant of it).

Finally, we suppose that the attacker has knowledge of a representative number of correctly labeled test instances. For simplicity, we just assume that the attacker knows the original test set $\mathcal{D}_{test}$, which is expected to be a good proxy of the actual test data within the standard machine learning pipeline. Note that a significant amount of test data may be publicly available online, e.g., through standard machine learning repositories like UCI or Kaggle.

C. Security Properties

Existing trigger-based watermarking schemes make different design choices, which are often informally motivated and just empirically evaluated in practice. In this work, we are concerned about the fundamental security impact of specific design choices, hence we introduce formal security definitions. Our definitions are probabilistic in nature and we use $\varepsilon \in [0, 1]$ to denote the probability that the attacker successfully breaks security, i.e., smaller values of ε provide better security guarantees. We design our definitions to capture fundamental properties that any trigger-based watermarking scheme should satisfy, rather than addressing specific shortcomings of individual schemes from the literature.

1) Verifiability: The first property we introduce is called *verifiability* and enforces a lower bound on the probability that watermark verification succeeds when it is performed with the correct $\mathcal{D}_{test}$ and $\mathcal{D}_{trigger}$. Verifiability is important because a non-verifiable watermarked model may not allow the legitimate model owner to claim intellectual property, thus rendering the watermark useless in practice.

Definition 1 (Verifiability)**.** Let $\mathcal{D}_{train}$, $\mathcal{D}_{test}$, $\mathcal{D}_{trigger}$, ω be the datasets and watermark of the model owner respectively and let $M = \text{EMBED}(\mathcal{D}_{train}, \omega)$. For a fixed $0 \leq \varepsilon \leq 1$, we say that M is *ε-verifiable* if and only if the probability that $\text{VERIFY}(M, \mathcal{D}_{test}, \mathcal{D}_{trigger})$ returns **True** is at least $1 - \varepsilon$.

The definition does not predicate over the attacker, because ensuring verifiability is just responsibility of the model owner based on the model accuracy on $\mathcal{D}_{test}$ and $\mathcal{D}_{trigger}$. Note that verifiability assumes that the model owner does not consider $\perp$ as an acceptable output of watermark verification, because the definition requires maximizing the probability that the output is **True**, i.e., the model owner wants to ensure that $\mathcal{D}_{test}$ and $\mathcal{D}_{trigger}$ are appropriate to prove model ownership.

2) Robust Verifiability: The second property we introduce is called *robust verifiability* and is an extension of verifiability to the adversarial setting, where the watermarked model (or a variant of it) is unduly operated by the attacker as the result of an intellectual property violation. Since the attacker unduly operates (a variant of) the watermarked model M, they may adapt their behavior to evade watermark verification. Since watermark verification operates by submitting instances to the model and checking the returned class predictions, we formalize the attacker in terms of a probabilistic function $A_M : \mathcal{X} \to \mathcal{Y}$ in charge of returning class predictions for the instances supplied during verification. The subscript M denotes that the function can query M in its definition. Since the attacker is modeled as a probabilistic function (in the mathematical sense), its behavior is inherently stateless, which simplifies the underlying probabilistic analysis, because all the inputs of the function A_M are treated as independent events. Extending the technical treatment to stateful adversaries would not invalidate the formal framework, but would require more involved mathematical machinery, meriting a dedicated investigation that we leave to future work.

The intuition underlying robust verifiability is simple. In our adversarial setting, the watermark verification procedure invoked by the model owner does not interact with M but with A_M, and robust verifiability enforces an upper bound on the probability that watermark verification fails despite the interaction with the attacker.

Definition 2 (Robust Verifiability)**.** Let $\mathcal{D}_{train}$, $\mathcal{D}_{test}$, $\mathcal{D}_{trigger}$, ω be the datasets and watermark of the model owner respectively and let $M = \text{EMBED}(\mathcal{D}_{train}, \omega)$. For a fixed $0 \leq \varepsilon \leq 1$, we say that M is *robustly ε-verifiable* against the attacker A_M if and only if the probability that $\text{VERIFY}(A_M, \mathcal{D}_{test}, \mathcal{D}_{trigger})$ returns **False** is at most ε.

Note that robust verifiability assumes that the model owner may consider $\perp$ as an acceptable output of watermark verification, because the definition requires minimizing the probability that the output is **False**. In other words, the model owner wants to be sure that the attacker cannot conclusively evade their legitimate ownership claims.

To better clarify the real-world implications of robust verifiability, we now provide two examples showing that our simple formalism is already expressive enough to capture different relevant threats.

Example 1 (Watermark Suppression)**.** Robust verifiability naturally formalizes security against *watermark suppression*, i.e., the informal property that the attacker should be unable to evade watermark verification by suppressing the unusual behavior evoked by the trigger set [1], [20], [8], [9]. Assume that the attacker operates a stolen watermarked model M and is aware of the risks of being detected through watermark verification. In this case, the attacker may actively try to make watermark verification fail. In particular, for any $\langle \vec{x}, y \rangle \in \mathcal{D}_{verify}$, the inner workings of watermark verification (Algorithm 1) substantiate the following observations:

- If $\langle \vec{x}, y \rangle \in \mathcal{D}^*_{test}$, then $y = \tau(\vec{x})$. In this case, the attacker's goal is returning y, because this increases the counter n and maximizes the probability that the verification algorithm does not return $\perp$. Since watermarked models are normally valuable assets providing high accuracy on the test set, it is likely that $M(\vec{x}) = y$, hence an advantageous action may be returning $M(\vec{x})$.
- If $\langle \vec{x}, y \rangle \in \mathcal{D}^*_{trigger}$, then $y \neq \tau(\vec{x})$. In this case, the attacker's goal is returning some $y' \neq y$, because this does not increase the counter n' and maximizes the probability that the verification algorithm returns **False**. Since the attacker does not know the label y, because the trigger set is secret, an advantageous action may be just returning a random label y'.

Based on this analysis, we can mitigate watermark suppression by requiring robust verifiability against the attacker $A_M(\vec{x})$ operating as follows:

1) The attacker tries to guess whether $\vec{x}$ is a test instance or a trigger instance, with probability of success $q \geq 0.5$.
2) If the attacker believes that $\vec{x}$ is a test instance, they return $M(\vec{x})$.

3) If the attacker believes that $\vec{x}$ is a trigger instance, they return a random label.

Example 2 (Model Extraction)**.** Real-world attackers do not necessarily steal a watermarked model M, but they may rather learn a new model offering performance similar to M by using techniques like *model extraction* [15], [16], [17]. In this case, the attacker may not actively try to make watermark verification fail, but rather the extracted model may differ from the original model (especially on the trigger set) to the point that its output does not enable watermark verification anymore. We can reason about this scenario in our formal model by requiring robust verifiability against the following attacker $A_M(\vec{x})$, which formalizes the variant of M obtained through model extraction:

1) If $\vec{x}$ is a test instance, then A_M returns $M(\vec{x})$ with probability q_1 and a random label with probability $1-q_1$.
2) If $\vec{x}$ is a trigger instance, then A_M returns $M(\vec{x})$ with probability q_2 and a random label with probability $1-q_2$.

The specific values of q_1 and q_2 model how much the extraction process preserved the accuracy of the original model M on the test set and the trigger set respectively. An ideal model extraction attack would have $q_1 \approx 1$ and $q_2 \approx 0$, leading to a model preserving the accuracy of M on test instances, while forgetting the intended behavior on trigger instances.

We observe that our treatment of model extraction attacks is rather general in light of its abstract nature. Indeed, we are not concerned with how the attack is carried out: we just assume that the attacker operates a variant of the watermarked model that might mask its abnormal behavior on the trigger set. The same mathematical treatment thus naturally captures model extraction, fine-tuning, parameter pruning or other types of model changes that may affect watermark verification [9].

3) Unforgeability: The last property we introduce is called *unforgeability* and concerns security against *watermark forgery*, i.e., the informal property that the attacker should be unable to forge a valid trigger set exposing some peculiar model behavior enabling illicit ownership claims [1], [9], [4], [21], [6], [22]. In other words, the attacker may unduly claim ownership of a watermarked model M by executing watermark verification with a forged trigger set $\mathcal{D}'_{trigger}$ in place of the original trigger set $\mathcal{D}_{trigger}$ of the legitimate owner. At the very least, this would create ambiguous situations where it is hard to discriminate between the real model owner and the attacker. Unforgeability enforces an upper bound on the probability that watermark verification succeeds when it is performed with the original test set $\mathcal{D}_{test}$ (which we assume to be known by the attacker) and the forged trigger set $\mathcal{D}'_{trigger}$.[1]

Definition 3 (Unforgeability)**.** Let $\mathcal{D}_{train}$, $\mathcal{D}_{trigger}$, $\mathcal{D}_{test}$, ω be the datasets and watermark of the model owner respectively and let $M = \text{EMBED}(\mathcal{D}_{train}, \omega)$. For fixed $0 \leq \varepsilon \leq 1$ and $0 \leq k \leq 1$, we say that M is *ε-unforgeable* despite k if and only if, for all the trigger sets $\mathcal{D}'_{trigger}$ such that $a(M, \mathcal{D}'_{trigger}) \leq k$, the probability that $\text{VERIFY}(M, \mathcal{D}_{test}, \mathcal{D}'_{trigger})$ returns **True** is at most ε.

Since our formalism is general and abstracts from specific details of watermarking schemes, the definition is not concerned about how the attacker concretely forged $\mathcal{D}'_{trigger}$, but just assumes that the attacker is equipped with some $\mathcal{D}'_{trigger}$ such that $a(M, \mathcal{D}'_{trigger}) \leq k$. The parameter k defines how much M exhibits the intended behavior on the forged trigger set, with higher values of k providing additional power to the attacker, i.e., proving unforgeability for higher values of k provides better security guarantees.

Note that unforgeability assumes that the model owner may consider $\perp$ as an acceptable output of the verification algorithm, because the definition requires minimizing the probability that the output is **True**. In other words, the model owner wants to be confident that the attacker cannot forge conclusive ownership claims.

IV. Formal Security Analysis

We showed that our formalism is expressive enough to encode a range of existing watermarking schemes (Appendix A) and define relevant security properties. We now show that our model enables useful forms of probabilistic reasoning about the security of trigger-based watermarking schemes. We first analyze each security property in isolation and we then reconcile them within a unified framework in Section IV-E.

A. Preliminary Assumptions

Trigger-based watermarking schemes should require a high number of correct predictions on the trigger set (N' in our formalism). This intuitive fact is needed to provide statistically significant assurance about the presence of the watermark in light of the abnormal model behavior exposed on the trigger set. The importance of requiring a sufficiently high value of N' is a well understood concept in the literature; for example Jia et al. [5] propose the adoption of the classic t-test to determine the minimum value of N' that makes watermark verification statistically significant. In turn, the required number of correct predictions on the test set (N in our formalism) should be small. This follows from the goal of making watermark verification conclusive: if N was too large, most verification attempts may return $\perp$ instead of **True** or **False**, which are the only outcomes that allow the model owner to conclusively prove or disprove ownership. For this reason, among all the possible choices of N and N' that ensure the satisfaction of our security properties, we naturally privilege those with small values of N and high values of N'.

B. Verifiability

We first use our model to guide the choice of an appropriate threshold for watermark verification which ensures verifiability. Recall that prior work provides just simple empirical recommendations on this important aspect (see Appendix A).

[1] The use of the original test set here simplifies the definition and uniformly models the fact that the attacker may have access to other test instances where the watermarked model shows comparable accuracy.

1) Case $\mathcal{D}_{verify} = \mathcal{D}^*_{trigger}$: We start our probabilistic analysis from the restricted case $\mathcal{D}_{verify} = \mathcal{D}^*_{trigger}$, which is the most common setting in the literature. Since $\mathcal{D}^*_{test} = \emptyset$ in this case, we enforce the threshold on the number of correct predictions on the test set to $N = -1$, hence the verification algorithm cannot return $\perp$.

