ChainMarks: Securing DNN Watermark with Cryptographic Chain

Brian Choi
Johns Hopkins University
bchoi11@jhu.edu

Shu Wang
Palo Alto Networks, Inc.
shuwang@paloaltonetworks.com

Isabelle Choi
University of California, Los Angeles
isabellechoi11@g.ucla.edu

Kun Sun
George Mason University
ksun3@gmu.edu

Abstract

With the widespread deployment of deep neural network (DNN) models, dynamic watermarking techniques are being used to protect the intellectual property of model owners. However, recent studies have shown that existing watermarking schemes are vulnerable to watermark removal and ambiguity attacks. Besides, the vague criteria for determining watermark presence further increase the likelihood of such attacks. In this paper, we propose a secure DNN watermarking scheme named ChainMarks, which generates secure and robust watermarks by introducing a cryptographic chain into the trigger inputs and utilizes a two-phase Monte Carlo method for determining watermark presence. First, ChainMarks generates trigger inputs as a watermark dataset by repeatedly applying a hash function over a secret key, where the target labels associated with trigger inputs are generated from the digital signature of model owner. Then, the watermarked model is produced by training a DNN over both the original and watermark datasets. To verify watermarks, we compare the predicted labels of trigger inputs with the target labels and determine ownership with a more accurate decision threshold that considers the classification probability of specific models. Experimental results show that ChainMarks exhibits higher levels of robustness and security compared to state-of-the-art watermarking schemes. With a better marginal utility, ChainMarks provides a higher probability guarantee of watermark presence in DNN models with the same level of watermark accuracy.

CCS Concepts

• **Security and privacy → Security services**.

Keywords

Deep Neural Networks, Watermarking, Cryptographic Chain

ACM Reference Format:
Brian Choi, Shu Wang, Isabelle Choi, and Kun Sun. 2025. ChainMarks: Securing DNN Watermark with Cryptographic Chain. In *ACM Asia Conference on Computer and Communications Security (ASIA CCS '25), August 25–29, 2025, Ha Noi, Vietnam.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/xxxxxxx.xxxxxxx

1 Introduction

Deep learning has shown its potential in multiple intelligent systems such as autonomous transportation, automated manufacture, and intelligent healthcare. However, the design and implementation of deep neural networks (DNNs) typically require significant resources for data collection, training, validation, and testing [58]. Because developing and possessing DNN models can provide a significant advantage, adversaries are highly motivated to steal the models for unauthorized use or resale. Therefore, it is crucial to protect the intellectual property (IP) of DNN models to avoid potential infringement; otherwise, it could impede the widespread deployment of these models.

Digital watermarking techniques are promising for safeguarding the intellectual property of DNN models by embedding covert information within the network for future verification. Similarly to traditional watermarking schemes designed for multimedia content, static watermarking techniques have been proposed to embed watermarks into the static parameters of DNN models (e.g., model weights) that are not changed during operation [10, 45, 72, 77]. However, static DNN watermark solutions imply a white-box model that needs to access the model parameters during verification, which is not practical to protect the intellectual property of DNNs.

To better protect the intellectual property of DNN models, researchers develop dynamic DNN watermarking techniques by deliberately training a DNN model on both the original dataset and a watermark dataset [1, 92]. By creating backdoors into the DNN model, these solutions output specific labels for a set of crafted inputs (i.e., watermark triggers). The over-parameterization of DNN models allows to plant the additional trigger inputs, i.e., watermarks, without affecting the overall classification accuracy of the DNN models. Trigger inputs can take the form of abstract images, adversarial examples, or inputs unrelated to the original task.

However, existing dynamic DNN watermarking schemes still face two challenges, namely, *vulnerability to multiple watermark attacks* and *vague criteria on watermark presence*. First, they are vulnerable to watermark removal attacks and watermark ambiguity attacks. A recent study [53] reveals the existing watermarking schemes are vulnerable to watermark removal attacks based on input preprocessing, model modification, or model extraction. Even worse, they are unable to resist watermark ambiguity attacks that allow attackers to forge additional watermarks to claim false DNN model ownership. One existing defense [21] attempts to defeat watermark ambiguity attacks by adding a secret called digital passport as an extra input to the DNN watermarking scheme; however, attackers may leverage adversarial learning to find out an alternative qualified digital passport. Moreover, this method is vulnerable to the watermark removal attacks that are based on input preprocessing. Second, existing watermarking schemes suffer from the unclear criteria for determining the presence of watermarks. The accuracy of empirical estimation method for the watermark decision threshold is limited in practice, as the threshold is dependent on various practical factors such as model classification properties, types of watermark patterns, and their probability distributions [53]. Also, the existing estimation method is incapable of calculating the threshold in certain scenarios, e.g., when the p-value is extremely small.

In this paper, we propose ChainMarks, a secure DNN watermarking scheme that can generate secure and robust watermarks by introducing *one-way cryptographic chain relationships* into the watermark trigger inputs and utilizing a *two-phase Monte Carlo estimation method* to determine the watermark presence. ChainMarks can efficiently defeat both watermark removal attacks and watermark ambiguity attacks by designing trigger inputs as a cryptographic chain. First, the noise-like/pseudo-random triggers perform better than other forms. They are far away from the data distribution of regular tasks and are hence robust against removal attacks via fine-tuning, model pruning, and input preprocessing. Second, the one-way chain property can stop the back-propagation algorithm of adversarial machine learning, which is used by watermark ambiguity attacks where attackers generate fake watermarks by satisfying multiple constraints (e.g., accuracy requirement). However, attackers cannot include hash functions in their optimization constraints and hence can only perform guessing attacks. Third, the accuracy of watermarks might not be 100% due to the model post-processing; thus, unclear criteria for watermark presence may increase the likelihood of attacks. To ensure detection accuracy, we develop a two-phase Monte Carlo method to enhance the estimation of watermark presence, by considering the probability distributions of both model classification and watermark generation.

ChainMarks consists of two main modules, namely, watermark generation and watermark verification. To generate watermarks, we first construct a watermark dataset that consists of trigger inputs and their target labels. The trigger inputs are generated as a cryptographically chained sequence by repeatedly applying a one-way hash function over a secret key, which is a random seed selected by the model owner. Then, the digital signature of model owner is transformed into a number in base C (that is, the number of classes), where each digit is assigned as the target label of a trigger input according to sequential order. By applying our cryptographic mechanism, the trigger inputs and their target labels are interrelated with the owner's seed key and digital signature, respectively. Therefore, an adversary is unable to apply optimization-based adversarial attacks, since the acquired trigger inputs can only satisfy either the consistency of predicted labels with target labels or the presence of cryptographical chain in trigger inputs, but not both. To embed the watermarks, we train a watermarked DNN model over both the original and the watermark datasets.

To verify the watermarks, the model owner provides the seed key to the verifier (e.g., a trusted third party) for regenerating the cryptographically chained trigger inputs, which are then fed into the DNN model to obtain the predicted labels. The verifier then calculates the Hamming distance between the predicted labels and the target labels extracted from the digital signature and determines the presence of watermarks based on a decision threshold, which is calculated via a two-phase Monte Carlo approach. In the first phase, the classification probability distribution of a DNN model is obtained by an empirical estimation method. In the second stage, our aim is to determine the number of additional random inputs required until the first hit occurs in the classes with the classification probability of zero. Deriving this threshold is equivalent to obtaining the probability distribution for the number of matching labels. The proposed two-phase Monte Carlo method enables us to obtain more accurate bounds on the distribution, since our method uses the real output distribution of DNN models instead of directly modeling the output classes with empirical probabilities. This advantage becomes more significant when the p-value is extremely small, as existing methods derive an output probability of zero.

To evaluate the robustness and security of ChainMarks, we conduct extensive experiments on the watermarked models trained on CIFAR-10 / CIFAR-100 datasets. Experiments show that our embedded watermarks do not affect the test accuracy of the original tasks. Compared with four state-of-the-art dynamic DNN watermarking schemes, ChainMarks shows higher robustness against 16 watermark removal attacks. Moreover, due to the introduction of a cryptographic chain, ChainMarks can resist the watermark ambiguity attack, which can easily bypass other schemes. By investigating the effects of p-values, our proposed threshold estimation method is applicable to the scenarios with smaller p-values (i.e., higher level of security). In addition, the marginal utility of ChainMarks is higher than that of other existing schemes, providing a higher probability guarantee of the watermark presence in the DNN models with the same level of watermark accuracy.

In summary, we make the following contributions:

- We propose a secure DNN watermarking scheme, ChainMarks, which introduces a cryptographic chain into trigger inputs to counter both watermark removal and watermark ambiguity attacks.
- We propose a new two-phase Monte Carlo method to estimate the decision threshold for watermark presence, providing a more accurate estimation and applying to the scenarios with higher security level.
- We conduct extensive experiments to prove the robustness and security of ChainMarks by comparing it with four state-of-the-art watermarking schemes against 17 watermark attacks over different models.

2 Background

2.1 Dynamic DNN Watermarking

DNN watermarking schemes are essential for protecting intellectual property rights and ensuring the integrity of the model by embedding watermarks into the model behaviors in response to a crafted set of trigger inputs. During watermark verification, model behaviors can be observed to verify the presence of watermarks [1, 25, 37, 66, 70, 91, 92]. In Figure 1, dynamic watermarking methods usually leverage DNN backdoors to generate trigger inputs, which are usually kept secret as they act as keys in the watermark embedding and verification processes.

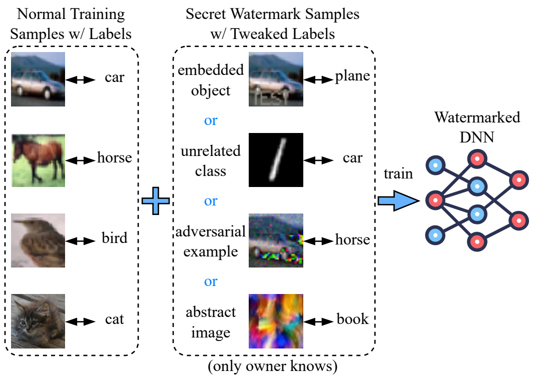


Figure 1: An example of dynamic deep neural network watermarking scheme [1, 92].