Theorem 1. Assume that $\textsc{Challenge}(M, \mathcal{D}_{test}) = \langle \emptyset, -1 \rangle$ and $\textsc{Challenge}(M, \mathcal{D}_{trigger}) = \langle \mathcal{D}^*_{trigger}, N' \rangle$. If p is the probability that M makes a correct prediction on $\mathcal{D}^*_{trigger}$, then $\textsc{Verify}(M, \mathcal{D}_{test}, \mathcal{D}_{trigger})$ returns **False** with probability:

$$\sum_{i=0}^{N'} \binom{|\mathcal{D}^*_{trigger}|}{i} \cdot p^i \cdot (1-p)^{|\mathcal{D}^*_{trigger}|-i}. \tag{1}$$

Proof. Since $\mathcal{D}_{verify} = \mathcal{D}^*_{trigger}$, we observe that watermark verification consists of $|\mathcal{D}^*_{trigger}|$ Bernoulli trials with probability of success p, hence the probability of success of exactly k trials is given by the binomial distribution as follows:

$$\binom{|\mathcal{D}^*_{trigger}|}{k} \cdot p^k \cdot (1-p)^{|\mathcal{D}^*_{trigger}|-k}.$$

The conclusion follows by observing that $\textsc{Verify}$ returns **False** if and only if the number of successes is at most N'. □

Leveraging the observation that the number of successes in our proof follows a binomial distribution, the probability that a model M fails watermark verification on a trigger set $\mathcal{D}_{trigger}$ can be effectively approximated by the Central Limit Theorem for large values of $|\mathcal{D}^*_{trigger}|$ as follows:

$$\Phi\left(\frac{N' - |\mathcal{D}^*_{trigger}| \cdot a(M, \mathcal{D}_{trigger})}{\sqrt{|\mathcal{D}^*_{trigger}| \cdot a(M, \mathcal{D}_{trigger}) \cdot (1 - a(M, \mathcal{D}_{trigger}))}}\right), \tag{2}$$

where Φ is the cumulative distribution function of the normal distribution. We can then replace the formula in Equation 1 with its normal approximation in Equation 2, which we denote by $\hat{\Phi}(a(M, \mathcal{D}_{trigger}), |\mathcal{D}^*_{trigger}|, N')$, to simplify notation and computations. In the following, we only work with normal approximations, because we always assume that the number of verification queries is large enough (≥ 30). We remark that our results do not critically rely on this approximation: the same probability can be computed exactly using the binomial distribution directly.

The following corollary of Theorem 1 provides a systematic way to reason about verifiability when $\mathcal{D}_{verify} = \mathcal{D}^*_{trigger}$.

Corollary 1. Assume that $\textsc{Challenge}(M, \mathcal{D}_{test}) = \langle \emptyset, -1 \rangle$ and $\textsc{Challenge}(M, \mathcal{D}_{trigger}) = \langle \mathcal{D}^*_{trigger}, N' \rangle$, then M is ε-verifiable iff $\hat{\Phi}(a(M, \mathcal{D}_{trigger}), |\mathcal{D}^*_{trigger}|, N') \leq \varepsilon$.

This result is useful because it allows one to properly set the value of N' to ensure verifiability, based on the value of $a(M, \mathcal{D}_{trigger})$ and $|\mathcal{D}^*_{trigger}|$, as we exemplify next.

Example 3 (Application of Corollary 1)**.** Assume that M is a watermarked model such that $a(M, \mathcal{D}_{trigger}) = 0.7$ and $|\mathcal{D}^*_{trigger}| = 800$. If $N' = 600$, we have $\hat{\Phi}(0.7, 800, 600) \approx 1$,

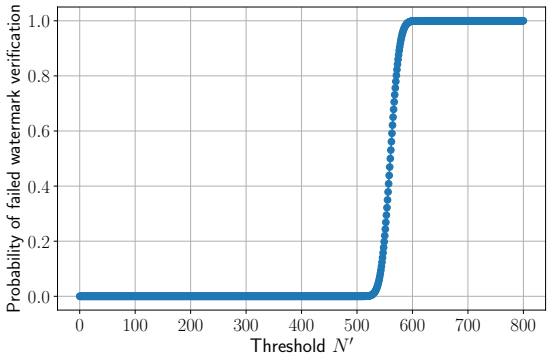


Fig. 1: Probability of failed watermark verification for the case $\mathcal{D}_{verify} = \mathcal{D}^*_{trigger}$ under different values of N', assuming that $a(M, \mathcal{D}_{trigger}) = 0.7$ and $|\mathcal{D}^*_{trigger}| = 800$.

meaning that the verification algorithm fails almost for sure. The picture would be quite different if we rather set $N' = 500$, because we would have $\hat{\Phi}(0.7, 800, 500) \approx 0$, i.e., the verification algorithm fails with negligible probability. This huge variation suggests that tuning N' to ensure verifiability without any formal guidance would be hard. Figure 1 shows how the probability of failed watermark verification changes for different values of N', while keeping fixed $a(M, \mathcal{D}_{trigger}) = 0.7$ and $|\mathcal{D}^*_{trigger}| = 800$. The function shows a sigmoid-like behavior which is monotonic in N', meaning that lower values of N' lead to a lower probability of watermark verification failure. The probability of verification failure starts near 0 for inputs from 0 to around 500, increases rapidly in the transition region between 500 and 600, and approaches 1 from 600 onwards. This tells us that setting $N' \approx 500$ is a reasonable option for verifiability, because requiring $N' > 500$ may expose the watermarked model to the risk of failed watermark verification even by the legitimate model owner.

2) Case $\mathcal{D}_{verify} = \mathcal{D}^*_{test} \cup \mathcal{D}^*_{trigger}$: We now extend our analysis to the general case $\mathcal{D}_{verify} = \mathcal{D}^*_{test} \cup \mathcal{D}^*_{trigger}$, thus supporting reasoning about the verifiability of watermarking schemes using also test instances for verification purposes.

Theorem 2. Assume that $\textsc{Challenge}(M, \mathcal{D}_{test}) = \langle \mathcal{D}^*_{test}, N \rangle$ and $\textsc{Challenge}(M, \mathcal{D}_{trigger}) = \langle \mathcal{D}^*_{trigger}, N' \rangle$. If p_1 and p_2 are the probabilities that M makes a correct prediction on $\mathcal{D}^*_{test}$ and $\mathcal{D}^*_{trigger}$ respectively, then $\textsc{Verify}(M, \mathcal{D}_{test}, \mathcal{D}_{trigger})$ returns:

- **True** with probability $(1 - \hat{\Phi}(p_1, |\mathcal{D}^*_{test}|, N)) \cdot (1 - \hat{\Phi}(p_2, |\mathcal{D}^*_{trigger}|, N'))$.
- **False** with probability $(1 - \hat{\Phi}(p_1, |\mathcal{D}^*_{test}|, N)) \cdot \hat{\Phi}(p_2, |\mathcal{D}^*_{trigger}|, N')$.
- $\perp$ with probability $\hat{\Phi}(p_1, |\mathcal{D}^*_{test}|, N)$.

Proof. We prove just the first point. The other two points follow by a similar argument. Observe that $\mathcal{D}_{verify} = \mathcal{D}^*_{test} \cup \mathcal{D}^*_{trigger}$ and that $\textsc{Verify}(M, \mathcal{D}_{test}, \mathcal{D}_{trigger})$ returns **True** if and only if M makes more than N correct predictions on $\mathcal{D}^*_{test}$

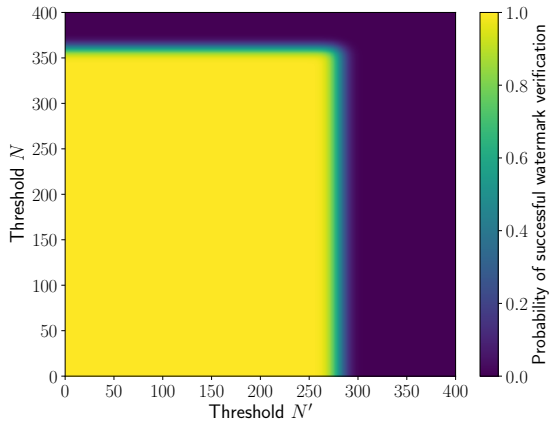


Fig. 2: Probability of successful watermark verification for the case $\mathcal{D}_{verify} = \mathcal{D}^*_{test} \cup \mathcal{D}^*_{trigger}$ under different values of N and N', assuming that $a(M, \mathcal{D}_{test}) = 0.9$, $a(M, \mathcal{D}_{trigger}) = 0.7$ and $|\mathcal{D}^*_{test}| = |\mathcal{D}^*_{trigger}| = 400$.

and more than N' correct predictions on $\mathcal{D}^*_{trigger}$. Since the number of successes in the two cases follows a binomial distribution with success probabilities p_1 and p_2 respectively, the relevant events occur with probabilities $1 - \hat{\Phi}(p_1, |\mathcal{D}^*_{test}|, N)$ and $1 - \hat{\Phi}(p_2, |\mathcal{D}^*_{trigger}|, N')$ respectively. Since the predictions on $\mathcal{D}^*_{test}$ and $\mathcal{D}^*_{trigger}$ are independent events, the probability that $\textsc{Verify}(M, \mathcal{D}_{test}, \mathcal{D}_{trigger})$ returns **True** is the product $(1 - \hat{\Phi}(p_1, |\mathcal{D}^*_{test}|, N)) \cdot (1 - \hat{\Phi}(p_2, |\mathcal{D}^*_{trigger}|, N'))$. □

The following corollary of Theorem 2 provides a systematic way to assess verifiability when $\mathcal{D}_{verify} = \mathcal{D}^*_{test} \cup \mathcal{D}^*_{trigger}$.

Corollary 2. Assume that $\textsc{Challenge}(M, \mathcal{D}_{test}) = \langle \mathcal{D}^*_{test}, N \rangle$ and $\textsc{Challenge}(M, \mathcal{D}_{trigger}) = \langle \mathcal{D}^*_{trigger}, N' \rangle$, then M is ε-verifiable iff $(1 - \hat{\Phi}(a(M, \mathcal{D}_{test}), |\mathcal{D}^*_{test}|, N)) \cdot (1 - \hat{\Phi}(a(M, \mathcal{D}_{trigger}), |\mathcal{D}^*_{trigger}|, N')) \geq 1 - \varepsilon$.

This result is useful because it quantifies the impact of the two main ways to ensure verifiability, i.e., lowering N or N', whenever the parameters $a(M, \mathcal{D}_{test})$, $a(M, \mathcal{D}_{trigger})$, $|\mathcal{D}^*_{test}|$ and $|\mathcal{D}^*_{trigger}|$ are fixed.

Example 4 (Application of Corollary 2)**.** Figure 2 shows how the probability of successful watermark verification changes for different values of N and N', assuming that $a(M, \mathcal{D}_{test}) = 0.9$, $a(M, \mathcal{D}_{trigger}) = 0.7$ and $|\mathcal{D}^*_{test}| = |\mathcal{D}^*_{trigger}| = 400$. The figure shows that M satisfies verifiability in the yellow area approximately described by $N \leq 350 \wedge N' \leq 275$, thus providing an effective way to set a bound on these two parameters. Note again the abrupt success probability drop outside the yellow area, coming from the fast transitions of the underlying sigmoid functions. In the end, the figure suggests that setting $N' \approx 275$ is appropriate for verifiability, provided that $N \leq 350$.

C. Robust Verifiability

We now reason about robust verifiability, i.e., verifiability in the adversarial setting. We first use our formal framework to identify a significant shortcoming of existing trigger-based watermarking schemes that enables a simple yet effective attack strategy and we propose an intuitive mitigation. We then introduce a more general attack strategy and we reason about the effectiveness of our proposed mitigation against it.

1) Insecurity of Existing Watermarking Schemes: As we said, most trigger-based watermarking schemes do not use test instances in the watermark verification process. Unfortunately, we can use our formalism to prove that this simple choice is unsound, despite its intuitive flavor. This result is interesting because it is the abnormal model behavior exposed on the trigger set that enables watermarking, hence the role of the test set for watermark verification may be unclear. Still, we can show that a simple attack strategy where the attacker detects watermark verification attempts and always returns a random label in response can effectively enforce watermark suppression. The intuition of the attack is that the random behavior of the attacker is unlikely to match the peculiar behavior that the watermarked model is intended to expose, leading to failed verifications.

Prior work has also acknowledged that the attacker might return random labels during a watermark suppression attack, after detecting trigger instances, to harm ownership verification [8], [9], [5], [6]. However, these works do not formally analyze the robustness against the attack, but they only empirically evaluate the detectability of trigger instances in specific watermarking schemes as a first step to perform watermark suppression. Our work, in turn, effectively quantifies the actual effectiveness of this attack strategy as follows.

Proposition 1. Assume that $\textsc{Challenge}(M, \mathcal{D}_{test}) = \langle \emptyset, -1 \rangle$ and $\textsc{Challenge}(M, \mathcal{D}_{trigger}) = \langle \mathcal{D}^*_{trigger}, N' \rangle$. Let $A_M(\vec{x})$ be the attacker that returns a random label for each possible input $\vec{x}$, then M is robustly ε-verifiable against A_M iff $\hat{\Phi}(1/|\mathcal{Y}|, |\mathcal{D}^*_{trigger}|, N') \leq \varepsilon$.

Proof. By Theorem 1, because the probability of making a correct prediction on $\mathcal{D}^*_{trigger}$ is $1/|\mathcal{Y}|$. □

Example 5 (Application of Proposition 1)**.** To appreciate the effectiveness of the attack strategy, consider a scenario where $a(M, \mathcal{D}_{trigger}) = 0.7$ and $|\mathcal{D}^*_{trigger}| = 800$. Figure 3 shows how the probability of watermark verification failure changes for different values of N' from the non-adversarial case (blue line, computed using Theorem 1) to the adversarial case (orange line, computed using Proposition 1), making clear the advantage that the attacker gains from the proposed watermark suppression strategy. The figure shows that setting $N' \approx 500$, which was a sound recommendation for verifiability in the considered setting (see Example 3), becomes unsound in the presence of an active adversary, because the probability of failures during watermark verification approaches 1 for $N' \geq 400$.

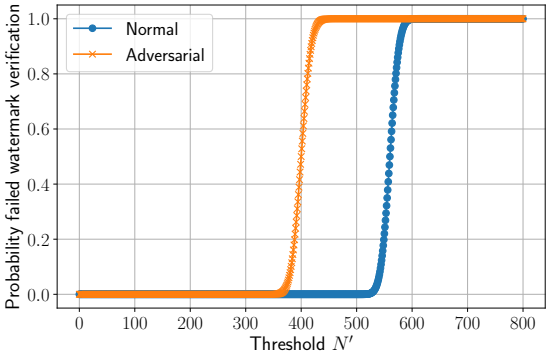


Fig. 3: Probability of failed watermark verification for the case $\mathcal{D}_{verify} = \mathcal{D}^*_{trigger}$ under different values of N' in the adversarial case where the attacker returns a random label (orange line) and the non-adversarial case (blue line), with $a(M, \mathcal{D}_{trigger}) = 0.7$, $|\mathcal{D}^*_{trigger}| = 800$ and $|\mathcal{Y}| = 2$.

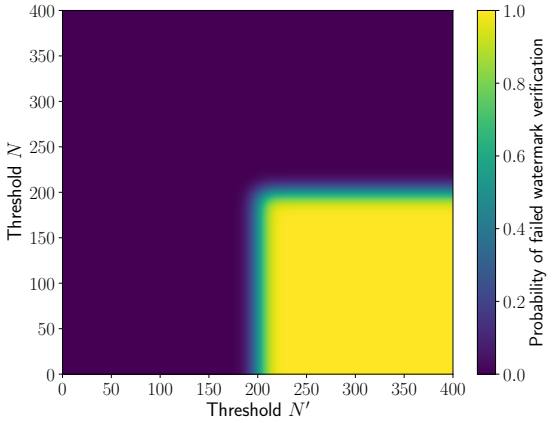


Fig. 4: Probability of failed watermark verification for the case $\mathcal{D}_{verify} = \mathcal{D}^*_{test} \cup \mathcal{D}^*_{trigger}$ under different values of N and N', assuming that the attacker returns a random label ($|\mathcal{Y}| = 2$).

Based on the previous analysis, it is apparent that using the trigger set alone does not yield a robust verification procedure and we propose to leverage the test set to improve security against watermark suppression. We can immediately show that the simple attack strategy presented for the case $\mathcal{D}_{verify} = \mathcal{D}^*_{trigger}$ is much less effective in the general setting where $\mathcal{D}_{verify} = \mathcal{D}^*_{test} \cup \mathcal{D}^*_{trigger}$. The intuition is that returning a random label is not an effective attack strategy in the general setting, because this significantly drops the accuracy on the test set, leading to an increased probability that the watermark verification algorithm returns $\perp$.

Proposition 2. Assume that $\textsc{Challenge}(M, \mathcal{D}_{test}) = \langle \mathcal{D}^*_{test}, N \rangle$ and $\textsc{Challenge}(M, \mathcal{D}_{trigger}) = \langle \mathcal{D}^*_{trigger}, N' \rangle$. Let $A_M(\vec{x})$ be the attacker that returns a random label for each input $\vec{x}$, then M is robustly ε-verifiable against A_M iff $(1 - \hat{\Phi}(1/|\mathcal{Y}|, |\mathcal{D}^*_{test}|, N)) \cdot \hat{\Phi}(1/|\mathcal{Y}|, |\mathcal{D}^*_{trigger}|, N') \leq \varepsilon$.

Proof. By Theorem 2, because the probability of making a correct prediction on both $\mathcal{D}^*_{test}$ and $\mathcal{D}^*_{trigger}$ is $1/|\mathcal{Y}|$. □

Example 6 (Application of Proposition 2)**.** To visualize the importance of including test instances in the watermark verification process, Figure 4 shows how the probability of failed watermark verification changes for different values of N and N' in the adversarial setting where the attacker always returns a random label, assuming a binary classification setting with $|\mathcal{Y}| = 2$. The figure shows that M fails robust verifiability in the yellow area described approximately by $N < 200 \wedge N' > 200$. Intuitively, this captures the observation that, since the attacker always returns a random label, the probability of making a sufficiently high number of correct predictions on the test set becomes negligible when $N \geq 200$, leading to the watermark verification algorithm returning $\perp$. This analysis tells us that it is possible to set $N' > 200$ while enforcing robust verifiability, provided that a sufficiently high number of correct predictions on test instances is required ($N \geq 200$). This is compatible with our previous verifiability analysis in Example 4, where we proposed to set $N' \approx 275$ with $N \leq 350$ in this setting.

2) General Attack Strategy: The previous analysis showed that, whenever the attacker is given an instance during the watermark verification phase in the general $\mathcal{D}_{verify} = \mathcal{D}^*_{test} \cup \mathcal{D}^*_{trigger}$ setting, they may not know whether the instance comes from $\mathcal{D}^*_{test}$ or from $\mathcal{D}^*_{trigger}$, hence suppressing the watermark is intuitively harder than for the case $\mathcal{D}_{verify} = \mathcal{D}^*_{trigger}$. In fact, the attacker may not know the best action to take to maximize the probability that the verification algorithm returns **False**. We here formalize an attacker who tries to guess whether an instance comes from $\mathcal{D}^*_{test}$ or from $\mathcal{D}^*_{trigger}$ with a given success probability, to then make another guess on the returned label with a corresponding probability of success.

Theorem 3. Assume that $\textsc{Challenge}(M, \mathcal{D}_{test}) = \langle \mathcal{D}^*_{test}, N \rangle$ and $\textsc{Challenge}(M, \mathcal{D}_{trigger}) = \langle \mathcal{D}^*_{trigger}, N' \rangle$. Let $A_M(\vec{x})$ be the attacker that operates as follows:

1) The attacker guesses whether $\vec{x}$ is a test instance or a trigger instance with probability of success $q \geq 0.5$.
2) If the attacker believes that $\vec{x}$ is a test instance, they return a label which is correct with probability q_{11} if $\vec{x}$ was indeed a test instance and is correct with probability q_{12} if $\vec{x}$ was instead a trigger instance.
3) If the attacker believes that $\vec{x}$ is a trigger instance, they return a label which is correct with probability q_{21} if $\vec{x}$ was indeed a trigger instance and is correct with probability q_{22} if $\vec{x}$ was instead a test instance.

M is robustly ε-verifiable against A_M if and only if $(1 - \hat{\Phi}(p_1, |\mathcal{D}^*_{test}|, N)) \cdot \hat{\Phi}(p_2, |\mathcal{D}^*_{trigger}|, N') \leq \varepsilon$, where $p_1 = q \cdot q_{11} + (1 - q) \cdot q_{22}$ and $p_2 = q \cdot q_{21} + (1 - q) \cdot q_{12}$.

Proof. Observe that $\mathcal{D}_{verify} = \mathcal{D}^*_{test} \cup \mathcal{D}^*_{trigger}$ and that $\textsc{Verify}(M, \mathcal{D}_{test}, \mathcal{D}_{trigger})$ returns **False** if and only if M makes more than N correct predictions on $\mathcal{D}^*_{test}$ and

at most N' correct predictions on $\mathcal{D}^*_{trigger}$. Let p_1 and p_2 be the probabilities of making a correct prediction on $\mathcal{D}^*_{test}$ and $\mathcal{D}^*_{trigger}$ respectively. Since the number of successes in the two cases follows a binomial distribution with success probabilities p_1 and p_2 respectively, the relevant events occur with probabilities $(1 - \hat{\Phi}(p_1, |\mathcal{D}^*_{test}|, N))$ and $\hat{\Phi}(p_2, |\mathcal{D}^*_{trigger}|, N')$ respectively. Since the predictions on $\mathcal{D}^*_{test}$ and $\mathcal{D}^*_{trigger}$ are independent events, the probability that $\textsc{Verify}(M, \mathcal{D}_{test}, \mathcal{D}_{trigger})$ returns **False** is the product $(1 - \hat{\Phi}(p_1, |\mathcal{D}^*_{test}|, N)) \cdot \hat{\Phi}(p_2, |\mathcal{D}^*_{trigger}|, N')$.

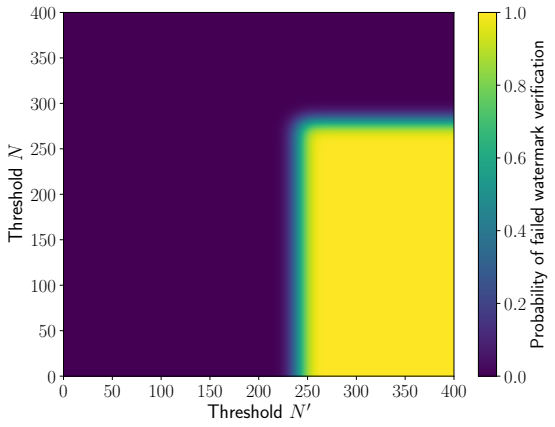
To conclude the proof, we just need to show that $p_1 = q \cdot q_{11} + (1-q) \cdot q_{22}$ and $p_2 = q \cdot q_{21} + (1-q) \cdot q_{12}$. This can be done as follows:

- p_1 is the probability of making a correct prediction on $\mathcal{D}^*_{test}$. For any test instance, we have two possibilities: either the attacker makes the correct guess with probability q and provides the correct label with probability q_{11}, or the attacker makes the wrong guess with probability $1 - q$ and provides the correct label with probability q_{22}. Combining these observations, we have $p_1 = q \cdot q_{11} + (1-q) \cdot q_{22}$.
- p_2 is the probability of making a correct prediction on $\mathcal{D}^*_{trigger}$. For any trigger instance, we have two possibilities: either the attacker makes the correct guess with probability q and provides the correct label with probability q_{21}, or the attacker makes the wrong guess with probability $1-q$ and provides the correct label with probability q_{12}. Combining these observations, we have $p_2 = q \cdot q_{21} + (1-q) \cdot q_{12}$.

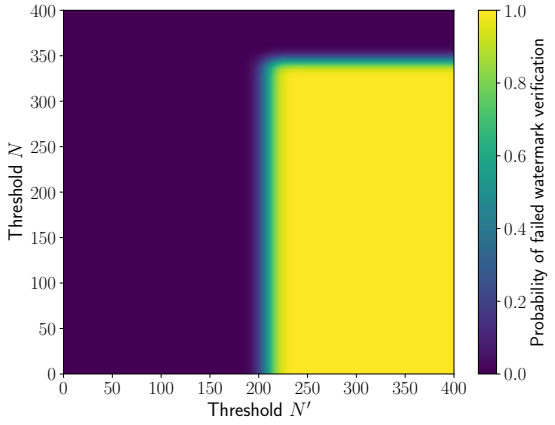
□

Example 7 (Application of Theorem 3 - Watermark Suppression)**.** We first exemplify our theorem at work for the concrete attack strategy described in Example 1. We compare the probability of failed watermark verification for the case $q = 0.5$, where the attacker randomly guesses whether an instance comes from $\mathcal{D}^*_{test}$ or $\mathcal{D}^*_{trigger}$, and the case $q = 0.9$, where the attacker can effectively discriminate test instances from trigger instances. In this scenario, we have $q_{11} = a(M, \mathcal{D}_{test})$, $q_{12} = a(M, \mathcal{D}_{trigger})$, $q_{21} = q_{22} = 1/|\mathcal{Y}|$. Figure 5 compares the distribution of the two probabilities of failure for the case $a(M, \mathcal{D}_{test}) = 0.9$, $a(M, \mathcal{D}_{trigger}) = 0.7$ and $|\mathcal{Y}| = 2$ when $|\mathcal{D}^*_{test}| = |\mathcal{D}^*_{trigger}| = 400$. The figure shows a noticeable increase in the area where watermark verification fails almost surely when q increases from 0.5 to 0.9. In particular, the area in the first sub-figure is approximately described by $N < 275 \wedge N' > 250$, while the area in the second sub-figure is approximately described by $N < 350 \wedge N' > 200$.

Example 7 formally captures that trigger-based watermarking schemes must ensure that trigger instances are indistinguishable from test instances to support robust verifiability. Indeed, schemes generating trigger instances by superimposing extra content, such as a logo or a mark [14], or those using out-of-distribution or random instances as triggers [21], [5], [23] often overlook the dissimilarity between test and trigger images, rendering them particularly vulnerable to watermark suppression. As confirmation, this distinguishability has been exploited by prior empirical work to mount a practical watermark suppression attack against these schemes [20], though without formally analyzing their security posture.