The fundamental requirements of an effective watermarking technique include fidelity, generality, efficiency, robustness, and security [46]. Fidelity ensures that the embedded watermarks do not significantly impact the model accuracy. Generality states that a watermarking technique should be applicable to different DNN architectures and datasets. The efficiency demands the overhead of watermark embedding and verification should be reasonable. Robustness demands that embedded watermarks should be accurate even with model post-processing, e.g., fine-tuning [77] and network pruning [95], which can be performed by an entity with access to internal details of DNN models. The security requirement ensures that a DNN watermarking scheme can withstand malicious attacks. An adversary may construct a surrogate model from the source model and then try to remove the embedded watermarks, replace the old watermarks with new ones, or fabricate fake watermarks to falsely claim ownership. A secure dynamic DNN watermarking scheme should be able to defeat all forms of watermark attacks.

2.2 One-Way Key Chain

A one-way key chain is constructed using a publicly known function H that is easy to compute, but computationally hard to invert [35]. Typically, the function H is selected as a cryptographic hash function, e.g., MD5 [65], SHA-1 [64], or SHA-256 [22]. A one-way key chain of length $L + 1$ is generated by iteratively applying H to an initial key K for L times, resulting in $\{K, H(K), H^2(K), ..., H^L(K)\}$, where $H^i(K)$ denotes the hash value of $H^{(i-1)}(K)$, $2 \le i \le L$. We can compute $H^i(K)$ from $H^{(i-1)}(K)$; however, inferring $H^{(i-1)}(K)$ from $H^i(K)$ would be infeasible. The one-way key chain is widely used in authentication [65] and wireless sensor networks [71].

3 Threat Model

We focus on protecting the intellectual property rights of DNN models via a secure dynamic watermarking technique. We assume that attackers have access to the model APIs to send queries and collect the outputs; thus, they may derive a surrogate model that approximates the watermarked model without knowledge of the secret watermark trigger inputs or target labels. A surrogate model with comparable accuracy to the source model effectively grants attackers access to a "white-box" version of the original model, including its parameters. Moreover, we assume that attackers have access either to unlabeled data from the same distribution [15] or to labeled data from any distribution. We further assume there is a trusted third party, which serves as the verifier to ascertain if the claimed watermarks are present in the given model. With access to the surrogate model, attackers may launch deep learning domain-specific attacks including watermark ambiguity attacks and watermark removal attacks.

Watermark Ambiguity Attacks. If an adversary successfully forges and embeds a second watermark into a watermarked model, there will be significant ambiguity with respect to model ownership, leading to false ownership claim. To compromise a watermarked model trained on the original dataset and trigger inputs, attackers can craft a new set of base trigger images with randomly assigned target labels. They then generate fake trigger images by adding trainable noise components, which have the same dimensions as the input images, to the base triggers. The attackers optimize a cross-entropy loss function between the target labels and the predicted labels of the fake trigger images. Given that attackers may construct surrogate models via transfer learning, existing DNN watermarking schemes remain vulnerable to watermark ambiguity attacks [21, 25, 36, 39]. In particular, backdoor-based watermarking methods embed trigger inputs into the unused space of DNNs while sharing the same classifier for the original task, exhibit inherent limitations in resisting watermark ambiguity attack [21].

Watermark Removal Attacks. A watermark removal attack involves using the source DNN model as input to generate a surrogate model as output. The objective is to eliminate embedded watermarks in the surrogate model while preserving a utility level comparable to that of the original model. In other words, the watermark retention rate in the surrogate model should be sufficiently low to prevent ownership claim with the source watermarks. There are three types of removal attacks [53]. In input preprocessing attacks, an attacker with white-box access intentionally modifies data samples before passing them to the surrogate model during watermark verification. In model modification attacks, an adversary with white-box knowledge may alter the internal parameters of the source model, typically through fine-tuning or pruning, to create a surrogate model. Model fine-tuning is a transfer learning technique that adjusts an already trained model to perform another related task; however, the original model parameters are modified during fine-tuning, thus disrupting the embedded watermarks [14]. Model pruning can set weights below a certain threshold to zero while maintaining the required model's accuracy; however, this process impact the watermark presence due to structural changes in the network. In model extraction attacks, an adversary trains a surrogate model by collecting input-output pairs from the source model, requiring only black-box access. The details of these three attack types are further discussed in Section 6.3.

4 ChainMarks Design

4.1 System Overview

Figure 2 shows the overall design of ChainMarks. Given a DNN model M_0, the model owner first embeds watermarks into M_0 to obtain the watermarked model M_w. Due to the white-box assumption, attackers can access both the architecture and weights of M_w. To remove existing watermarks or embed new watermarks, a pirate can perform watermark removal and watermark ambiguity attacks to deploy a surrogate model M_s. To protect intellectual property,

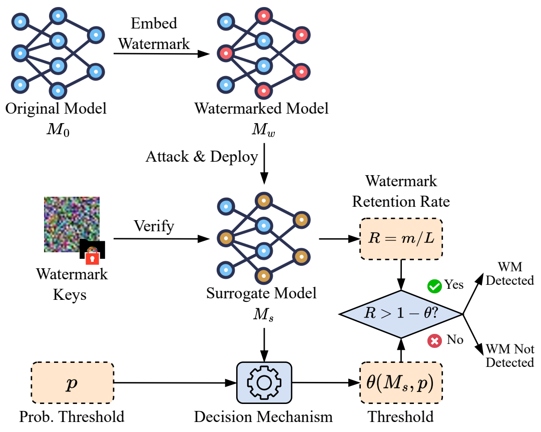


Figure 2: The overview of our ChainMarks schema.

the model owner can present watermark keys (i.e., the seed key to generate trigger inputs) to a trusted third party for verification of the claimed watermarks in M_s. Then the verifier calculates the watermark retention rate R by evaluating if m out of L triggers are matched. If R exceeds a retention threshold of $(1-\theta)$, the watermark is considered detected and ownership is successfully established; otherwise, the ownership claim fails. The detection threshold (θ) depends on both the surrogate model (M_s) and the acceptable success probability for a guessing attack (i.e., p-value).

In ChainMarks, we design watermarks based on a cryptographic chain, motivated by three hypotheses. First, one-way chain input can hinder the backpropagation algorithms used in adversarial machine learning, which underlie watermark ambiguity attacks. For example, watermarks (x_1, x_2) have labels (y_1, y_2). Attackers can apply optimization (adversarial ML) to find alternative inputs (x'_1, x'_2) that produce the same labels to falsely claim ownership. However, by introducing a chaining constraint, i.e., requiring x'_1, x'_2 satisfy $x'_1 = hash(x'_2)$, we dramatically increase the searching difficulty since adding a one-way function into optimization constraints is computationally infeasible. Second, we observe that noise-like/pseudo-random triggers are more effective than other formats. This is because, in feature space, they are far from the distribution of natural data and hence are robust against fine-tuning/retraining. Third, the digital signature can be employed to verify that the triggers are truly generated and owned by the original model creator.

4.2 Watermark Generation and Embedding

All known dynamic watermarking schemes are vulnerable to watermark ambiguity attacks, due to the absence of a robust authentication mechanism for the trigger inputs. To address this issue, we introduce additional cryptographic requirements for both the trigger inputs and the corresponding watermarked model. Using our cryptographic chain, the feasibility of optimization-based ambiguity attacks is effectively eliminated. In Figure 3, ChainMarks establishes sequential cryptographic relationships for both the trigger inputs and the target labels.

Trigger Inputs. The trigger inputs are generated as a cryptographically chained sequence by applying a cryptographic hash function over a seed key. In this sequence, each trigger inputs is ordered such that each input is derived from its predecessor by repeated

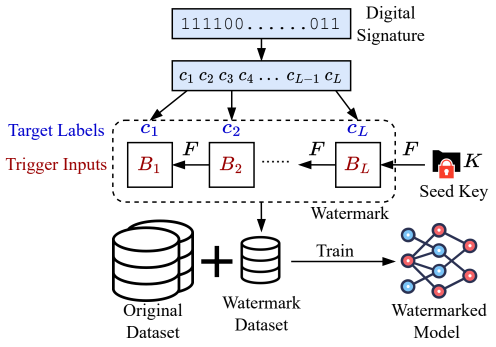


Figure 3: Embedding a watermark into a DNN model. The model owner generates a chain by selecting a seed key K and repeatedly applying a one-way function F for L iterations. The owner's digital signature is converted into target labels $\{c_i\}$, which are assigned to the triggers $\{B_i\}$ in sequential order.

application of the same hash function. Only the watermark owner, who possesses the secret key K, can generate the trigger inputs $\{B_i\}_{i=1}^{L}$, where $B_L = F(K)$ and $B_{i-1} = F(B_i)$, for $i \in \{2, ..., L\}$.

Target Labels. The target labels, which correspond to the ordered trigger inputs, are derived from the digital signature of the model owner. The digital signature, such as the hash value of the model owner's name, is converted into a number in base of the output dimension. For example, if the watermarked model has 8 classes, the digital signature is transformed into an octal number. Each digit of this number is then designated as the target label for a corresponding trigger input in sequential order. Thus, the generated sequence of target labels is denoted as $\{c_i\}$, where $i \in \{1, 2, ..., L\}$.

By applying our cryptographic mechanism, the trigger inputs and their corresponding target labels are cryptographically interrelated with the owner's seed key and digital signature, respectively. Therefore, an adversary is unable to apply optimization-based adversarial attacks, e.g., watermark ambiguity attacks.

The seed key K, serving as a secret key, is randomly selected by the DNN model owner and provided to the hash function to construct a one-way chain of trigger inputs. Note that the trigger inputs generated in the chain are of the same size as the DNN inputs, whereas the seed key size is not constrained. In Figure 3, F is an instance of the cryptographic one-way function; the sequence of watermark trigger inputs (or watermark keys) is generated by applying F, repeatedly. Once a chain of size $L + 1$ is constructed, the watermark keys can be disclosed and used in reverse order, meaning the last generated key will be the first to be used in the verification process. The number of disclosed keys is determined by the model owner based on specific applications. Due to the one-way property of F, the security of the remaining undisclosed keys is still intact, allowing them to be used in subsequent rounds of watermark verification. Thus, our method introduces cryptographic inter-constraints between the watermark trigger inputs, enabling multi-stage watermark verification using only a single seed key.

In the watermark embedding process, the model owner's digital signature is partitioned into target labels. Let the digital signature, denoted as S, be represented as a binary number with $|S|$ bits. Since the number of classes supported by a DNN may not be a power of 2, the binary digital signature (in base 2) must first be converted into a number in base C, where C represents the number of classes. For example, $C = 10$ for CIFAR-10 and $C = 100$ for CIFAR-100. Let S_C denote the digital signature represented in base C. The number of trigger input blocks is given by $L = \lceil \log_C S_C \rceil$. Thus, the digital signature S can be viewed as a sequence of L digits in base C, e.g., $S = (c_1 c_2 c_3 ... c_L)_C$, where $0 \le c_i < C$ for $1 \le i \le L$. These c_i values are then used as the target labels for the watermark. Compared to directly dividing the binary signature S into segments of length $\lceil \log_2 C \rceil$, our method allows for more trigger input blocks because $\lceil \log_C S_C \rceil \ge \lceil \log_2 S_2 / \lceil \log_2 C \rceil \rceil$.

The watermark-embedded DNN is built by training a model from scratch on both the original dataset and a watermark dataset. The watermark dataset consists of the trigger inputs and their corresponding target labels. In Section 7.1, we demonstrate that with ChainMarks, the watermark embedding accuracy can achieve 100%, without compromising the validation accuracy. There are two possible implementation approaches to obtain the watermarked model. The first approach is to train the model on both the original data and the watermark data. Another approach is to fine-tune a pre-trained DNN model with the watermark dataset. We adopt the first method based on a reasonable assumption that the model owner can access both the original dataset and the watermark dataset.

4.3 Watermark Verification

Figure 4 illustrates the overview of watermark verification. In the verification procedure, the model owner presents the seed key K and uses it to re-generate the chain of L trigger inputs, i.e., from B_1 to B_L. These trigger input blocks are then fed into the DNN model to obtain the corresponding output labels, denoted as $c'_1, c'_2, ..., c'_L$. These labels $\{c'_i\}$ can be concatenated to derive the digital signature.

$$S' := c'_1 \mid\mid c'_2 \mid\mid ... \mid\mid c'_{L-1} \mid\mid c'_L, \tag{1}$$

where $c'_i \in \{0, 1, ..., C-1\}$ represents a number in base-C numeral system. In addition, c'_i is a digit of S' in base C.

Then, the Hamming distance between S' and S, denoted as $d(S', S)$, is calculated as a measure of similarity between them. Hamming distance is a metric used to compare two data sequences of equal length, indicating the number of positions at which the corresponding symbols differ. This metric is typically employed in error detection and error correction, particularly in coding theory and data communications over computer networks.

Another parameter in the proposed watermarking scheme is the decision threshold θ for the Hamming distance. The verifier determines that the claimed watermark exists in the tested model only if $d(S', S) \le \theta \cdot L$. For example, if $\theta = 0.3$, the Hamming distance must be less than or equal to $0.3 \cdot L$ for S' to be considered a match for S, indicating at least 70% of the symbols must be identical. Section 4.4 explains how to determine the threshold θ based on a given probability threshold, which corresponds to the success rate of simple watermark guessing attacks. This probability threshold can serve as a representative metric for the security of a model.

4.4 Watermark Decision Threshold

Watermark presence detection with Hamming distance is closely related to estimating the probability distribution of Hamming distances when a simple watermark guessing attack is performed. In this scenario, an adversary can easily launch a simple guessing

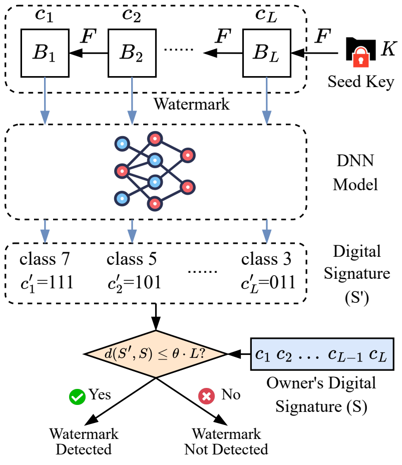


Figure 4: The overview of watermark verification with a cryptographic side-classifier. $d(S', S)$ stands for the Hamming distance between two binary strings, S' and S.

attack by providing a random seed key along with target labels derived from their digital signature, thereby attempting to claim the watermark's presence. By estimating the probability distribution of Hamming distances in such cases, we can derive the likelihood (or probability) of the watermark's presence given a particular seed key and target labels. This distribution is influenced by the DNN model's actual classification probabilities for random inputs. To address this, we introduce a new dynamic Monte Carlo method that enables the derivation of tight bounds on the distribution.

5 Two-Phase Monte Carlo Estimation

To derive the probability distribution of the number of matching labels under watermark guessing attacks, we first need to obtain the DNN classification probability distribution for random inputs. Then, the upper bounds of the matching probabilities can be derived by utilizing the classification probability distribution. However, due to the complexity of obtaining an analytical solution directly from a DNN model, we propose a dynamic Monte Carlo method that employs a two-phase sampling approach to approximate the matching probability distribution. In this section, we present a quantitative method for determining the detection threshold, addressing the question: given particular watermarks, how many retrieved watermarks are sufficient to assert ownership?

5.1 Classification Distribution Estimation

A significant challenge in estimating the classification probability distribution lies in the skewness across different classes. To address this, we conduct a simulation experiment and summarize the results in Table 1. Our findings reveals that the probability distribution exhibits a large standard deviation. Specifically, after feeding 10 million random inputs into a Resnet-18 model trained on CIFAR-10, we observed that 5 out of 10 classes have a hit probability of 0. Similarly, when 10 million random inputs are fed into the same model trained on CIFAR-100, 49 out of 100 classes are never hit. To derive an approximate classification probability distribution for a DNN model, we define the following terms in this section:

- N: the total number of random inputs created in simulation.

- C: the number of classes (or possible outputs) of the neural network. $C = 10$ for CIFAR-10 and C=100 for CIFAR-100.
- Γ: the set of all class indices, $\{0, 1, 2, ..., C - 1\}$.
- n_i: the number of inputs assigned to the class i, $0 \leq i \leq C-1$.
- $U = \{i_1, i_2, ..., i_k\}$: the set of k class indices within $[0, C - 1]$ that are not hit by any of the N inputs, with $|U| = k$.
- p_i: the classification probability that an input falls into class i, calculated as n_i/N.
- p_U: the 0-hit probability, defined as $p_{i_1} + p_{i_2} + ... + p_{i_k}$.

In the experiment shown in Table 1, we obtain a set of class indices U, comprising 49 out of 100 indices for the DNN model trained over CIFAR-100. However, the classification probabilities for the classes in U cannot be directly estimated from the experiment data as no input samples are classified into these classes.

To estimate the probability p_U, we devise a new approximation technique by modeling the experiment as a Bernoulli trial with a success probability of p_U and a failure probability of $1 - p_U$. To estimate the probability p_U of the set U, we can calculate the expected number of random inputs required to observe the first successful classification into any class within U. For example, in Table 1, we define U as the set of 49 class indices that receive no hits among the 10 million random inputs in the first stage. That means that the number of random inputs N is selected as 10 million to determine the initial sets U and $\Gamma - U$. In the second stage, we generate additional random inputs until a class in U is hit for the first time. We perform 50 simulations and observe that on average 11,531,629 additional inputs are required to achieve this first hit. Given that the number of trials before the first success in a Bernoulli process follows a geometric distribution, we can obtain $(1 - p_U)/p_U = 11,531,629$, which yields an estimated probability of $p_U = 1/(11,531,629 + 1) = 8.672 \times 10^{-8}$.

After deriving p_U, the previous probabilities (calculated under the assumption of $p_U = 0$), i.e., $p_i = n_i/N$ for all i in $\Gamma - U$, need to be adjusted by applying a normalization technique. Specifically, $p_i = p_i^{old} - p_U \cdot p_i^{old} = (1 - p_U) \cdot p_i^{old}$, for all i in $\Gamma - U$.

We also need to determine the approximate values of p_i for all i in U. Let i_j denote the j-th index in U, for $1 \leq j \leq k$. Then, equation $\sum_{j=1}^{k} p_{i_j} = p_U$ must hold for the definition of U.

5.2 Matching Probability Bound Estimation

Our objective is to assess the probability that a randomly generated one-way input chain is erroneously accepted as a valid watermark under a Hamming distance threshold θ. A valid watermark chain should have at least $\lceil L \cdot (1 - \theta) \rceil$ preserved blocks, where the labels produced by the DNN model match the digital signature of the legitimate owner of the model. The number of distinct combinations for selecting $\lceil L \cdot (1 - \theta) \rceil$ preserved blocks out of L blocks is

$$\binom{L}{\lceil L \cdot (1 - \theta) \rceil} = \binom{L}{\lfloor L \cdot \theta \rfloor}. \tag{2}$$

Given a DNN model and a sequence of L target labels corresponding to the model owner's digital signature, we aim to determine the success probability of a random guessing attack. Let $c_1, c_2, ..., c_L$ ($c_i \in [0, C - 1]$ for $1 \leq i \leq L$) denote the target classes (or labels) that match the owner's digital signature. A successful ownership claim is defined as achieving at least $m = \lceil L \cdot (1 - \theta) \rceil$ matching

Dataset	Avg. Prob.	Min Prob.	Max Prob.	Prob. Stdev	# of classes never hit
CIFAR-10	0.1	0	0.9962	0.2987	5
CIFAR-100	0.01	0	0.9433	0.0399	49

Table 1: Skewed probability distribution across different classes for DNN models trained on CIFAR-10/CIFAR-100.

labels between the attack inputs and the model owner's digital signature. The process can be modeled as counting the successful matches in a sequence of L independent yes/no experiments with the success probabilities of $p_{c_1}, p_{c_2}, ..., p_{c_L}$, where $c_i \in [0, C - 1]$. Poisson binomial distribution is well suited to calculate the above probabilities [27]. Let M be a random variable that indicates the number of matches. The probability of obtaining exactly m matches out of L chained random inputs can be expressed as

$$Pr(M = m) = \sum_{A \in F_m} \prod_{i \in A} p_{c_i} \prod_{j \in A^c} (1 - p_{c_j}), \tag{3}$$

where F_m is the set of all subsets of m integers that can be selected from $\{1, 2, ..., L\}$. For example, if $L = 3$ and $m = 2$, then $F_2 = \{\{1, 2\}, \{1, 3\}, \{2, 3\}\}$. For any subset $A \in F_m$, A^c is the complement of A. For example, if $A = \{1, 3\} \in F_2$, then $A^c = \{2\}$.

Obviously, F_m will contain $L!/((L-m)! \cdot m!)$ elements; hence, the Equation (3) is infeasible to compute in practice unless L is small. However, the following formula has been derived to approximate $Pr(M \geq m)$ using simple calculations.