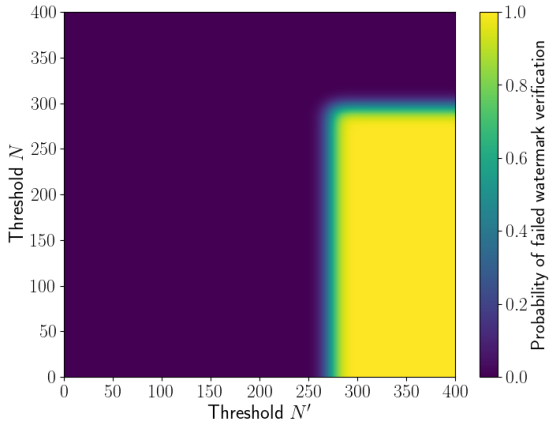
(a) Case $q = 0.5$



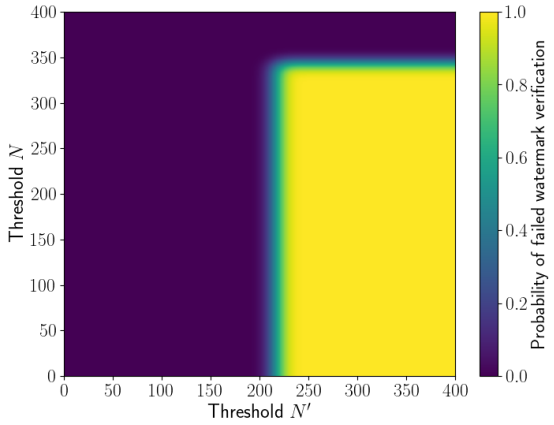
(b) Case $q = 0.9$

Fig. 5: Probability of failed watermark verification for the case $\mathcal{D}_{verify} = \mathcal{D}^*_{test} \cup \mathcal{D}^*_{trigger}$ under different values of N and N', assuming that the attacker can correctly discriminate test instances from trigger instances with probability q, $a(M, \mathcal{D}_{test}) = 0.9$, $a(M, \mathcal{D}_{trigger}) = 0.7$, $|\mathcal{Y}| = 2$.

Example 8 (Application of Theorem 3 - Model Extraction)**.** We further show that our theorem generalizes to the attack strategy described in Example 2. Differently from Example 1, the attacker does not actively attempt to make watermark verification fail. Instead, they return the prediction of the extracted model, that may differ from that of the original model, especially on the trigger set, potentially causing verification to fail. Recall that in Example 2 we introduced an attacker for model extraction in terms of two probabilities, q_1 and q_2, that model how much the extraction process



(a) Case $q_1 = 0.6$ and $q_2 = 0.9$



(b) Case $q_1 = 0.9$ and $q_2 = 0.2$

Fig. 6: Probability of failed watermark verification for the case $\mathcal{D}_{verify} = \mathcal{D}^*_{test} \cup \mathcal{D}^*_{trigger}$ under different values of N and N', assuming that the attacker has performed model extraction preserving the performance of the original model on the test and trigger sets, respectively, with probability q_1 and q_2, $a(M, \mathcal{D}_{test}) = 0.9$, $a(M, \mathcal{D}_{trigger}) = 0.7$, $|\mathcal{Y}| = 2$.

preserved the accuracy of the original model on the test set and trigger set, respectively. We compare the probability of failed watermark verification in two cases: $q_1 = 0.6$ and $q_2 = 0.9$, representing a negative scenario for the attacker, where the extracted model exhibits degraded performance on the test set while preserving performance on the trigger set; and $q_1 = 0.9$ and $q_2 = 0.2$, representing a positive scenario for the attacker, where the extracted model preserves performance on the test set but shows degraded performance on the trigger set. We reason about this scenario using Theorem 3 by setting $q = 1$, $q_{11} = q_1 \cdot a(M, \mathcal{D}_{test}) + (1 - q_1) \cdot 1/|\mathcal{Y}|$ and $q_{21} = q_2 \cdot a(M, \mathcal{D}_{trigger}) + (1 - q_2) \cdot 1/|\mathcal{Y}|$; q_{12} and q_{22} are irrelevant because $q = 1$. Figure 6 compares the distribution of the failure probabilities for $a(M, \mathcal{D}_{test}) = 0.9$, $a(M, \mathcal{D}_{trigger}) = 0.7$ and $|\mathcal{Y}| = 2$ when $|\mathcal{D}^*_{test}| = |\mathcal{D}^*_{trigger}| = 400$. The figure shows a larger area where watermark verification fails almost surely when q_1 increases from 0.6 to 0.9 and q_2 decreases from 0.9 to 0.2. In particular, the area in the first sub-figure is approximately described by $N < 300 \wedge N' > 275$, while the area in the second sub-figure is approximately described by $N < 350 \wedge N' > 225$.

D. Unforgeability

We conclude our formal analysis with a discussion on unforgeability. Recall that unforgeability provides an upper bound on the probability that watermark verification succeeds when the attacker unduly attempts an ownership claim of a model M using a forged trigger set $\mathcal{D}'_{trigger}$ such that $a(M, \mathcal{D}'_{trigger}) \leq k$, where k models how effective the forgery attack was. Similar to the analyses performed in the previous sections, we are able to characterize the unforgeability of a model by means of the following theorem.

Theorem 4. Assume that $\textsc{Challenge}(M, \mathcal{D}_{test}) = \langle \mathcal{D}^*_{test}, N \rangle$ and $\textsc{Challenge}(M, \mathcal{D}_{trigger}) = \langle \mathcal{D}^*_{trigger}, N' \rangle$. We have that M is ε-unforgeable despite k iff $(1 - \hat{\Phi}(a(M, \mathcal{D}_{test}), |\mathcal{D}^*_{test}|, N)) \cdot (1 - \hat{\Phi}(k, |\mathcal{D}^*_{trigger}|, N')) \leq \varepsilon$.

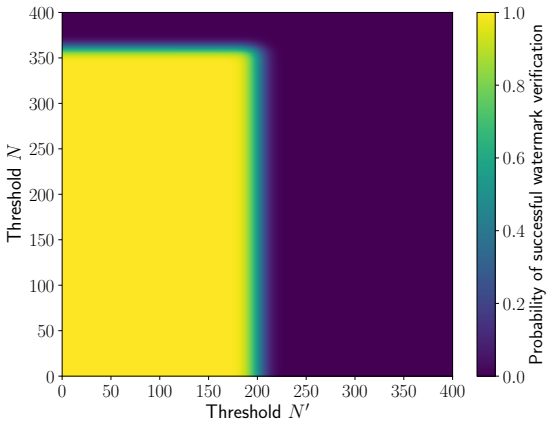
Proof. By Theorem 2, observing that the probability of making a correct prediction on $\mathcal{D}^*_{test}$ is $a(M, \mathcal{D}_{test})$ and the probability of making a correct prediction on $\mathcal{D}^*_{trigger}$ is at most k. □

Example 9 (Application of Theorem 4)**.** We here show our theorem at work by comparing the probability of successful watermark verification for the cases $k = 0.5$ and $k = 0.65$, corresponding to the attacker being able to forge trigger sets of different quality. Figure 7 compares the distribution of the two probabilities of successful verification for the forged trigger set in the case $a(M, \mathcal{D}_{test}) = 0.9$ when $|\mathcal{D}^*_{test}| = |\mathcal{D}^*_{trigger}| = 400$. The figure shows an increase in the area where watermark verification succeeds almost surely when k increases from 0.5 to 0.65. In particular, the area in the first sub-figure is roughly described by $N < 350 \wedge N' < 200$, while the area in the second sub-figure is approximately described by $N < 350 \wedge N' < 250$. This tells us that setting $N' \geq 250$ provides appropriate unforgeability guarantees and grants effective protection even for the stronger attack where $k = 0.65$. Note that the role of N is almost immaterial for unforgeability, because we conservatively assume that the attacker has access to the original test set.

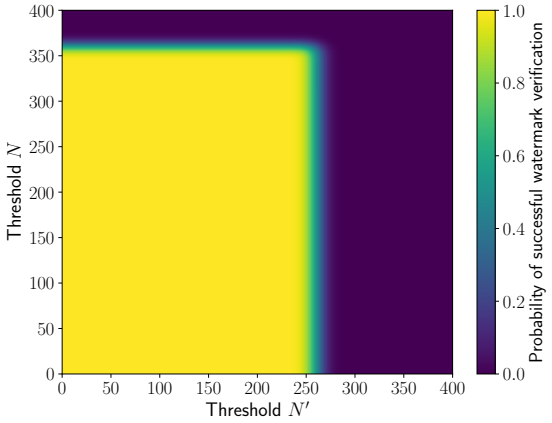
E. Foundations of Secure Trigger-Based Watermarking

We here summarize the key take-away messages of our formal analysis and we discuss how to reconcile the different security properties of interest in practice.

1) Take-Away Messages: A sound design of trigger-based watermarking schemes should not leave unspecified the number of successful verification queries, otherwise watermark verification may unexpectedly fail even for the legitimate model owner. This would break verifiability (Definition 1) and void the intended intellectual property protection. Importantly,



(a) Case $k = 0.5$



(b) Case $k = 0.65$

Fig. 7: Probability of successful watermark verification for the case $\mathcal{D}_{verify} = \mathcal{D}^*_{test} \cup \mathcal{D}^*_{trigger}$ under different values of N and N', assuming that the attacker can craft a trigger set $\mathcal{D}'_{trigger}$ such that $a(M, \mathcal{D}'_{trigger}) \leq k$, with $a(M, \mathcal{D}_{test}) = 0.9$.

the number of successful verification queries does not need to be empirically set on a case-by-case basis, but it is possible to choose it based on the accuracy of the watermarked model (see Corollary 1 and Corollary 2). Unfortunately, prior work in the field just proposed empirical techniques to set the number of successes [8], [6], [13], did not discuss this point at all [4], [20], [24], or focused on the definition of a *minimum* number of successes to ensure statistical significance of watermark verification, without discussing the verifiability risks of requiring an excessively high number of successes [5], [7], [23].

Moreover, trigger-based watermarking should not rely on the trigger set alone for verification purposes, otherwise effective watermark suppression attacks are trivial to perform. Intuitively, if the attacker has control of the watermarked model, they can answer the verification queries with random labels, as also empirically shown in prior work [8], [9], [5], [6], thus hiding the model behavior that watermark embedding was intended to expose. We then recommend the inclusion of the test set in the watermark verification algorithm to increase security against watermark suppression attacks, which is sufficient to mitigate the simple attack strategy based on random answers (see Proposition 2). Unfortunately, virtually all the watermarking schemes that we surveyed do not comply with this recommendation, with the single exception of the proposal by Guo and Potkonjak [4]. For quantitative recommendations on how to set the number of test and trigger instances to ensure robust verifiability, we refer to Theorem 3 and the related discussion. An interesting consequence of our theorem is that trigger-based watermarking schemes based on *abnormal* trigger instances, such as [14], [4], [21], [5], [23], are unlikely to produce models satisfying robust verifiability. An attacker who can discriminate test instances from trigger instances can adapt their behavior to make watermark verification fail with high probability, thus preventing the model owner from claiming ownership [8], [9], [5], [6]. In addition, a robust watermarking scheme should ensure that any model extraction attack preserving accuracy on the main task also preserves it on the trigger set. This requirement is not satisfied by several existing schemes [9]; for this reason, recent proposals address the issue by drawing trigger inputs from the main task distribution, e.g., by *entangling* them with it [5], [8], [24], [7].

2) Enforcing Security: Verifiability, robust verifiability and unforgeability are all important concerns for trigger-based watermarking schemes and any meaningful security analysis cannot consider these properties in isolation, but should carefully consider all of them. An important observation we make is that these properties are often at stake, because they introduce conflicting requirements. For example, setting a low number of successful verification queries N' for the trigger set helps verifiability, but harms unforgeability.

To set the parameters N and N' to simultaneously enforce all our security properties, one could just enumerate all the possible pairs $\langle N, N' \rangle$, checking if any of them satisfies all the constraints required for the individual security properties. This exhaustive search has complexity $O(|\mathcal{D}^*_{test}| \cdot |\mathcal{D}^*_{trigger}|)$. In practice, recall that it is generally preferable to consider solutions with small values of N and high values of N' (see the discussion in Section IV-A). For this reason, our prototype implementation explores the search space by starting from the highest possible value of N' and iterating over all the possible values of N to check whether any pair $\langle N, N' \rangle$ allows constraint satisfaction. If no choice of N works for the given N', the value of N' is decremented and a new iteration over N starts. The process fails when no solution is found. The specific constraints to be enforced depend on a preliminary threat modeling phase, where the model owner sets the security parameter $\varepsilon \in [0, 1]$ and performs additional evaluations, e.g., determines whether model extraction attacks are a relevant threat for them. Appendix B shows our approach at work for several values of the parameters modeling the attacker's capabilities, highlighting cases where a solution can be found and cases where no solution is possible. We further report on the benefits of our principled approach in the next section.

V. Experimental Evaluation

Since all the results presented so far have been computed using synthetic data, we now show the practical utility of our formal framework on real watermarked models.

A. Methodology

We experiment with watermarked deep neural networks for image classification, because most research on model watermarking focuses on this setting. Specifically, we work with two widely used image classification datasets: Fashion-MNIST [25] and CIFAR10 [26]. Fashion-MNIST (FMNIST) comprises 70,000 28×28 grayscale images of fashion items, categorized into 10 classes. CIFAR10, by contrast, consists of 60,000 32×32 color images of animals and vehicles across 10 class labels. Both datasets were preprocessed so that the test set includes 10,000 images, while the size of the trigger set was chosen based on the recommendations provided for the given watermarking schemes; the rest of the images were used for training. Following the experimental setup in [7], we trained a Convolutional Neural Network (CNN) on Fashion-MNIST and a ResNet-18 [27] on CIFAR10. For CIFAR10, we applied standard data augmentation techniques such as random cropping, horizontal flipping, and rotation. Models trained on Fashion-MNIST ran for 50 epochs, while those trained on CIFAR10 ran for 40 epochs. All models were optimized using the Adam optimizer with a learning rate of 0.001, and the batch size was set to 128 across all datasets. Accuracy on the test set ranges between 0.89 and 0.92, while accuracy on the trigger set ranges between 0.91 and 0.99 (see Table I).

We consider three existing trigger-based watermarking schemes to assess their security posture using our framework:

1) The scheme by Guo and Potkonjak [4], because it is the only one making use of test instances for watermark verification and our analysis confirms the soundness of this design choice for robust verifiability. Unfortunately, this scheme does not include a method or recommendation for setting the value of the parameters N and N'. We set both thresholds N and N' to 85% of the size of the test and trigger sets, consistent with the design of the scheme where these two values are identical.
2) The scheme by Li et al. [6], because it is representative of a class of watermarking schemes where the parameter N' is empirically set. Specifically, the authors recommend setting the threshold "close to 1". Thus, we set N' to 90% of the trigger set, while we set N to -1 because the test set is not used.
3) The scheme by Tan et al. [7], because it is representative of a class of watermarking schemes where the parameter N' is set by means of a statistical test (the t-test). This principled approach is useful to ensure that the behavior of the watermarked model differs statistically from that of a non-watermarked model, however it does not take security into full account as we do in our framework. We set N' using the t-test, setting a significance level of 0.05 to detect differences with respect to the accuracy of a non-watermarked model on the trigger set. Finally, we set N to -1 because the test set is not used.

We reason about the security properties of the watermarked models using our formal framework. We set a single $\varepsilon = 0.05$, i.e., we want to satisfy all the security properties with probability 0.95. In each experiment, we first evaluate security after setting the parameters N and N' as described above. We then use the approach of Section IV-E to automatically look for appropriate values of N and N' enforcing all the security properties at the same time. Finally, we discuss the results of our security analysis to assess whether our principled approach to set N and N' pays off in practice. We evaluate the security of watermarking schemes considering different values of the parameters modeling the attacker's strength, i.e., q, q_1 and q_2 for robust verifiability[2] and k for unforgeability. To support reproducibility, we make our code available online [28].

B. Experimental Results

In our first experiment, we are concerned about security against watermark suppression and forgery. Accordingly, we set parameters $q \in \{0.5, 0.8\}$ and $k \in \{0.5, 0.8\}$, corresponding to different assumptions about the attacker's strength, and we analyze the security posture of existing schemes using our formal framework. Table I reports the results of our experiment. We first observe that two out of the three watermarking schemes fail to satisfy one of the three security properties when setting the thresholds N and N' as recommended in the original papers. In particular, Li et al.'s empirical method of setting a very high value of N' makes the watermarked model secure against forgery, but does not satisfy robust verifiability: even a weak attacker ($q = 0.5$) can effectively suppress watermark verification. This result is concerning because the attacker does not perform any clever attempt to discriminate between test instances and trigger instances: random guessing suffices to make watermark verification fail in practice using the simple attack strategy of Example 1. Instead, Tan et al.'s scheme leads to setting a very low value of N', which provides robust verifiability, but breaks unforgeability: the threshold is so small that even a weak attacker ($k = 0.5$) could forge a trigger set sufficient to pass verification. We consider this attacker weak, because their performance on the trigger set is significantly lower than the accuracy on the original trigger set (0.99). In contrast, the model watermarked using the scheme by Guo and Potkonjak satisfies all three properties. While this result shows that the properties can be met with empirical thresholds in practice, the watermarked models exhibit them without a specific design intent: these thresholds have not been set taking security into due account, but rather to differentiate watermarked from non-watermarked model behavior. It is plausible that other models and datasets will not achieve appropriate security guarantees in the absence of principled indications on how to set the thresholds, and this is precisely the gap our framework helps to fill with formal rigor.

[2]Recall that q is used to model watermark suppression (Example 1), while q_1 and q_2 are used to reason about model extraction (Example 2).

TABLE I: Results of our formal analysis of three existing watermarking schemes. Here we consider watermark suppression as the attacker for robust verifiability. We report the size of $\mathcal{D}^*_{test}$, $\mathcal{D}^*_{trigger}$, as well as the values of N and N' obtained following the original papers in the **Empirical** column. We then indicate if our three security properties hold for the values of N and N' set following the original papers, for different values of q and k. Finally, in the **Formal** column, we present the values found by our principled approach for various values of q and k.

Scheme	Dataset	$a(M, \mathcal{D}_{test})$	$a(M, \mathcal{D}_{trigger})$	Empirical	q	k	Ver.	Rob. Ver.	Unf.	Formal
Guo and Potkonjak	FMNIST	0.90	0.91	$\lvert\mathcal{D}^*_{test}\rvert$ = 10,000 $\lvert\mathcal{D}^*_{trigger}\rvert$ = 10,000 N = 8,500 N' = 8,500	0.5 0.8 0.5 0.8	0.5 0.5 0.8 0.8	✓ ✓ ✓ ✓	✓ ✓ ✓ ✓	✓ ✓ ✓ ✓	N = 5,083, N' = 9,052 N = 7,473, N' = 9,052 N = 5,083, N' = 9,052 N = 7,473, N' = 9,052
	CIFAR10	0.89	0.92	$\lvert\mathcal{D}^*_{test}\rvert$ = 10,000 $\lvert\mathcal{D}^*_{trigger}\rvert$ = 10,000 N = 8,500 N' = 8,500	0.5 0.8 0.5 0.8	0.5 0.5 0.8 0.8	✓ ✓ ✓ ✓	✓ ✓ ✓ ✓	✓ ✓ ✓ ✓	N = 5,033, N' = 9,155 N = 7,393, N' = 9,155 N = 5,033, N' = 9,155 N = 7,393, N' = 9,155
Li et al.	FMNIST	0.90	0.99	$\lvert\mathcal{D}^*_{test}\rvert$ = 10,000 $\lvert\mathcal{D}^*_{trigger}\rvert$ = 600 $N = -1$ N' = 540	0.5 0.8 0.5 0.8	0.5 0.5 0.8 0.8	✓ ✓ ✓ ✓	✗ ✗ ✗ ✗	✓ ✓ ✓ ✓	N = 5,083, N' = 589 N = 7,473, N' = 589 N = 5,083, N' = 589 N = 7,473, N' = 589
	CIFAR10	0.91	0.96	$\lvert\mathcal{D}^*_{test}\rvert$ = 10,000 $\lvert\mathcal{D}^*_{trigger}\rvert$ = 500 $N = -1$ N' = 450	0.5 0.8 0.5 0.8	0.5 0.5 0.8 0.8	✓ ✓ ✓ ✓	✗ ✗ ✗ ✗	✓ ✓ ✓ ✓	N = 5,133, N' = 472 N = 7,552, N' = 472 N = 5,133, N' = 472 N = 7,552, N' = 472
Tan et al.	FMNIST	0.92	0.99	$\lvert\mathcal{D}^*_{test}\rvert$ = 10,000 $\lvert\mathcal{D}^*_{trigger}\rvert$ = 100 $N = -1$ N' = 11	0.5 0.8 0.5 0.8	0.5 0.5 0.8 0.8	✓ ✓ ✓ ✓	✓ ✓ ✓ ✓	✗ ✗ ✗ ✗	N = 5,183, N' = 97 N = 7,631, N' = 97 N = 5,183, N' = 97 N = 7,631, N' = 97
	CIFAR10	0.88	0.99	$\lvert\mathcal{D}^*_{test}\rvert$ = 10,000 $\lvert\mathcal{D}^*_{trigger}\rvert$ = 100 $N = -1$ N' = 4	0.5 0.8 0.5 0.8	0.5 0.5 0.8 0.8	✓ ✓ ✓ ✓	✓ ✓ ✓ ✓	✗ ✗ ✗ ✗	N = 4,983, N' = 97 N = 7,314, N' = 97 N = 4,983, N' = 97 N = 7,314, N' = 97

Remarkably, our principled approach identifies appropriate, security-aware thresholds for each watermarked model across the three schemes. This is a positive finding, because it shows that these schemes are not inherently insecure: they can be set up securely when parameters are chosen with care. For example, for Li et al.'s scheme on CIFAR10, considering the strongest attacker ($q = 0.8$, $k = 0.8$), our algorithm found $N = 7{,}552$ and $N' = 472$ as a possible solution satisfying all the three properties of interest. Moreover, if we follow the intuitions provided in Section IV-A, the thresholds found by our algorithm for the scheme by Guo and Potkonjak are more useful than those set using the informal recipes of the original paper. For CIFAR10 under the strongest attacker, our algorithm selects $N' = 9{,}155$, which is larger than the empirically set value (8,500), thus providing a stronger statistical guarantee of watermark presence. Moreover, our algorithm suggests setting $N = 7{,}393$, which is smaller than the empirically set value (8,500). This makes verification more conclusive by reducing the number of cases in which $\perp$ may be returned.

We also performed a similar experimental evaluation focusing on model extraction rather than watermark suppression. We set $q_1 \in \{0.5, 0.8\}$, $q_2 \in \{0.2, 0.8\}$, and $k \in \{0.5, 0.8\}$. Table V in Appendix C reports the results. As before, two of the three watermarking schemes fail to satisfy at least one of the three security properties when using the thresholds N and N' prescribed in the original papers. Li et al.'s trained model does not satisfy robust verifiability, even under a very weak attacker (i.e., $q_1 = 0.5$ and $q_2 = 0.8$). Tan et al.'s scheme remains vulnerable to watermark forgery, but its trained model satisfies robust verifiability even against a strong attacker (i.e., $q_1 = 0.8$ and $q_2 = 0.2$) since it sets a very low value for N'. Finally, the scheme by Guo and Potkonjak once more satisfies all three properties, despite its thresholds not being chosen with security considerations in mind. Once again, our approach enables the model owner to identify security-aware thresholds for each scheme, even against the most powerful model extraction attack we consider. For example, for Li et al.'s scheme on CIFAR10, under the strongest attacker ($q_1 = 0.8$, $q_2 = 0.2$, $k = 0.8$), our algorithm is able to find $N = 7{,}552$ and $N' = 472$ as a valid solution.

VI. Related Work

Trigger-based watermarking schemes [1], [2], [3] have been extensively studied, since they enable black-box verification of model ownership. Several schemes have been proposed [24], [6], [5], [7], [13], [29], [30], [4], [8], [23], [20], [31], [14], [21], [32], generally adhering to the pattern in Section II-B. Our work introduces a formal framework for the security analysis of such schemes and can successfully encode a relevant subset of them. We refer to Appendix A for the details about the considered schemes and their encoding.

The definition of existing trigger-based watermarking schemes often lacks formal foundations. As a consequence, even though watermarking schemes are empirically evaluated on a subset of security threats including watermark forgery [4], [21], [6], watermark suppression [5], [8], [20], [6], and water-

mark removal [23], [5], [13], [24], [6], prior work proposing comprehensive security evaluations highlighted that they still remain vulnerable to these attacks [9], [10]. However, even these evaluations are largely empirical and do not provide methods to improve the robustness grounded on a formal analysis. An exception is represented by Adi et al. [21], who proposed a formal framework for trigger-based watermarking, tailored for schemes employing cryptographic primitives, like the one they propose. They also formalized and proved security properties of their scheme within this framework. However, their proposal cannot be applied to trigger-based watermarking schemes which do not use cryptographic primitives, like those analyzed in this work. Shafieinejad et al. [11] extended the framework in [21] to generalize security against watermark removal to cover two other attackers, considering general trigger-based watermarking schemes without cryptographic primitives. However, they focus on a single security property (different from the ones we consider) and do not formalize existing schemes within their framework.

Existing watermarking schemes lack a standardized approach for setting their parameters, in particular N and N' in the verification procedure. This poses challenges for their usability and their security evaluation [10]. Some works leave this unspecified [24], [4], [20], while others offer only empirical approaches [13], [8], [10], [6]. More disciplined methods leverage statistical tests, like the t-test [7], [5], to set thresholds that statistically differentiate watermarked model behavior from non-watermarked models. However, these methods do not account for potential security threats, thus leaving schemes vulnerable. Through our proposed framework, instead, we show that it is possible to design a principled technique to set the scheme parameters while taking security into account, and our experiments confirm that our technique works in practice.