Claim 1. *The probability of obtaining at least m matches out of L candidates is*

$$Pr(M \geq m) \approx \Phi(\frac{L + 0.5 - \mu}{\sigma'}) - \Phi(\frac{m - 0.5 - \mu}{\sigma'}), \tag{4}$$

where Φ is the cumulative distribution function (CDF) of the standard normal distribution, where the mean $\mu = L/C$ and the standard deviation σ' follows the equation:

$$\sigma' = \sqrt{\sum_{i=1}^{L} p_{c_i} - \sum_{i \in (\Gamma - U), 1 \leq i \leq L} p_{c_i}^2 - (p_U^2/k)}. \tag{5}$$

Proof. An approximation technique [27] can be employed to obtain the estimated probability. Approximation methods are still widely used due to their computational efficiency, especially when L is large. We will utilize the normal approximation method, which is based on the central limit theorem (CLT). If we define

$$\mu = \sum_{i=1}^{L} p_{c_i}, \ \sigma = \sqrt{\sum_{i=1}^{L} p_{c_i}(1 - p_{c_i})}. \tag{6}$$

Then, we can approximate the probability mass function using the normal approximation method with small errors for reasonably large values of $L (L \geq 10)$ [18, 78].

$$Pr(M = m) \approx \varphi(\frac{m + 0.5 - \mu}{\sigma}), \tag{7}$$

where M denotes the number of matching labels (i.e., class indices) to the digital signature, and φ stands for the *probability distribution function (PDF)* of the standard normal distribution.

For example, with the threshold $\theta = 0.3$, $L = 100$ inputs, and $C = 100$ classes, the attack success probability, i.e., the probability of a randomly generated chain of length L yielding $m = \lceil L \cdot (1-\theta) \rceil = 70$ or more matches, would be

$$\begin{aligned} P_r(M \geq 70) &= 1 - Pr(M < 70) \\ &= 1 - \sum_{i=0}^{69} \varphi(\frac{i+0.5-\mu}{\sigma}) \\ &= \sum_{i=70}^{L} \varphi(\frac{i+0.5-\mu}{\sigma}). \end{aligned} \tag{8}$$

Instead of using the *PDF* of $\varphi(x)$, $P_r(M \geq m)$ may be obtained from $\Phi(x)$, which is the *cumulative distribution function (CDF)* of the standard normal distribution.

$$P_r(M \geq m) = \Phi(\frac{L+0.5-\mu}{\sigma}) - \Phi(\frac{m-0.5-\mu}{\sigma}). \tag{9}$$

In our problem setting, we aim to approximate the probabilities p_j for j in U to obtain the values of μ and σ, and then apply the above approximation formula. Our goal is to find the upper bound on the attack success probability, whose example is shown in Equation (8). The following formula holds for reasonable values of θ:

$$\mu \approx \frac{L}{C} \ll L \cdot \theta \tag{10}$$

From the last formula in Equation (8), we observe that an upper bound for the attack success probability P_r' can be derived by finding a tight upper-bound σ' for σ. This is feasible because $\varphi(x)$ is a decreasing function for $x > 0$.

The value of σ can be rewritten as follows, noting that $|U| = k$.

$$\begin{aligned} \sigma &= \sqrt{\sum_{i=1}^{L}(p_{c_i} - p_{c_i}^2)} = \sqrt{\sum_{i=1}^{L} p_{c_i} - \sum_{i=1}^{L} p_{c_i}^2} \\ &= \sqrt{\sum_{i=1}^{L} p_{c_i} - \sum_{i\in(\Gamma-U),1\leq i\leq L} p_{c_i}^2 - \sum_{i\in U,1\leq i\leq L} p_{c_i}^2} \\ &\leq \sqrt{\sum_{i=1}^{L} p_{c_i} - \sum_{i\in(\Gamma-U),1\leq i\leq L} p_{c_i}^2 - \sum_{i\in U,1\leq i\leq L} (\frac{p_U}{k})^2} \\ &= \sqrt{\sum_{i=1}^{L} p_{c_i} - \sum_{i\in(\Gamma-U),1\leq i\leq L} p_{c_i}^2 - \frac{p_U^2}{k}} = \sigma'. \end{aligned} \tag{11}$$

Hence, the upper bound, σ', of σ is given in Equation (11). The inequality in Equation (11) holds due to the following optimization.

$$\begin{aligned} & min \sum_{i\in U,1\leq i\leq L} p_{c_i}^2, \\ & s.t., \sum_{i\in U,1\leq i\leq L} p_{c_i} = p_U, \\ & 0 \leq p_i \leq 1, 1 \leq i \leq L. \end{aligned} \tag{12}$$

In Equation (12), the item can achieve the minimum value *only if* every $p_{c_i} = p_U/k$, where $i \in U$, $1 \leq i \leq L$. □

For instance, the margin of error (MOE) of the approximation formula in Claim 1 is calculated to be less than 1.2% when compared to the precise values derived from Equation (3), for relatively small values (i.e., $L \leq 20$) using ResNet-18 models trained on CIFAR-10 and CIFAR-100 datasets.

m	0-6	7-8	9	10	11
$Pr(M \geq m)$	1.0	0.9999	0.9984	0.8382	0.1618
m	12	13	14	15-100	
$Pr(M \geq m)$	0.0015	4.01e-7	2.45e-12	0.0	

Table 2: The success probabilities over different match numbers for watermark guessing attacks against the ResNet-18 model trained on CIFAR-10 ($C = 10, L = 100$).

5.3 Threshold Decision

The probability distribution in Table 2 shows the number of matches obtained by a basic watermark guessing attack against a ResNet-18 model trained on the CIFAR-10 dataset. For the data in Table 2, the probability distribution of regular data would not change much even after watermark embedding, due to the small proportion of watermark data (as Table 5 shows the accuracy only drops slightly).

With the probability distribution, we set a threshold of the success probability for a simple watermark guessing attack. Then, the success probability threshold can be mapped to the min match number, whereas we derive the decision threshold θ. For example, if the success probability threshold is set to 10^{-7} (i.e., the compromise probability should be less than 10^{-7}), the number of matches m should be at least 14 based on the probability distribution in Table 2. Then, we can derive the decision threshold of Hamming distance as $\theta = 1 - (m/L) = 0.86$. Thus, when the chain length is set to $L = 100$, the max tolerance error rate for a match achieves 86%, i.e., a 14% match in trigger inputs is sufficient for ownership claim. In practice, the match ratio is typically higher, e.g., a 90% match can undoubtedly establish model ownership. Therefore, the decision threshold for determining a watermark presence is dependent on both the watermarked model, M_W, and the success probability threshold, p. This threshold can be expressed as a function of $\theta(M_W, p)$.

Our proposed two-phase estimation method is more precise especially when the output probability distributions are skewed (i.e., small p-value), where the traditional one-phase estimation methods cannot work. It is because, for traditional estimation methods, the CDF function cannot accumulate quickly to reach the target probability sum $(1-p)$ due to the requirement of a large number of empirical estimations. Therefore, the state-of-the-art estimation method [32] only uses a moderate p-value with low confidence. Our method is applicable to use smaller p-values, which correspond to higher marginal utility and higher level of security (Section-7.3).

6 Experiments

All watermarking schemes and removal attacks are implemented in PyTorch, on a server equipped with an NVIDIA GTX 1080 GPU.

6.1 Datasets and DNN Models.

Two image classification datasets, CIFAR-10 and CIFAR-100 [33], are used to build watermarked DNN models. In our experiments, we use two model types, i.e., ResNet-18 [26] and ResNet 28x10 [88], which are trained on CIFAR-10 and CIFAR-100 datasets. Model extraction attacks generally require extensive data access; thus, we assume adversaries have access to the full training dataset and know the architecture of the source model.

Scheme	Category	Verification	Capacity
ChainMarks	model dependent/independent	black-box	multi-bit
Adi	model dependent/independent	black-box	multi-bit
Content	model independent	black-box	zero-bit
Noise	model independent	black-box	zero-bit
Unrelated	model independent	black-box	zero-bit

Table 3: Black-box watermarking schemes in evaluation.

6.2 Other DNN Watermarking Schemes

We compare our scheme against four black-box watermarking methods [1, 92], summarized in Table 3. In Adi et al.'s approach [1], watermark trigger inputs are abstract images paired with randomly assigned labels from the full class space. They explore two embedding strategies: training a model from scratch on a combined dataset (original + watermark data) and fine-tuning a pre-trained model. Their results show that training from scratch offers greater robustness to model modification attacks. Accordingly, we adopt this setting and reproduce their models using the combined dataset.

Three other watermarking methods are proposed based on different types of trigger images: *Content*, *Noise*, and *Unrelated* images [92]. In the *content*-based approach, trigger inputs are randomly chosen from a single class and modified with a fixed secret mask, such as a white square over a specific image region. The *noise*-based method uses a mask generated from Gaussian noise, while the *unrelated*-image approach selects trigger inputs from a domain unrelated to that of the original DNN. The procedures for generating target labels, embedding watermarks, and verifying ownership are similar to those of the *Adi* scheme.

6.3 Watermark Removal Attacks

To compare the security and robustness of ChainMarks with four other schemes, we evaluate them against three categories of watermark removal attacks [53]: *input preprocessing*, *model modification*, and *model extraction*, as summarized in Table 4.

1) Watermark Removal via Input Preprocessing. These attacks remove watermarks without retraining the model by modifying the input images. For example, we can perform adaptive denoising [7] (or add Gaussian noise [89]) to the entire image. The JPEG compression attack [20] reduces image quality using JPEG encoding, which can eliminate watermarks. In the input quantization attack [48], pixel values are mapped to $2b$ evenly spaced intervals and replaced with the mean value of their interval. The input smoothing attack [84] applies a mean, median, or Gaussian filter, resulting in a blurred image that may suppress watermark features.