Finally, we also acknowledge an intriguing connection between trigger-based watermarking and *data auditing* frameworks [33], [34]. Data auditing is an important research area concerned with the detection of the use of personal data within the training process of ML models operated by third parties. Similar to trigger-based watermarking, data auditing may rely on the use of perturbed or specially marked inputs that introduce specific model behavior upon training; in turn, this induced behavior can be exercised in a black-box manner by checking whether the model output exhibits relevant evidence. As in trigger-based watermarking, data auditing assumes that the model is operated in an adversarial context, with the adversary doing their best to suppress the specific behavior that the user wants to expose. At the same time, there are significant differences. First, regarding the threat model, in trigger-based watermarking the training process is operated by the model owner, while in data auditing the training process is controlled by the attacker. This impacts the security analysis: for example, robustness against data cleaning prior to training is a relevant threat for data auditing, but not for trigger-based watermarking. Moreover, trigger-based watermarking and data auditing have very different objectives. Trigger-based watermarking is concerned with model ownership ("this model originates from me"), while data auditing is concerned with data provenance ("this model was trained on my data"). As such, the granularity of verification is different, because trigger-based watermarking establishes a global property of the model, while data auditing evaluates a property of a specific input of the model. We also note a related but conceptually distinct line of work on backdoor attacks (e.g., BadNets [35]), which also exploit trigger-induced behavior; however, in that setting, triggers are used maliciously by a third party who interferes with the training pipeline to compromise model integrity, rather than to intentionally establish model ownership or data provenance.

VII. Conclusion

In this paper, we addressed the critical gap in the formal understanding and security analysis of trigger-based watermarking. We introduced a novel and practical formal framework that uniformly captures existing trigger-based watermarking schemes. Using this framework, we proposed rigorous security definitions and revealed inherent weaknesses and flawed design assumptions of existing watermarking schemes. We then proposed a principled technique for setting watermarking scheme parameters to guarantee protection against security threats. Finally, we applied our framework to three selected watermarking schemes, experimentally demonstrating that their original parameter choices are often insufficient and leave them vulnerable to attacks. By applying our principled techniques, we showed that it is possible to identify sound parameter choices that significantly enhance security. This is as a first step toward a rigorous design methodology for trigger-based watermarking schemes, that complements the empirical practices that the previous work has followed so far.

As future work, we plan to investigate how to incorporate other types of attacks, e.g., operated by stateful adversaries, into our formalism. We also plan to extend our empirical evaluation to larger models, datasets, and practical adversarial threats like fine-tuning and parameter pruning. This would allow us to determine how to choose watermarking scheme parameters that ensure security against additional, real-world security threats. Moreover, we foresee that extending our formal framework to capture additional security properties and application scenarios, e.g., data auditing [33], [34], may be an interesting avenue for future work.

Acknowledgments

We thank the reviewers for their constructive feedback, which has greatly contributed to the improvement of this paper. This research was supported by project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU, by the European Union - Next-GenerationEU - PNRR – M.4 C.2, I.1.1 - PRIN 2022 WHAM!, 2022ZZX57L, H53D23003750006 and in part by the high-performance computing infrastructure developed under the project "CONVECS", funded by the PR Veneto FESR 2021-2027 program, Priority 1 – Specific Objective 1.1 – Action 1.1.2.

REFERENCES

[1] F. Boenisch, "A systematic review on model watermarking for neural networks," *Frontiers Big Data*, vol. 4, p. 729663, 2021. [Online]. Available: https://doi.org/10.3389/fdata.2021.729663

[2] Y. Li, H. Wang, and M. Barni, "A survey of deep neural network watermarking techniques," *Neurocomputing*, vol. 461, pp. 171–193, 2021. [Online]. Available: https://doi.org/10.1016/j.neucom.2021.07.051

[3] M. Xue, Y. Zhang, J. Wang, and W. Liu, "Intellectual property protection for deep learning models: Taxonomy, methods, attacks, and evaluations," *IEEE Trans. Artif. Intell.*, vol. 3, no. 6, pp. 908–923, 2022. [Online]. Available: https://doi.org/10.1109/TAI.2021.3133824

[4] J. Guo and M. Potkonjak, "Watermarking deep neural networks for embedded systems," in *Proceedings of the International Conference on Computer-Aided Design, ICCAD 2018, San Diego, CA, USA, November 05-08, 2018*, I. Bahar, Ed. ACM, 2018, p. 133. [Online]. Available: https://doi.org/10.1145/3240765.3240862

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

[6] Z. Li, C. Hu, Y. Zhang, and S. Guo, "How to prove your model belongs to you: a blind-watermark based framework to protect intellectual property of DNN," in *Proceedings of the 35th Annual Computer Security Applications Conference, ACSAC 2019, San Juan, PR, USA, December 09-13, 2019*, D. M. Balenson, Ed. ACM, 2019, pp. 126–137. [Online]. Available: https://doi.org/10.1145/3359789.3359801

[7] J. Tan, N. Zhong, Z. Qian, X. Zhang, and S. Li, "Deep neural network watermarking against model extraction attack," in *Proceedings of the 31st ACM International Conference on Multimedia, MM 2023, Ottawa, ON, Canada, 29 October 2023- 3 November 2023*, A. El-Saddik, T. Mei, R. Cucchiara, M. Bertini, D. P. T. Vallejo, P. K. Atrey, and M. S. Hossain, Eds. ACM, 2023, pp. 1588–1597. [Online]. Available: https://doi.org/10.1145/3581783.3612515

[8] P. Lv, H. Ma, K. Chen, J. Zhou, S. Zhang, R. Liang, S. Zhu, P. Li, and Y. Zhang, "Mea-defender: A robust watermark against model extraction attack," in *IEEE Symposium on Security and Privacy, SP 2024, San Francisco, CA, USA, May 19-23, 2024*. IEEE, 2024, pp. 2515–2533. [Online]. Available: https://doi.org/10.1109/SP54263.2024.00099

[9] S. Lee, W. Song, S. Jana, M. Cha, and S. Son, "Evaluating the robustness of trigger set-based watermarks embedded in deep neural networks," *IEEE Trans. Dependable Secur. Comput.*, vol. 20, no. 4, pp. 3434–3448, 2023. [Online]. Available: https://doi.org/10.1109/TDSC.2022.3196790

[10] N. Lukas, E. Jiang, X. Li, and F. Kerschbaum, "Sok: How robust is image classification deep neural network watermarking?" in *43rd IEEE Symposium on Security and Privacy, SP 2022, San Francisco, CA, USA, May 22-26, 2022*. IEEE, 2022, pp. 787–804. [Online]. Available: https://doi.org/10.1109/SP46214.2022.9833693

[11] M. Shafieinejad, N. Lukas, J. Wang, X. Li, and F. Kerschbaum, "On the robustness of backdoor-based watermarking in deep neural networks," in *IH&MMSec '21: ACM Workshop on Information Hiding and Multimedia Security, Virtual Event, Belgium, June, 22-25, 2021*, D. Borghys, P. Bas, L. Verdoliva, T. Pevný, B. Li, and J. Newman, Eds. ACM, 2021, pp. 177–188. [Online]. Available: https://doi.org/10.1145/3437880.3460401

[12] S. Shao, Y. Li, H. Yao, Y. He, Z. Qin, and K. Ren, "Explanation as a watermark: Towards harmless and multi-bit model ownership verification via watermarking feature attribution," in *32nd Annual Network and Distributed System Security Symposium, NDSS 2025, San Diego, California, USA, February 24-28, 2025*. The Internet Society, 2025. [Online]. Available: https://www.ndss-symposium.org/ndss-paper/explanation-as-a-watermark-towards-harmless-and-multi-bit-model-ownership-verification-via-watermarking-feature-attribution/

[13] Z. Ye, X. Zhang, and G. Feng, "Deep neural networks watermark via universal deep hiding and metric learning," *Neural Comput. Appl.*, vol. 36, no. 13, pp. 7421–7438, 2024. [Online]. Available: https://doi.org/10.1007/s00521-024-09469-5

[14] M. Li, Q. Zhong, L. Y. Zhang, Y. Du, J. Zhang, and Y. Xiang, "Protecting the intellectual property of deep neural networks with watermarking: The frequency domain approach," in *19th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2020, Guangzhou, China, December 29, 2020 - January 1, 2021*, G. Wang, R. K. L. Ko, M. Z. A. Bhuiyan, and Y. Pan, Eds. IEEE, 2020, pp. 402–409. [Online]. Available: https://doi.org/10.1109/TrustCom50675.2020.00062

[15] T. Orekondy, B. Schiele, and M. Fritz, "Knockoff nets: Stealing functionality of black-box models," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. Computer Vision Foundation / IEEE, 2019, pp. 4954–4963. [Online]. Available: http://openaccess.thecvf.com/content_CVPR_2019/html/Orekondy_Knockoff_Nets_Stealing_Functionality_of_Black-Box_Models_CVPR_2019_paper.html

[16] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction apis," in *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016*, T. Holz and S. Savage, Eds. USENIX Association, 2016, pp. 601–618. [Online]. Available: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/tramer

[17] L. Gao, W. Liu, K. Liu, and J. Wu, "Augsteal: Advancing model steal with data augmentation in active learning frameworks," *IEEE Trans. Inf. Forensics Secur.*, vol. 19, pp. 4728–4740, 2024. [Online]. Available: https://doi.org/10.1109/TIFS.2024.3384841

[18] X. Chen, W. Wang, C. Bender, Y. Ding, R. Jia, B. Li, and D. Song, "REFIT: A unified watermark removal framework for deep learning systems with limited data," in *ASIA CCS '21: ACM Asia Conference on Computer and Communications Security, Virtual Event, Hong Kong, June 7-11, 2021*, J. Cao, M. H. Au, Z. Lin, and M. Yung, Eds. ACM, 2021, pp. 321–335. [Online]. Available: https://doi.org/10.1145/3433210.3453079

[19] X. Liu, F. Li, B. Wen, and Q. Li, "Removing backdoor-based watermarks in neural networks with limited data," in *25th International Conference on Pattern Recognition, ICPR 2020, Virtual Event / Milan, Italy, January 10-15, 2021*. IEEE, 2020, pp. 10 149–10 156. [Online]. Available: https://doi.org/10.1109/ICPR48806.2021.9412684

[20] R. Namba and J. Sakuma, "Robust watermarking of neural network with exponential weighting," in *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security, AsiaCCS 2019, Auckland, New Zealand, July 09-12, 2019*, S. D. Galbraith, G. Russello, W. Susilo, D. Gollmann, E. Kirda, and Z. Liang, Eds. ACM, 2019, pp. 228–240. [Online]. Available: https://doi.org/10.1145/3321705.3329808

[21] Y. Adi, C. Baum, M. Cissé, B. Pinkas, and J. Keshet, "Turning your weakness into a strength: Watermarking deep neural networks by backdooring," in *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018*, W. Enck and A. P. Felt, Eds. USENIX Association, 2018, pp. 1615–1631. [Online]. Available: https://www.usenix.org/conference/usenixsecurity18/presentation/adi

[22] L. Fan, K. W. Ng, and C. S. Chan, "Rethinking deep neural network ownership verification: Embedding passports to defeat ambiguity attacks," in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, Eds., 2019, pp. 4716–4725. [Online]. Available: https://proceedings.neurips.cc/paper/2019/hash/75455e062929d32a333868084286bb68-Abstract.html

[23] B. D. Rouhani, H. Chen, and F. Koushanfar, "Deepsigns: An end-to-end watermarking framework for ownership protection of deep neural networks," in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS 2019, Providence, RI, USA, April 13-17, 2019*, I. Bahar, M. Herlihy, E. Witchel, and A. R. Lebeck, Eds. ACM, 2019, pp. 485–497. [Online]. Available: https://doi.org/10.1145/3297858.3304051

[24] Z. Xi, Z. Qu, W. Lu, X. Luo, and X. Cao, "Invisible DNN watermarking against model extraction attack," *IEEE Trans. Cybern.*, vol. 55, no. 2, pp. 800–811, 2025. [Online]. Available: https://doi.org/10.1109/TCYB.2024.3514838

[25] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *CoRR*, vol. abs/1708.07747, 2017. [Online]. Available: http://arxiv.org/abs/1708.07747

[26] A. Krizhevsky, "Learning multiple layers of features from tiny images," 2009. [Online]. Available: https://api.semanticscholar.org/CorpusID:18268744

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision*

and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016. IEEE Computer Society, 2016, pp. 770–778. [Online]. Available: https://doi.org/10.1109/CVPR.2016.90

[28] "Artifact Formal Foundations Trigger-Based Watermarking," https://github.com/LorenzoCazzaro/formal-foundations-trigger-based-watermarking.git [Accessed: 01/05/2026].