2) Watermark Removal via Model Modification. These attacks alter the model itself to remove embedded watermarks. Adversarial training [57] improves model robustness by injecting adversarial examples, which are generated via Projected Gradient Descent [57], into the training set and fine-tuning the model on them using ground-truth labels. The fine-tuning attacks refer to a set of model stealing attacks that apply a transformation to the model by fine-tuning [77]. Fine-tuning attacks [77] modify the model by fine-tuning to induce parameter changes. Four variants are considered: *Fine-Tune All Layers (FTAL)* and *Fine-Tune Last Layer (FTLL)* (with other layers frozen), both using ground-truth labels; and *Retrain All Layers (RTAL)* and *Retrain Last Layer (RTLL)*, which reinitialize either all layers or just the last layer and fine-tune using the model's

Attack	Category	Param. Access	Data Access
Adaptive Denoising	Input Preprocessing	White-box	None
JPEG Compression			
Input Quantization			
Input Smoothing			
Adversarial Training	Model Modification		Domain
Fine-Tuning (RTLL, RTAL)			
Weight Quantization			
Weight Pruning			
Regularization			
Fine-Tuning (FTLL, FTAL)			Labeled Subset
Transfer Learning	Model Extraction	Black-box	Domain
Retraining			
Cross-Architecture Retraining			
Adversarial Training (From Scratch)			

Table 4: Watermark removal attacks in our evaluation.

predicted labels. Weight pruning [95] removes a random subset of weights from the model until a target sparsity level ρ is reached. Weight quantization [28], unlike input quantization, reduces the precision of model weights instead of input images. Regularization attack [68] involves two phases: first, applying strong regularization to shift the model to a new set of parameters (potentially far from the original), which lowers test accuracy; second, recovering accuracy through fine-tuning.

3) Watermark Removal via Model Extraction. Model extraction can remove watermarks by training a surrogate model to replicate the source model's behavior while discarding embedded watermarks [55]. In the retraining attack [75], a surrogate model is trained from scratch using input-label pairs obtained via API access. The cross-architecture retraining variant uses a different model architecture to reduce watermark transfer. The transfer learning attack [74] initializes the surrogate from a pre-trained model in another domain. In adversarial training (from scratch) [57], the surrogate is trained from scratch using adversarial examples, similar to standard adversarial training.

7 Performance Analysis

7.1 Efficiency of Watermark Embedding

We conduct 20 independent experiments to evaluate the test accuracy (i.e., accuracy on the original test dataset) of source models before and after watermark embedding, as well as the watermark accuracy (i.e., accuracy on watermark dataset) before and after watermark removal attacks.

Table 5 presents the average test/watermark accuracies under watermark ambiguity and 16 watermark removal attacks. After embedding the watermark using ChainMarks, the watermark accuracy reaches 100%, while the test accuracy experiences only a minor drop−0.8% for models trained on CIFAR-10 (92.3%→91.5%) and CIFAR-100 (69.1%→68.3%). This indicates that the impact of watermark embedding on model utility is negligible, typically under 1%. Table 5 further demonstrates that ChainMarks achieves higher overall robustness than other methods while maintaining embedding efficiency comparable to *Adi*. However, ChainMarks is more secure and robust than *Adi*. First, *Adi* relies on multiple independent backdoor samples and is therefore susceptible to watermark ambiguity attacks when attackers generate adversarial alternatives for each backdoor trigger. In contrast, ChainMarks can resist such ambiguity attacks via cryptographic chain (see Table 6). Second, for

Accuracy	Accuracies (CIFAR-10/CIFAR-100)				
	ChainMarks	Adi	Content	Noise	Unrelated
Test Accuracy w/o WM embedding	0.923/0.691	0.921/0.692	0.915/0.684	0.913/0.685	0.914/0.682
Test Accuracy w/ WM embedding	0.915/0.683	0.916/0.685	0.91/0.681	0.911/0.678	0.909/0.676
Test Accuracy after Attack	0.78/0.68	0.77/0.69	0.56/0.52	0.81/0.73	0.53/0.51
WM Accuracy after Embedding	1.0/1.0	1.0/1.0	1.0/1.0	1.0/1.0	1.0/1.0
WM Accuracy after Attack	0.67/0.34	0.69/0.37	0.58/0.33	0.73/0.41	0.64/0.35

Table 5: Test and watermark (WM) accuracy before/after watermark embedding and after watermark attacks.

the verification with small p-value, *Adi* does not provide effective support, whereas ChainMarks remains robust (see Section 7.2).

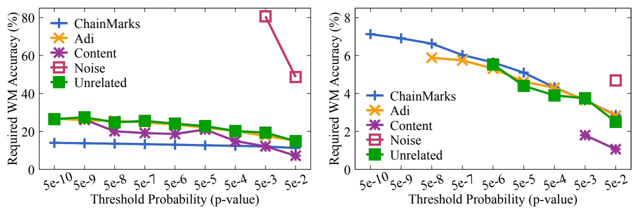
After watermark removal/ambiguity attacks, the watermark accuracy decreases from 100% to 67% (34%) on CIFAR-10 (CIFAR-100); however, the number of remaining valid watermarks is sufficient for ownership verification, based on the quantitative analysis in Section 4.4. More analysis of watermark robustness under various attacks is provided in Section 8.2. In addition, a notable decline in test accuracy is observed when attackers perform watermark removal/ambiguity attacks. This indicates that adversaries cannot effectively reduce watermark accuracy in the surrogate model without significantly compromising its utility on the original task. Besides, traditional watermarking methods improve embedding performance by injecting more watermark samples. However, ChainMarks employs a hash function to ensure each watermark sample is cryptographically independent. Therefore, adding more trigger samples (increasing chain length) does not lead to greater model memorization of the watermark.

7.2 Effects of p-values

To investigate the relationships between the required watermark accuracy $(1 - \theta)$ and the threshold probability p, we apply a two-phase Monte Carlo estimation method to analyze the ChainMarks scheme. For comparison, we adopt the empirical estimation approach proposed in [53] to evaluate four existing watermarking schemes. For a range of threshold probabilities (i.e., p-values), we conduct 20 independent experiments to determine the average watermark accuracy necessary to successfully verify ownership. The results are presented in Figure 5 for both CIFAR-10 and CIFAR-100.

As shown in Figure 5, one notable observation is that, for small p-values, certain schemes encounter erroneous conditions where the cumulative distribution function (CDF) fails to reach the target probability mass. This limitation arises from the insufficient number of models used in the empirical estimation of decision thresholds [53], suggesting that the empirical method may not be reliable for the settings of small p-values. Such issues are observed in the *Noise*- and *Content*-based schemes on CIFAR-10, and in all schemes except ChainMarks on CIFAR-100. Also, the *Noise*-based scheme consistently exhibits high watermark accuracy requirements and may become impractical when the probability threshold p is set below 0.01.

In Figure 5(a), for models trained on CIFAR-10, ChainMarks exhibits the lowest required watermark accuracy among all evaluated



(a) models on CIFAR-10. **(b) models on CIFAR-100.**

Figure 5: Required watermark accuracy $(1 - \theta)$ vs. threshold probability p, for different watermarking schemes.

schemes. This is attributed to ChainMarks's high level of security, which enables a greater tolerance for errors, i.e., a higher allowable Hamming distance threshold θ. In Figure 5(b), for models trained on CIFAR-100, the required watermark accuracy of ChainMarks is comparable to that of the *Adi* and *Unrelated* image-based schemes, while decision thresholds cannot be computed for the *Noise*- and *Content*-based schemes in most cases. Besides, as the p-value increases, the required watermark accuracy decreases, since a higher p-value indicates a higher probability of compromise and hence requires fewer matched watermarks to verify ownership. Thus, ChainMarks is able to meet higher security requirements.

7.3 Watermark Marginal Utility

Comparing the required watermark accuracies across different schemes is inherently challenging due to the different decision thresholds. Therefore, relying only on the required watermark accuracy (or retention rate) is insufficient to evaluate the robustness of a watermarking scheme or the effectiveness of an attack.

An effective metric is needed to be derived to quantify the contribution of retained watermarks (i.e., watermark accuracy in a surrogate model) in terms of probabilistic guarantees on the success probability of random guessing attacks. Therefore, we define this metric as *watermark marginal utility*, which represents the average reduction factor in the attack success probability per unit increase in watermark accuracy (or decision threshold) within the surrogate model. In other words, the metric indicates the degree of reduction that can be achieved in the threshold probability p by increasing the watermark accuracy (or decreasing the decision threshold θ) in a surrogate model.

The watermark marginal utility is illustrated in Figure 5 by dividing the ratio of p-values by the difference in watermark accuracy for two consecutive p-values on the x-axis. If the p-value is reduced from p_1 to p_2 and the corresponding required watermark accuracy increases from τ_1 to τ_2, the watermark marginal utility can be calculated as $(p_1/p_2)/(\tau_2 - \tau_1)$. For example, if the p-value reduces from 0.05 to 0.005 and the corresponding watermark accuracy increases from 0.1123 to 0.1204, the estimated watermark marginal utility is computed as (0.05/0.005)/(0.1204 - 0.1123) = 1234.56.

Figure 6 presents the computed watermark marginal utilities for CIFAR10 and CIFAR100 models across different watermarking schemes. The results show that ChainMarks provides a higher watermark marginal utility compared to other schemes. The marginal utility values for CIFAR-100 models are not provided for the *Adi*, *Content*, *Noise*, and *Unrelated*-based schemes due to the limitations in computing small p-values with empirical estimation method [53].

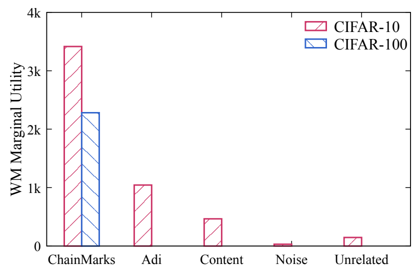


Figure 6: Watermark marginal utility for various schemes.

7.4 Overhead

Training the model on CIFAR-10 for 200 epochs takes 2 hours, with a RAM usage of 2.5 GB and a GPU memory usage of 2.4 GB. Training the model on CIFAR-100 for 200 epochs takes 7 hours, with a RAM usage of 3.7 GB and a GPU memory usage of 6.6 GB. When watermark images are applied in the training set, the computational overhead remains negligible and does not significantly affect training time or memory usage.

8 Security Analysis

8.1 Defeating Watermark Ambiguity Attacks

The two cryptographic constraints introduced in ChainMarks for trigger inputs and target labels render any optimization-based attacks, such as watermark ambiguity attacks [21, 25, 36, 39], infeasible. In ambiguity attacks, attackers find adversarial watermarks by optimizing with the "perturbed input - expected output" pairs. When trigger inputs are independent (see Figure 1), such objectives are easily optimized since the added items are independent. However, with cryptographic chaining, each trigger depends on a one-way hash function, which lacks gradients and thus obstructs backpropagation. As illustrated in Figure 7, even when attackers inject trainable noise into fake triggers and optimize noise to match the output with digital signature, the optimized inputs break the required cryptographic chain, invalidating the watermark structure.

An adversary may also launch a guessing attack with trial and error. It can first choose a random seed key to create a one-way trigger input chain. Then, the adversary simply applies the original DNN model in a feedforward manner to check if the output labels match the claimed digital signature. Attackers can repeatedly attempt with different seed keys until a match is obtained. However, according to our analysis in Section 4.4, the success rate of random guessing is extremely low and can be determined by the selected threshold. Therefore, this approach will require an exponential number of trials, rendering it computationally infeasible.

8.2 Countering Watermark Removal Attacks

Because trigger inputs are derived from hash values, they can be regarded as random noise. Compared to the data distribution of primary task (e.g., image object recognition), these noise-like triggers are out-of-distribution with respect to both training and fine-tuning datasets. Thus, these triggers are robust against removal attacks, since fine-tuning typically alters model behavior within the task-specific feature space, leaving the trigger space unaffected.

Evaluating the success of an attack requires careful consideration of both test accuracy loss and watermark accuracy degradation

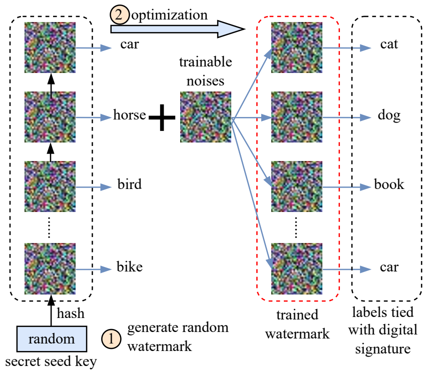


Figure 7: Attackers can use optimization to generate adversarial trigger inputs with associated invalid signature labels, but the resulting inputs are no longer cryptographic chained.

in surrogate models. To assess watermark robustness, we deploy five watermarking schemes, including ChainMarks, to construct watermarked models trained on the CIFAR-10/CIFAR-100 datasets. As listed in Table 6, we then apply 17 distinct attacks to the watermarked (source) models. Let M_s denote the surrogate model obtained by an attacker, and let p represent the threshold probability used to derive the decision threshold θ. For an attack to be considered successful, it must meet both of the following criteria.

Test Accuracy Criteria. To satisfy the accuracy drop threshold of 0.1, the test accuracy of the surrogate model should be at least 90% of that of the watermarked model.

Watermark Accuracy Criteria. The watermark accuracy in M_s should be less than the original decision threshold $\theta(M_W, p)$, where the threshold probability $p = 0.01$.

We obtain the decision threshold $\theta(M_W, p)$ for each watermarking scheme. For ChainMarks, the threshold is derived using a two-phase Monte Carlo estimation method, as described in Section 4.4. For the other four watermarking schemes, no precise method exists for computing θ; therefore, we adopt the empirical estimation technique proposed in [53]. This approach estimates the watermark accuracy of an unmarked model with two random variables. Specifically, we estimate the cumulative probability that a randomly generated watermark key (image and label) yields a watermark accuracy exceeding a specified threshold on an unmarked model.

Under the independence assumption, we generate 100 random watermarking keys and evaluate their label-matching accuracy on a set of 30 unmarked models. The distribution of matching counts is approximated using a cumulative normal distribution, and the decision threshold is selected to correspond to a p-value of 0.05. However, this technique provides only a rough estimate due to two limitations: each model has a unique classification probability distribution, even with the same architecture, owing to variations in training data and hyper-parameters; and (ii) the distribution of matching probabilities for random watermark keys is not explicitly modeled. Section 7.2 has demonstrated the limitations in calculating the cumulative probability functions for small p-values.

After evaluating each watermarking scheme against 17 distinct attack types, we present their robustness in Table 6. The results show that ChainMarks is resistant to the watermark ambiguity attack, whereas all four baseline schemes are vulnerable. Also, against the remaining 16 attacks, ChainMarks, *Adi*, and *Noise*-based scheme

Attack Types	Robust (-) or Vulnerable (V) for CIFAR-10 / CIFAR-100				
	ChainMarks	Adi	Content	Noise	Unrelated
WM Ambiguity Attack	-/-	V/V	V/V	V/V	V/V
Adaptive Denoising	-/-	-/-	-/-	-/-	-/-
JPEG Compression	-/-	-/-	-/-	-/-	-/-
Input Quantization	-/-	-/-	-/-	-/-	-/-
Input Smoothing	-/-	-/-	-/-	-/-	-/-
Adversarial Training	-/-	-/-	-/-	-/-	-/-
Fine-Tuning (RTAL)	-/-	-/-	-/-	-/-	-/-
Fine-Tuning (RTLL)	-/-	-/-	-/-	-/-	-/-
Fine-Tuning (FTAL)	-/-	-/-	V/V	-/-	V/V
Fine-Tuning (FTLL)	-/-	-/-	-/-	-/-	-/-
Weight Quantization	-/-	-/-	-/-	-/-	-/-
Weight Pruning	-/-	-/-	-/-	-/-	-/-
Regularization	-/-	V/-	V/-	-/-	V/-
Retraining	-/-	-/-	V/V	V/-	V/V
Transfer Learning	V/V	V/V	V/V	V/V	V/V
Cross-Architecture Retraining	-/-	-/-	V/-	-/-	V/-
Adversarial Training	-/-	-/-	-/-	-/-	-/-

Table 6: Robustness of different watermarking schemes against 17 attack types (threshold probability $p = 0.01$).

exhibit relatively higher robustness. However, compared to ChainMarks, *Adi* is vulnerable to regularization attacks, and *Noise*-based scheme is vulnerable to retraining attacks. With the exception of transfer learning where the p-value is 0.012/0.035 on CIFAR-10/CIFAR-100, ChainMarks consistently achieves p-values between 6×10^{-3} and 1×10^{-8} across all other watermark removal attacks. Thus, ChainMarks is the most robust scheme that is able to resist multiple watermark removal attacks.

9 Discussion

9.1 Usability

The ChainMarks scheme can be adopted by model owners (whether commercial vendors or individual developers) to protect their intellectual property. Based on dynamic watermarking, ChainMarks does not interfere with the model's primary functionality, since the triggers are out-of-distribution inputs that resemble random noise and do not affect regular inference. We tried various hash functions (e.g., MD5, SHA1, SHA128) and observed the selection does not affect the final results. Block cipher in counter mode could be an alternative; however, hashing is faster and the hard-bound is not an issue. Moreover, ChainMarks introduces negligible training overhead since the number of triggers is not comparable to the size of original training dataset. Besides, cryptographic chains can be extended to watermark RNNs and LLMs, but in different formats.

ChainMarks provides a higher security guarantee due to its higher marginal utility. Specifically, for the ResNet-18 model trained on CIFAR-10, 14 matches out of 100 triggers are sufficient to support a successful ownership claim. For models trained on CIFAR-100, even fewer matches are required, as the threshold is dependent on the output dimension. A larger output space corresponds to a lower random guessing probability, thereby allowing a more relaxed matching requirement. Besides, the watermarks can be embedded through fine-tuning, using initial weights and learning rates that differ from those used in "training-from-scratch".

In practical watermark verification, it is not necessary to disclose the entire key chain. For example, for a model trained on CIFAR-10, ownership can be verified by presenting only the first 20 trigger inputs (B_1 to B_{20}). If at least 14 of 20 triggers match, the ownership claim is considered valid. The remaining triggers can be reserved for further verification rounds. Also, due to the one-way property, any unused triggers in key chain remain secure and undisclosed.

9.2 Scalability

To extend ChainMarks to larger and more complex datasets (e.g., ImageNet), several adjustments are required to the watermark configuration. Although the increased class number reduces the likelihood of a single successful guess, the probability of accidentally achieving the minimum match threshold may not decrease proportionally and can even increase. Thus, the chain length L should be increased accordingly, but kept sufficiently short to preserve the nature of out-of-distribution. In addition, the digital signature should be encoded in a higher-base numeral system to match the class number. Due to the higher dimension of ImageNet data, hash-like triggers are more likely to be memorized, as they reside in sparser regions of the data manifold, far from natural image distributions.

Hyperparameters should be selected carefully to balance watermark security, robustness, and efficiency. We recommend setting $L \geq \sqrt{C}$, where C is the output class number, to ensure sufficient watermark entropy. The p-value, which determines the Hamming distance threshold, should range between 10^{-2} and 10^{-6}, depending on the desired security level. While the specific hash function has limited impact on cryptographic strength, we recommend strong cryptographic hashes (e.g., SHA-256) for high-security applications.

9.3 Limitations and Future Work

The main contribution of ChainMarks is to defeat watermark ambiguity attacks, which are emerging threats against all existing DNN model watermarking methods. ChainMarks cannot effectively defeat removal attacks via transfer learning and knowledge distillation. In fact, none of the existing watermarking techniques is robust against these two methods. Besides, ChainMarks focuses on watermarking the classification models with the inputs of images. However, our idea of watermarking with a key chain can be extended to other input formats or other modeling tasks. For text data, it is feasible to convert a hash value into a word (i.e., word ID) [41] or a pseudorandom string [80]. For graph-based input, we can transform a binary hash value into an adjacency matrix to generate graph-structured data. We leave this topic to future work.

10 Related Work

10.1 Backdoor Poisoning Attacks

Backdoor poisoning attack is a special case of targeted poisoning attacks that maintain overall performance and induce misbehaviors in triggers [67]. Data manipulation is the main technique for backdoor poisoning attacks [73]. Adversaries introduce either visible [24, 38] or invisible [44, 52, 62, 63] patterns into poisoning samples. Also, triggers for poisoned samples can be generated by optimization to achieve better performance [42, 51, 94]. Semantic backdoor attacks leverage the semantic part of the samples as trigger patterns, so it is unnecessary to modify the input at inference time [3, 4]. Similarly, a hidden backdoor can be activated by combining certain objects in images [49]. It is possible to conceal triggers using image scaling attacks [83]. However, almost all backdoor attacks are sample-agnostic and therefore can be defeated by trigger-synthesis-based defenses [79] and saliency-based defenses [19]. Hence, sample-specific backdoor attacks are proposed to contain different trigger patterns for different poisoned samples [44, 60]. Backdoor attacks can also be launched in the physical world using a pair of glasses [12] or a post-it note [24]. In addition, backdoor attacks can be applied in different fields, e.g., computer vision [31, 61, 87], natural language processing [13], speech recognition [90], software code [86], and graph learning [82, 93].