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

[30] E. L. Merrer, P. Pérez, and G. Trédan, "Adversarial frontier stitching for remote neural network watermarking," *Neural Comput. Appl.*, vol. 32, no. 13, pp. 9233–9244, 2020. [Online]. Available: https://doi.org/10.1007/s00521-019-04434-z

[31] B. Kim, S. Lee, S. Lee, S. Son, and S. J. Hwang, "Margin-based neural network watermarking," in *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, ser. Proceedings of Machine Learning Research, A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, Eds., vol. 202. PMLR, 2023, pp. 16 696–16 711. [Online]. Available: https://proceedings.mlr.press/v202/kim23o.html

[32] M. Pautov, N. Bogdanov, S. Pyatkin, O. Rogov, and I. V. Oseledets, "Probabilistically robust watermarking of neural networks," in *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI 2024, Jeju, South Korea, August 3-9, 2024*. ijcai.org, 2024, pp. 4778–4787. [Online]. Available: https://www.ijcai.org/proceedings/2024/528

[33] Z. Huang, N. Z. Gong, and M. K. Reiter, "A general framework for data-use auditing of ML models," in *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security, CCS 2024, Salt Lake City, UT, USA, October 14-18, 2024*, B. Luo, X. Liao, J. Xu, E. Kirda, and D. Lie, Eds. ACM, 2024, pp. 1300–1314. [Online]. Available: https://doi.org/10.1145/3658644.3690226

[34] Z. Chen and K. Pattabiraman, "Anonymity unveiled: A practical framework for auditing data use in deep learning models," in *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security, CCS 2025, Taipei, Taiwan, October 13-17, 2025*, C. Huang, J. Chen, S. Shieh, D. Lie, and V. Cortier, Eds. ACM, 2025, pp. 513–527. [Online]. Available: https://doi.org/10.1145/3719027.3744794

[35] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg, "Badnets: Evaluating backdooring attacks on deep neural networks," *IEEE Access*, vol. 7, pp. 47 230–47 244, 2019. [Online]. Available: https://doi.org/10.1109/ACCESS.2019.2909068

APPENDIX

A. Encoding Existing Watermarking Schemes

To demonstrate the generality and expressiveness of our formalism, we consider a number of different watermarking schemes from the literature and show how they can be encoded by instantiating the different parameters of our model. Below, we provide details about the encoding and summarize it in Table II, along with the notation used. Note that Guo and Potkonjak [4] propose the only watermarking scheme that employs $\mathcal{D}_{test}$, and consequently the only one with a meaningful, non-negative threshold N in the ownership verification procedure. Therefore, we will discuss how N is set only for this specific scheme.

a) Xi et al. [24]: define the watermark as $\omega = \langle \rho, y_t \rangle$ where $\rho : \mathcal{X} \to \mathcal{X}$ is a transformation of the input instance, and $y_t \in \mathcal{Y}$ is a predefined target label. ρ adds a perturbation to the input instance that depends on the model to be watermarked. The objective of the transformation is to make the instance in output imperceptibly different from the input instance, while being classified as belonging to the target class y_t by the watermarked model (more details in [24]). The trigger set is then obtained by applying ρ to instances of the training set and replacing their labels with y_t. The scheme provides no rules for setting the number of instances in $\mathcal{D}^*_{trigger}$, but in the experiments presented in the paper the trigger instances are sampled from a trigger set containing a number of instances equal to 10% of the instances in $\mathcal{D}_{train}$. Finally, the threshold N' is not set and no explicit recommendation is proposed.

b) Lv et al. [8]: define a watermark $\omega = \langle \psi, y_t, y_1, y_2 \rangle$, where $\psi : \mathcal{X} \times \mathcal{X} \to \mathcal{X}$ is a transformation of input instances that takes two instances and generates a new instance using a technique or a combination of techniques from a predefined set of input-mixing techniques. The labels $y_t, y_1, y_2 \in \mathcal{Y}$ are chosen such that all three are distinct from each other. More formally, given a pair of instances from $\mathcal{D}_{train}$ or $\mathcal{D}_{test}$ denoted by $(\langle \vec{x}_1, y_1 \rangle, \langle \vec{x}_2, y_2 \rangle)$, where $y_1 \neq y_2$, we obtain a new instance $\langle \psi(\vec{x}_1, \vec{x}_2), y_t \rangle$, where $y_t \neq y_1 \wedge y_t \neq y_2$, that will belong to the trigger set. Note that only instances in the trigger set generated from instances in $\mathcal{D}_{train}$ are used in the EMBED algorithm, while the instances of the trigger set used in the VERIFY algorithm are generated from instances in $\mathcal{D}_{test}$. No indication is provided on how to set the number of instances of $\mathcal{D}^*_{trigger}$, but, in the experimental section of the paper, at least 1,000 instances have been used. Finally, N' is empirically set to 30% of $|\mathcal{D}^*_{trigger}|$. The intuition is that the accuracy of the watermarked model on trigger instances should significantly exceed this threshold, while its accuracy on false trigger instances (those created with different source labels or input transformations) should fall below it to avoid false claims of ownership. Similarly, the accuracy of a non-watermarked model on trigger instances should also be below this threshold. The authors determined the value of the threshold by measuring, across 200 clean and watermarked

TABLE II: Encoding of existing watermarking schemes in our formal model. $\vec{m} \in \mathcal{X}$ denotes a perturbation vector with the same dimensionality as the instances in the feature space. y_t and y_s represent a target label and a source label, respectively. X is a random variable representing an instance sampled from the feature space, and Y is a random variable representing a label sampled from the set of labels. $\rho : \mathcal{X} \to \mathcal{X}$ denotes a function that transforms an instance in the feature space into another instance, $\psi : \mathcal{X} \times \mathcal{X} \to \mathcal{X}$ denotes a function that transforms a pair of instances from the feature space into a single instance, $\phi : \mathcal{Y} \to \mathcal{Y}$ denotes a mapping between labels, and $\delta : \mathcal{X} \to \{0, 1\}$ denotes a decision function that takes an instance as input.

Watermarking scheme	ω	Uses $\mathcal{D}_{test}$?	$\mathcal{D}_{trigger}$	N	N'
Xi et al. [24]	$\langle \rho, y_t \rangle$	✗	$\{\langle \rho(\vec{x}), y_t \rangle \mid \langle \vec{x}, y \rangle \in \mathcal{D}_{train}\}$	Unused	Unspecified
Lv et al. [8]	$\langle \psi, y_t, y_1, y_2 \rangle$	✗	$\{\langle \psi(\vec{x_1}, \vec{x_2}), y_t \rangle \mid \langle \vec{x_1}, y_1 \rangle, \langle \vec{x_2}, y_2 \rangle \in \mathcal{D}_{test} \wedge y_1, y_2, y_t$ pairwise distinct$\}$	Unused	Empirical (30%)
Ye et al. [13]	$\langle \vec{m}, y_t, y_s \rangle$	✗	$\{\langle \vec{x} + \vec{m}, y_t \rangle \mid \langle \vec{x}, y \rangle \in \mathcal{D}_{train} \wedge y = y_s\}$	Unused	Empirical (60%)
Tan et al. [7]	$\langle \rho, y_t \rangle$	✗	$\{\langle \rho(\vec{x}), y_t \rangle \mid \langle \vec{x}, y \rangle \notin \mathcal{D}_{train}\}$	Unused	t-test
Jia et al. (in-distribution) [5]	$\langle \rho, y_t, y_s \rangle$	✗	$\{\langle \rho(\vec{x}), y_t \rangle \mid \langle \vec{x}, y \rangle \in \mathcal{D}_{train} \wedge y = y_s\}$	Unused	t-test
Jia et al. (out-of-distribution) [5]	$\langle \rho, y_t, y_s \rangle$	✗	$\{\langle \rho(\vec{x}), y_t \rangle \mid \langle \vec{x}, y \rangle \notin \mathcal{D}_{train} \wedge y = y_s\}$	Unused	t-test
Rouhani et al. [23]	$\langle X, Y, \delta \rangle$	✗	$\{\langle \vec{x}, y \rangle \mid \vec{x} \leftarrow X \wedge \delta(\vec{x}) = 1 \wedge y \leftarrow Y\}$	Unused	binomial
Namba and Sakuma [20]	Y	✗	$\{\langle \vec{x}, y_w \rangle \mid \langle \vec{x}, y \rangle \in \mathcal{D}_{train} \wedge y_w \leftarrow Y \wedge y_w \neq y\}$	Unused	Unspecified
Li et al. [6]	$\langle \rho, Y \rangle$	✗	$\{\langle \rho(\vec{x}), y_w \rangle \mid \langle \vec{x}, y \rangle \in \mathcal{D}_{train} \wedge y_w \leftarrow Y\}$	Unused	Empirical ($\sim$ 90%)
Guo and Potkonjak [4]	$\langle \vec{m}, \phi \rangle$	✓	$\{\langle \vec{x} + \vec{m}, \phi(y) \rangle \mid \langle \vec{x}, y \rangle \in \mathcal{D}_{test}\}$	Unspecified	Unspecified

models, the accuracy on trigger instances and on generated false trigger instances, confirming it met these requirements.

c) Ye et al. [13]: define the watermark as $\omega = \langle \vec{m}, y_t, y_s \rangle$, where $\vec{m}$ is a learned perturbation, $y_t \in \mathcal{Y}$ is the target label and $y_s \in \mathcal{Y}$ is the source label. $\vec{m}$ is learned from a secret instance $\vec{x_k}$ of the model owner through the application of a U-net, which is trained to learn a subtle and invisible perturbation recognizable by the watermarked model. Then, the trigger set is obtained by adding the perturbation to samples of $\mathcal{D}_{train}$ whose label is y_s and assigning them the label y_t. No indication is provided on how to set the size of $\mathcal{D}^*_{trigger}$. However, in the experimental part of the paper, 100 trigger instances have been used. Finally, the threshold N' is set to 60% of $|\mathcal{D}^*_{trigger}|$ without providing a precise motivation for this choice.

d) Tan et al. [7]: define the watermark $\omega = \langle \rho, y_t \rangle$ where $\rho : \mathcal{X} \to \mathcal{X}$ is a function that takes an input instance and returns the "optimized" instance, while $y_t \in \mathcal{Y}$ is the target label. $\mathcal{D}_{trigger}$ is generated in three steps. First, some instances from a specific class are sampled from a dataset unrelated to the original task and to $\mathcal{D}_{train}$. Then, each instance of $\mathcal{D}_{trigger}$ is associated with a class y_t of $\mathcal{Y}$, specifically the class on which a non-watermarked model trained on the original task obtains the worst classification accuracy. Finally, ρ transforms each instance of $\mathcal{D}_{trigger}$ jointly with the training of the watermarked model. In particular, the transformation operates in a way such that other surrogate models, extracted from the watermarked model, classify the instances of $\mathcal{D}_{trigger}$ as y_t, while a non-watermarked model does not classify them as y_t. The scheme does not suggest a disciplined way to set the size of $\mathcal{D}^*_{trigger}$. However, in the experimental part of the paper, the authors use 100 instances of $\mathcal{D}^*_{trigger}$. Finally, the threshold N' is set by using the one-sample one-tailed t-test based on the accuracy of some non-watermarked models on $\mathcal{D}_{trigger}$ and at some confidence level.

e) Jia et al. [5]: define the watermark as $\omega = \langle \rho, y_t, y_s \rangle$, where $\rho : \mathcal{X} \to \mathcal{X}$ is a transformation applied to some instances, y_s is the source label and $y_t \in \mathcal{Y}$ is the target label. The watermarking scheme supports both in-distribution and out-of-distribution watermarking, i.e., building a trigger set using instances from the same distribution as $\mathcal{D}_{train}$ or from a different distribution. In the case of in-distribution watermarking, the instances used to generate $\mathcal{D}_{trigger}$ are samples of class $y_s \in \mathcal{Y}$ drawn from $\mathcal{D}_{train}$. In contrast, in the case of out-of-distribution watermarking, the instances used to generate $\mathcal{D}_{trigger}$ are samples of class $y_s \notin \mathcal{Y}$, taken from a dataset associated with a different task. The function ρ is responsible for positioning the pre-defined input mask $\vec{m} \in \mathcal{X}$ (the same for all the instances in $\mathcal{D}_{trigger}$) onto the input instances and modifying the instances to decrease the confidence of the watermarked model in predicting the target class, thereby making the watermarking more robust (see Algorithm 1 in [5] for more details). The scheme does not specify how many instances should be included in $\mathcal{D}^*_{trigger}$. Finally, the threshold N' is set by computing the number of instances needed to reject the null hypothesis of a t-test at the 95% confidence level. The assumption is that the accuracy of the watermarked model on the trigger set is the mean of a binomial distribution characterizing if watermarked data is correctly classified. According to the Central Limit Theorem, it is normally distributed when the number of queries, $|\mathcal{D}^*_{trigger}|$, is greater than 30. The null hypothesis of the statistical test states that this accuracy is less than or equal the false watermark rate, i.e., the accuracy of a non-watermarked model on the trigger set. If the null hypothesis is rejected, then the behavior of the watermarked model is peculiar on $\mathcal{D}^*_{trigger}$ and not due to chance.

f) Rouhani et al. [23]: [3] define the watermark as $\omega = \langle X, Y, \delta \rangle$ where X is a random variable representing a randomly sampled instance from $\mathcal{X}$, Y is a random variable representing a randomly sampled label from $\mathcal{Y}$ and $\delta : \mathcal{X} \to \{0, 1\}$

[3]This paper proposes two watermarking schemes: one for embedding a watermark in the intermediate layers of a neural network, and another for embedding it in the output layer. We will encode the second scheme within our formalism, since it uses a trigger set and verifies model ownership as defined in our framework.

is a decision function. Trigger candidates are sampled as values of X and assigned random labels drawn from Y. A sampled instance is included in $\mathcal{D}_{trigger}$ if it lies in a "low-density region" of the model to be watermarked (see [23] for more details). This check is represented by the decision function δ, that returns 1 if the condition is satisfied on the input instance, otherwise 0. The scheme does not provide any specification about how to set the size of $\mathcal{D}^*_{trigger}$. In the experimental section of the paper, 20 and 30 instances are used. Finally, the threshold N' is set as the quantile of a binomial distribution with respect to a fixed probability (e.g., 0.999, as used in the paper), and with the probability of success equal to $\frac{1}{|\mathcal{Y}|}$.

g) Namba and Sakuma [20]: define the watermark as $\omega = Y$, where Y is the random variable representing a label from $\mathcal{Y}$. Trigger instances are randomly sampled from $\mathcal{D}_{train}$. Then, the associated labels are sampled as values of Y and are different from the original labels of the instances. The scheme does not specify a precise number of instances to sample in $\mathcal{D}^*_{trigger}$. However, in the experimental part of the paper, 30, 20, 10 or 5 instances are used. Finally, the threshold N' is not set and no intuition is provided about how to set it.

h) Li et. al. [6]: define the watermark as $\omega = \langle \rho, Y \rangle$, where $\rho : \mathcal{X} \to \mathcal{X}$ is a transformation of input instances and Y is a random variable representing a label $y \in \mathcal{Y}$. ρ is realized by an encoder network that embeds a logo $\vec{m}$, i.e., an image, not necessarily belonging to $\mathcal{X}$, inside the input instance to generate an instance that is as similar as possible to the original instance but distinguishable by the watermarked model. The trigger set is defined by applying ρ to instances of $\mathcal{D}_{train}$ and assigning the label y_w, sampled from Y, to the resulting instances. The scheme does not suggest a specific way to set the size of $\mathcal{D}^*_{trigger}$. However, in the experimental part of the paper, 1% of the training data is used. Finally, N' is not set, but the paper states that N' should be close to $|\mathcal{D}^*_{trigger}|$.

i) Guo and Potkonjak [4]: define the watermark as $\omega = \langle \vec{m}, \phi \rangle$, where $\vec{m} \in \mathcal{X}$ is a vector and $\phi : \mathcal{Y} \to \mathcal{Y}$ is an irreflexive class mapping. $\vec{m}$ represents the signature of the model owner, while ϕ specifies the substitute label for each original label. The trigger set used in the verification algorithm is then obtained by adding the perturbation to the instances of $\mathcal{D}_{test}$ and substituting their original labels through ϕ. The same procedure is applied to the instances of $\mathcal{D}_{train}$ to generate the trigger set used in the training of the model to be watermarked. The size of $\mathcal{D}^*_{trigger}$ is set as the same of $\mathcal{D}_{trigger}$. $\mathcal{D}_{verify}$ consists of the union of $\mathcal{D}^*_{trigger}$ and $\mathcal{D}^*_{test}$. Finally, there is no formal definition or suggestion about how to set N and N' in the specification of the watermarking scheme. However, the authors provide a quantitative analysis estimating, for a given accuracy of the model on the test set and the trigger set, the probability that the number of errors on each set stays below a common bound.

B. Examples of Principled Parameter Setting

We show our approach to parameter setting at work for a model M such that $a(M, \mathcal{D}_{test}) = 0.9$, $a(M, \mathcal{D}_{trigger}) = 0.7$, $|\mathcal{D}^*_{test}| = |\mathcal{D}^*_{trigger}| = 400$ and $|\mathcal{Y}| = 2$. We consider watermark suppression as the attacker for robust verifiability, parameterized by $q \in \{0.5, 0.6, 0.7, 0.8, 0.9\}$, and forgery as the attacker for unforgeability, parameterized by $k \in \{0.35, 0.42, 0.49, 0.56, 0.63\}$, where the values of k have been set to gradually approach $a(M, \mathcal{D}_{trigger}) = 0.7$.

TABLE III: Values of q and k for which the optimization algorithm succeeds (checkmark) or fails (cross) in finding a pair of values N and N' that satisfies the constraints of the three properties, assuming $\varepsilon = 0.03$.

q / k	**0.35**	**0.42**	**0.49**	**0.56**	**0.63**
0.5	✓	✓	✓	✓	✗
0.6	✓	✓	✓	✓	✗
0.7	✓	✓	✓	✓	✗
0.8	✓	✓	✓	✓	✗
0.9	✓	✓	✗	✗	✗

TABLE IV: Values of q and k for which the optimization algorithm succeeds (checkmark) or fails (cross) in finding a pair of values N and N' that satisfies the constraints of the three properties, assuming $\varepsilon = 0.1$.

q / k	**0.35**	**0.42**	**0.49**	**0.56**	**0.63**
0.5	✓	✓	✓	✓	✓
0.6	✓	✓	✓	✓	✓
0.7	✓	✓	✓	✓	✓
0.8	✓	✓	✓	✓	✓
0.9	✓	✓	✓	✗	✗

Table III reports for several combinations of the parameters whether our approach finds a solution satisfying the security constraints for $\varepsilon = 0.03$. The table indicates that a solution exists for various combinations of q and k. As an example, consider $q = 0.5$ and $k = 0.56$, i.e., we want to satisfy all the intended security properties with probability 0.97 against an attacker who is able to successfully discriminate between test and trigger instances with probability 0.5 and forge a trigger set $\mathcal{D}'_{trigger}$ such that $a(M, \mathcal{D}'_{trigger}) = 0.56$. In this case, the algorithm returns $N = 298$ and $N' = 262$ as a possible solution of the problem. Observe that a few combinations of q and k do not enable any solution, e.g., if k increases to 0.63. To admit solutions under these stronger attackers, we must increase ε and accept lower security guarantees. We report the cases in which the proposed approach succeeds for $\varepsilon = 0.1$ in Table IV. The approach can now find solutions for additional cases where $k = 0.63$. For example, setting $k = 0.63$ and $q = 0.5$ leads to the following solution: $N = 292$ and $N' = 268$. However, for more powerful attackers with $k = 0.63$ and $q = 0.9$, lower security guarantees (higher ε) are still required to find a solution.

C. Additional Experimental Results

Table V reports the results of our experiments when considering model extraction as the attacker for robust verifiability. We set $q_1 \in \{0.5, 0.8\}$, $q_2 \in \{0.2, 0.8\}$, and $k \in \{0.5, 0.8\}$.

TABLE V: Results of our formal analysis of three existing watermarking schemes. Here we consider model extraction as the attacker for robust verifiability. We report the size of $\mathcal{D}^*_{test}$, $\mathcal{D}^*_{trigger}$, as well as the values of N and N' obtained following the original papers in the **Empirical** column. We then indicate if our three security properties hold for the values of N and N' set following the original papers, for different values of q_1, q_2 and k. Finally, in the **Formal** column, we present the values found by our optimization algorithm for various values of q_1, q_2 and k.

Scheme	Dataset	$a(M, \mathcal{D}_{test})$	$a(M, \mathcal{D}_{trigger})$	Empirical	q_1	q_2	k	Ver.	Rob. Ver.	Unf.	Formal
Guo and Potkonjak	FMNIST	0.90	0.91	$\lvert\mathcal{D}^*_{test}\rvert = 10{,}000$ $\lvert\mathcal{D}^*_{trigger}\rvert = 10{,}000$ $N = 8{,}500$ $N' = 8{,}500$	0.5	0.2	0.5	✓	✓	✓	$N = 5{,}083$, $N' = 9{,}052$
					0.8	0.2	0.5	✓	✓	✓	$N = 7{,}473$, $N' = 9{,}052$
					0.5	0.8	0.5	✓	✓	✓	$N = 5{,}083$, $N' = 9{,}052$
					0.8	0.8	0.5	✓	✓	✓	$N = 7{,}473$, $N' = 9{,}052$
					0.5	0.2	0.8	✓	✓	✓	$N = 5{,}083$, $N' = 9{,}052$
					0.8	0.2	0.8	✓	✓	✓	$N = 7{,}473$, $N' = 9{,}052$
					0.5	0.8	0.8	✓	✓	✓	$N = 5{,}083$, $N' = 9{,}052$
					0.8	0.8	0.8	✓	✓	✓	$N = 7{,}473$, $N' = 9{,}052$
	CIFAR10	0.89	0.92	$\lvert\mathcal{D}^*_{test}\rvert = 10{,}000$ $\lvert\mathcal{D}^*_{trigger}\rvert = 10{,}000$ $N = 8{,}500$ $N' = 8{,}500$	0.5	0.2	0.5	✓	✓	✓	$N = 5{,}033$, $N' = 9{,}155$
					0.8	0.2	0.5	✓	✓	✓	$N = 7{,}393$, $N' = 9{,}155$
					0.5	0.8	0.5	✓	✓	✓	$N = 5{,}033$, $N' = 9{,}155$
					0.8	0.8	0.5	✓	✓	✓	$N = 7{,}393$, $N' = 9{,}155$
					0.5	0.2	0.8	✓	✓	✓	$N = 5{,}033$, $N' = 9{,}155$
					0.8	0.2	0.8	✓	✓	✓	$N = 7{,}393$, $N' = 9{,}155$
					0.5	0.8	0.8	✓	✓	✓	$N = 5{,}033$, $N' = 9{,}155$
					0.8	0.8	0.8	✓	✓	✓	$N = 7{,}393$, $N' = 9{,}155$
Li et al.	FMNIST	0.90	0.99	$\lvert\mathcal{D}^*_{test}\rvert = 10{,}000$ $\lvert\mathcal{D}^*_{trigger}\rvert = 600$ $N = -1$ $N' = 540$	0.5	0.2	0.5	✓	✗	✓	$N = 5{,}083$, $N' = 589$
					0.8	0.2	0.5	✓	✗	✓	$N = 7{,}473$, $N' = 589$
					0.5	0.8	0.5	✓	✗	✓	$N = 5{,}083$, $N' = 589$
					0.8	0.8	0.5	✓	✗	✓	$N = 7{,}473$, $N' = 589$
					0.5	0.2	0.8	✓	✗	✓	$N = 5{,}083$, $N' = 589$
					0.8	0.2	0.8	✓	✗	✓	$N = 7{,}473$, $N' = 589$
					0.5	0.8	0.8	✓	✗	✓	$N = 5{,}083$, $N' = 589$
					0.8	0.8	0.8	✓	✗	✓	$N = 7{,}473$, $N' = 589$
	CIFAR10	0.91	0.96	$\lvert\mathcal{D}^*_{test}\rvert = 10{,}000$ $\lvert\mathcal{D}^*_{trigger}\rvert = 500$ $N = -1$ $N' = 450$	0.5	0.2	0.5	✓	✗	✓	$N = 5{,}133$, $N' = 472$
					0.8	0.2	0.5	✓	✗	✓	$N = 7{,}552$, $N' = 472$
					0.5	0.8	0.5	✓	✗	✓	$N = 5{,}133$, $N' = 472$
					0.8	0.8	0.5	✓	✗	✓	$N = 7{,}552$, $N' = 472$
					0.5	0.2	0.8	✓	✗	✓	$N = 5{,}133$, $N' = 472$
					0.8	0.2	0.8	✓	✗	✓	$N = 7{,}552$, $N' = 472$
					0.5	0.8	0.8	✓	✗	✓	$N = 5{,}133$, $N' = 472$
					0.8	0.8	0.8	✓	✗	✓	$N = 7{,}552$, $N' = 472$
Tan et al.	FMNIST	0.92	0.99	$\lvert\mathcal{D}^*_{test}\rvert = 10{,}000$ $\lvert\mathcal{D}^*_{trigger}\rvert = 100$ $N = -1$ $N' = 11$	0.5	0.2	0.5	✓	✓	✗	$N = 5{,}183$, $N' = 97$
					0.8	0.2	0.5	✓	✓	✗	$N = 7{,}631$, $N' = 97$
					0.5	0.8	0.5	✓	✓	✗	$N = 5{,}183$, $N' = 97$
					0.8	0.8	0.5	✓	✓	✗	$N = 7{,}631$, $N' = 97$
					0.5	0.2	0.8	✓	✓	✗	$N = 5{,}183$, $N' = 97$
					0.8	0.2	0.8	✓	✓	✗	$N = 7{,}631$, $N' = 97$
					0.5	0.8	0.8	✓	✓	✗	$N = 5{,}183$, $N' = 97$
					0.8	0.8	0.8	✓	✓	✗	$N = 7{,}631$, $N' = 97$
	CIFAR10	0.88	0.99	$\lvert\mathcal{D}^*_{test}\rvert = 10{,}000$ $\lvert\mathcal{D}^*_{trigger}\rvert = 100$ $N = -1$ $N' = 4$	0.5	0.2	0.5	✓	✓	✗	$N = 4{,}983$, $N' = 97$
					0.8	0.2	0.5	✓	✓	✗	$N = 7{,}314$, $N' = 97$
					0.5	0.8	0.5	✓	✓	✗	$N = 4{,}983$, $N' = 97$
					0.8	0.8	0.5	✓	✓	✗	$N = 7{,}314$, $N' = 97$
					0.5	0.2	0.8	✓	✓	✗	$N = 4{,}983$, $N' = 97$
					0.8	0.2	0.8	✓	✓	✗	$N = 7{,}314$, $N' = 97$
					0.5	0.8	0.8	✓	✓	✗	$N = 4{,}983$, $N' = 97$
					0.8	0.8	0.8	✓	✓	✗	$N = 7{,}314$, $N' = 97$