10.2 DNN Watermarking Schemes

White-box/Black-box/Box-free Watermarking. Based on the information accessible during watermark verification, watermarking schemes can be classified as white-box, black-box, and box-free [43]. A white-box watermarking scheme grants users or adversaries access to the internal information of DNNs, e.g., model structures, model weights, and hyperparameters [34, 54, 77]. However, due to the strong assumption, white-box watermarking has a larger capacity but limited applicability [11]. Black-box watermarking schemes only allow users to access the final outputs of DNN models by feeding a set of inputs [1, 37], allowing IP protection for Machine Learning as a Service (MLaaS) [32]. Box-free watermarking is similar to the black-box one; however, this mechanism is only applied to DNNs with high-dimensional outputs, i.e., image processing models, since watermarks can be embedded into the outputs for any inputs by a high output entropy [81, 91].

Static vs. Dynamic Watermarking. Watermarking schemes can be classified as static or dynamic based on watermarking methods [6]. Static methods embed watermarks in the static DNN parameters that are not changed during the operation. For example, watermarks can be embedded as the probability distribution of weights [10] or model weights [72, 77]. However, most static methods imply white-box watermarking, since the model parameters need to be accessible during verification. Dynamic methods associate watermarks with the network behaviors in correspondence to some specific inputs [1, 25, 37, 66, 70, 91, 92]. A set of secret inputs/patterns with target labels (i.e., triggers) are carefully crafted as the watermarks to be ingrained into the DNN in the training process along with the original data set. In watermark verification, model behaviors will be tested to verify the presence of the watermark. Dynamic watermarking usually generates trigger input by leveraging DNN backdoor poisoning attacks [47]. However, dynamic watermarking does not imply black-box watermarking, as it can also be used as white-box watermarking. For example, Rouhani et al. use activation maps to embed and verify watermarks [66].

Zero-bit vs. Multi-bit Watermarking. Based on the type of watermark contents disclosed in the verification, watermarking schemes can be classified as zero-bit and multi-bit [40]. In zero-bit watermarking, only the watermark presence is detected [1, 37, 92]; while in multi-bit watermarking, both the watermark and its presence should be present in the verification process [11]. A multi-bit watermarking scheme can be converted to a zero-bit one.

10.3 Attacks/Defenses on DNN Watermarks

DNN watermark attacks include model modification attacks, evasion attacks, and active attacks [85].

Model Modification Attacks. Model weights are often modified by the pirate. Model modification attacks include model fine-tuning [14, 15, 77], model pruning or parameter pruning [66], model weight compression [77], and model retraining [8, 59].

Evasion Attacks. Evasion attacks are more complicated. Shafieinejad et al. investigate the removal of backdoor-based watermarks with white-box, black-box, and inference attacks [68]. Also, DNN laundering is shown to reset backdoor watermarks [2]. Attackers can manipulate a model to remove the owner's signature if watermark presence is known in advance [25]. Reverse engineering can be used if the original training dataset is obtained [21]. Liu et al. propose a data augmentation scheme to mimic the backdoor trigger behaviors [50]. Gong et al. dynamically adjust the learning rate to purify backdoors [23]. Attackers can leverage resource-efficient attacks [29, 69, 76] to find black-box adversarial examples. By combining hybrid attacks with seed prioritization, adversarial examples can be obtained using only a few queries. The two main optimization techniques used in attacks are AutoZOOM [76] and NES [29].

Active Attacks. Ambiguity attack tends to forge an additional watermark on the DNN model to doubt the ownership verification [16, 21]. Also, several methods are proposed to detect watermarks for further attacks [17, 56, 84]. Attackers can overwrite the watermarks if they know the watermarking method [10, 11, 66]. In addition, it is possible to prevent the copyright owner from verifying the ownership by using a watermark collusion attack [10]. Similarly, attackers can also detect and modify the watermark query to prevent watermark verification [59].

Countermeasures. Entangled watermarks increase the similarity between watermarks and task features, improving resistance to trigger detection [30]; however, ChainMarks prevents attackers from adding new ambiguous watermarks. DynaMarks defeats model extraction attacks by dynamically changing the responses of the model's prediction API during the inference phase [9]. Bansal et al. propose randomized smoothing to improve the difficulty of watermark removal attacks [5]. To defeat watermark ambiguity attack, Fan et al. propose a passport layer so that model performance deteriorates due to forged signatures [21].

11 Conclusion

We propose a new DNN watermarking scheme, ChainMarks, which is resistant to watermark ambiguity attacks by introducing cryptographic constraints among watermark triggers and target labels, along with the model owner's digital signature. Experiments show that ChainMarks exhibits higher or comparable levels of resistance compared to other watermark schemes against various watermark attacks, including input processing, model modification, and model extraction attacks. To determine watermark decision threshold, the proposed two-phase Monte Carlo method shows its accuracy and applicability across a range of watermarked DNN models. The marginal utility of ChainMarks is higher than that of the other schemes, providing a higher probability guarantee of the watermark presence in the DNN models with the same level of watermark accuracy.

Acknowledgments

This work is partially supported by Office of Naval Research grant N00014-23-1-2122 and Commonwealth Cyber Initiative award N-3Q25-002.

References

[1] Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. 2018. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *27th USENIX Security Symposium (USENIX Security 18)*. 1615–1631.

[2] William Aiken, Hyoungshick Kim, Simon Woo, and Jungwoo Ryoo. 2021. Neural network laundering: Removing black-box backdoor watermarks from deep neural networks. *Computers & Security* 106 (2021), 102277.

[3] Eugene Bagdasaryan and Vitaly Shmatikov. 2021. Blind backdoors in deep learning models. In *Usenix Security*.

[4] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. 2020. How to backdoor federated learning. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 2938–2948.

[5] Arpit Bansal, Ping-yeh Chiang, Michael J Curry, Rajiv Jain, Curtis Wigington, Varun Manjunatha, John P Dickerson, and Tom Goldstein. 2022. Certified Neural Network Watermarks with Randomized Smoothing. In *International Conference on Machine Learning*. PMLR, 1450–1465.

[6] Mauro Barni, Fernando Pérez-González, and Benedetta Tondi. 2021. DNN watermarking: Four challenges and a funeral. In *Proceedings of the 2021 ACM Workshop on Information Hiding and Multimedia Security*. 189–196.

[7] Antoni Buades, Bartomeu Coll, and J-M Morel. 2005. A non-local algorithm for image denoising. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, Vol. 2. Ieee, 60–65.

[8] Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. 2021. IPGuard: Protecting intellectual property of deep neural networks via fingerprinting the classification boundary. In *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*. 14–25.

[9] Abhishek Chakraborty, Daniel Xing, Yuntao Liu, and Ankur Srivastava. 2022. DynaMarks: Defending Against Deep Learning Model Extraction Using Dynamic Watermarking. *arXiv preprint arXiv:2207.13321* (2022).

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

[11] Huili Chen, Bita Darvish Rouhani, and Farinaz Koushanfar. 2019. Blackmarks: Blackbox multibit watermarking for deep neural networks. *arXiv preprint arXiv:1904.00344* (2019).

[12] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. 2017. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526* (2017).

[13] Xiaoyi Chen, Ahmed Salem, Dingfan Chen, Michael Backes, Shiqing Ma, Qingni Shen, Zhonghai Wu, and Yang Zhang. 2021. Badnl: Backdoor attacks against nlp models with semantic-preserving improvements. In *Annual Computer Security Applications Conference*. 554–569.

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

[15] Xinyun Chen, Wenxiao Wang, Yiming Ding, Chris Bender, Ruoxi Jia, Bo Li, and Dawn Song. 2019. Leveraging unlabeled data for watermark removal of deep neural networks. In *ICML workshop on Security and Privacy of Machine Learning*. 1–6.

[16] Yiming Chen, Jinyu Tian, Xiangyu Chen, and Jiantao Zhou. 2023. Effective ambiguity attack against passport-based dnn intellectual property protection schemes through fully connected layer substitution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8123–8132.

[17] Siyuan Cheng, Guanhong Tao, Yingqi Liu, Shengwei An, Xiangzhe Xu, Shiwei Feng, Guangyu Shen, Kaiyuan Zhang, Qiuling Xu, Shiqing Ma, et al. 2023. BEAGLE: Forensics of Deep Learning Backdoor Attack for Better Defense. *arXiv preprint arXiv:2301.06241* (2023).

[18] KP Choi and Aihua Xia. 2002. Approximating the number of successes in independent trials: Binomial versus Poisson. *The Annals of Applied Probability* 12, 4 (2002), 1139–1148.

[19] Edward Chou, Florian Tramer, and Giancarlo Pellegrino. 2020. Sentinet: Detecting localized universal attacks against deep learning systems. In *2020 IEEE Security and Privacy Workshops (SPW)*. IEEE, 48–54.

[20] Gintare Karolina Dziugaite, Zoubin Ghahramani, and Daniel M Roy. 2016. A study of the effect of jpg compression on adversarial images. *arXiv preprint arXiv:1608.00853* (2016).

[21] Lixin Fan, Kam Woh Ng, and Chee Seng Chan. 2019. Rethinking deep neural network ownership verification: Embedding passports to defeat ambiguity attacks. *Advances in neural information processing systems* 32 (2019).

[22] Henri Gilbert and Helena Handschuh. 2003. Security analysis of SHA-256 and sisters. In *International workshop on selected areas in cryptography*. Springer, 175–193.

[23] Xueluan Gong, Yanjiao Chen, Wang Yang, Qian Wang, Yuzhe Gu, Huayang Huang, and Chao Shen. 2023. REDEEM MYSELF: Purifying Backdoors in Deep Learning Models using Self Attention Distillation. In *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, 755–772.

[24] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. 2017. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733* (2017).

[25] Jia Guo and Miodrag Potkonjak. 2018. Watermarking deep neural networks for embedded systems. In *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 1–8.

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[27] Yili Hong. 2013. On computing the distribution function for the Poisson binomial distribution. *Computational Statistics & Data Analysis* 59 (2013), 41–51.

[28] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2017. Quantized neural networks: Training neural networks with low precision weights and activations. *The Journal of Machine Learning Research* 18, 1 (2017), 6869–6898.

[29] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. 2018. Black-box adversarial attacks with limited queries and information. In *International conference on machine learning*. PMLR, 2137–2146.

[30] Hengrui Jia, Christopher A Choquette-Choo, Varun Chandrasekaran, and Nicolas Papernot. 2021. Entangled Watermarks as a Defense against Model Extraction.. In *USENIX Security Symposium*. 1937–1954.

[31] Jinyuan Jia, Yupei Liu, and Neil Zhenqiang Gong. 2022. Badencoder: Backdoor attacks to pre-trained encoders in self-supervised learning. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2043–2059.

[32] Katarzyna Kapusta, Vincent Thouvenot, Olivier Bettan, Hugo Beguinet, and Hugo Senet. 2021. A protocol for secure verification of watermarks embedded into machine learning models. In *Proceedings of the 2021 ACM Workshop on Information Hiding and Multimedia Security*. 171–176.

[33] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. [n. d.]. CIFAR-10 and CIFAR-100 datasets. https://www.cs.toronto.edu/~kriz/cifar.html.

[34] Minoru Kuribayashi, Takuro Tanaka, Shunta Suzuki, Tatsuya Yasui, and Nobuo Funabiki. 2021. White-box watermarking scheme for fully-connected layers in fine-tuning model. In *Proceedings of the 2021 ACM Workshop on Information Hiding and Multimedia Security*. 165–170.

[35] Leslie Lamport. 1981. Password authentication with insecure communication. *Commun. ACM* 24, 11 (1981), 770–772.

[36] Yingjie Lao, Weijie Zhao, Peng Yang, and Ping Li. 2022. DeepAuth: A DNN authentication framework by model-unique and fragile signature embedding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 9595–9603.

[37] Erwan Le Merrer, Patrick Perez, and Gilles Trédan. 2020. Adversarial frontier stitching for remote neural network watermarking. *Neural Computing and Applications* 32 (2020), 9233–9244.

[38] Chaoran Li, Xiao Chen, Derui Wang, Sheng Wen, Muhammad Ejaz Ahmed, Seyit Camtepe, and Yang Xiang. 2021. Backdoor attack on machine learning based android malware detectors. *IEEE Transactions on Dependable and Secure Computing* 19, 5 (2021), 3357–3370.

[39] Huiying Li, Emily Wenger, Shawn Shan, Ben Y Zhao, and Haitao Zheng. 2019. Piracy resistant watermarks for deep neural networks. *arXiv preprint arXiv:1910.01226* (2019).

[40] Li Li, Weiming Zhang, and Mauro Barni. 2023. Universal BlackMarks: Key-Image-Free Blackbox Multi-Bit Watermarking of Deep Neural Networks. *IEEE Signal Processing Letters* 30 (2023), 36–40.

[41] Peixuan Li, Pengzhou Cheng, Fangqi Li, Wei Du, Haodong Zhao, and Gongshen Liu. 2023. PLMmark: a secure and robust black-box watermarking framework for pre-trained language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 14991–14999.

[42] Shaofeng Li, Minhui Xue, Benjamin Zi Hao Zhao, Haojin Zhu, and Xinpeng Zhang. 2020. Invisible backdoor attacks on deep neural networks via steganography and regularization. *IEEE Transactions on Dependable and Secure Computing* 18, 5 (2020), 2088–2105.

[43] Yiming Li, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. 2022. Backdoor learning: A survey. *IEEE Transactions on Neural Networks and Learning Systems* (2022).

[44] Yuezun Li, Yiming Li, Baoyuan Wu, Longkang Li, Ran He, and Siwei Lyu. 2021. Invisible backdoor attack with sample-specific triggers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 16463–16472.

[45] Yue Li, Benedetta Tondi, and Mauro Barni. 2021. Spread-transform dither modulation watermarking of deep neural network. *Journal of Information Security and Applications* 63 (2021), 103004.

[46] Yue Li, Hongxia Wang, and Mauro Barni. 2021. A survey of deep neural network watermarking techniques. *Neurocomputing* 461 (2021), 171–193.

[47] Yiming Li, Mingyan Zhu, Xue Yang, Yong Jiang, Tao Wei, and Shu-Tao Xia. 2023. Black-box Dataset Ownership Verification via Backdoor Watermarking. *IEEE Transactions on Information Forensics and Security* (2023).

[48] Ji Lin, Chuang Gan, and Song Han. 2019. Defensive quantization: When efficiency meets robustness. *arXiv preprint arXiv:1904.08444* (2019).

[49] Junyu Lin, Lei Xu, Yingqi Liu, and Xiangyu Zhang. 2020. Composite backdoor attack for deep neural network by mixing existing benign features. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. 113–131.

[50] Xuankai Liu, Fengting Li, Bihan Wen, and Qi Li. 2021. Removing backdoor-based watermarks in neural networks with limited data. In *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 10149–10156.

[51] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. 2018. Trojaning attack on neural networks. In *25th Annual Network And Distributed System Security Symposium (NDSS 2018)*. Internet Soc.

[52] Zeyan Liu, Fengjun Li, Zhu Li, and Bo Luo. 2022. LoneNeuron: a Highly-Effective Feature-Domain Neural Trojan Using Invisible and Polymorphic Watermarks. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. 2129–2143.

[53] Nils Lukas, Edward Jiang, Xinda Li, and Florian Kerschbaum. 2022. Sok: How robust is image classification deep neural network watermarking?. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 787–804.

[54] Peizhuo Lv, Pan Li, Shengzhi Zhang, Kai Chen, Ruigang Liang, Hualong Ma, Yue Zhao, and Yingjiu Li. 2023. A Robustness-Assured White-Box Watermark in Neural Networks. *IEEE Transactions on Dependable and Secure Computing* (2023).

[55] Peizhuo Lv, Hualong Ma, Kai Chen, Jiachen Zhou, Shengzhi Zhang, Ruigang Liang, Shenchen Zhu, Pan Li, and Yingjun Zhang. 2024. MEA-defender: a robust watermark against model extraction attack. In *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2515–2533.

[56] Wanlun Ma, Derui Wang, Ruoxi Sun, Minhui Xue, Sheng Wen, and Yang Xiang. 2022. The" Beatrix"Resurrections: Robust Backdoor Detection via Gram Matrices. *arXiv preprint arXiv:2209.11715* (2022).

[57] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2017. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083* (2017).

[58] Risto Miikkulainen, Jason Liang, Elliot Meyerson, Aditya Rawal, Daniel Fink, Olivier Francon, Bala Raju, Hormoz Shahrzad, Arshak Navruzyan, Nigel Duffy, et al. 2019. Evolving deep neural networks. In *Artificial intelligence in the age of neural networks and brain computing*. Elsevier, 293–312.

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

[60] Tuan Anh Nguyen and Anh Tran. 2020. Input-aware dynamic backdoor attack. *Advances in Neural Information Processing Systems* 33 (2020), 3454–3464.

[61] Maximilian Noppel, Lukas Peter, and Christian Wressnegger. 2022. Disguising Attacks with Explanation-Aware Backdoors. In *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, 996–1013.

[62] Xudong Pan, Mi Zhang, Beina Sheng, Jiaming Zhu, and Min Yang. 2022. Hidden trigger backdoor attack on {NLP} models via linguistic style manipulation. In *31st USENIX Security Symposium (USENIX Security 22)*. 3611–3628.

[63] Erwin Quiring and Konrad Rieck. 2020. Backdooring and poisoning neural networks with image-scaling attacks. In *2020 IEEE Security and Privacy Workshops (SPW)*. IEEE, 41–47.

[64] Vincent Rijmen and Elisabeth Oswald. 2005. Update on SHA-1. In *Topics in Cryptology–CT-RSA 2005: The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005. Proceedings*. Springer, 58–71.

[65] Ronald Rivest. 1992. *The MD5 message-digest algorithm*. Technical Report.

[66] Bita Darvish Rouhani, Huili Chen, and Farinaz Koushanfar. 2019. Deepsigns: an end-to-end watermarking framework for protecting the ownership of deep neural networks. In *ACM International Conference on Architectural Support for Programming Languages and Operating Systems*.

[67] Giorgio Severi, Jim Meyer, Scott E Coull, and Alina Oprea. 2021. Explanation-Guided Backdoor Poisoning Attacks Against Malware Classifiers.. In *USENIX Security Symposium*. 1487–1504.

[68] M Shafieinejad et al. 1906. On the robustness of the backdoor-based watermarking in deep neural networks. CoRR (2019).

[69] Fnu Suya, Jianfeng Chi, David Evans, and Yuan Tian. 2020. Hybrid batch attacks: Finding black-box adversarial examples with limited queries. In *29th USENIX Security Symposium*.

[70] Sebastian Szyller, Buse Gul Atli, Samuel Marchal, and N Asokan. 2021. Dawn: Dynamic adversarial watermarking of neural networks. In *Proceedings of the 29th ACM International Conference on Multimedia*. 4417–4425.

[71] Hailun Tan, Sanjay Jha, Diet Ostry, John Zic, and Vijay Sivaraman. 2008. Secure multi-hop network programming with multiple one-way key chains. In *Proceedings of the first ACM conference on Wireless network security*. 183–193.

[72] Enzo Tartaglione, Marco Grangetto, Davide Cavagnino, and Marco Botta. 2021. Delving in the loss landscape to embed robust watermarks into neural networks. In *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 1243–1250.

[73] Zhiyi Tian, Lei Cui, Jie Liang, and Shui Yu. 2022. A Comprehensive Survey on Poisoning Attacks and Countermeasures in Machine Learning. *Comput. Surveys* 55, 8 (2022), 1–35.

[74] Lisa Torrey and Jude Shavlik. 2010. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*. IGI global, 242–264.

[75] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. 2016. Stealing Machine Learning Models via Prediction APIs.. In *USENIX security symposium*, Vol. 16. 601–618.

[76] Chun-Chen Tu, Paishun Ting, Pin-Yu Chen, Sijia Liu, Huan Zhang, Jinfeng Yi, Cho-Jui Hsieh, and Shin-Ming Cheng. 2019. Autozoom: Autoencoder-based zeroth order optimization method for attacking black-box neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 742–749.

[77] Yusuke Uchida, Yuki Nagai, Shigeyuki Sakazawa, and Shin'ichi Satoh. 2017. Embedding watermarks into deep neural networks. In *Proceedings of the 2017 ACM on international conference on multimedia retrieval*. 269–277.

[78] A Yu Volkova. 1996. A refinement of the central limit theorem for sums of independent random indicators. *Theory of Probability & Its Applications* 40, 4 (1996), 791–794.

[79] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. 2019. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 707–723.

[80] Shu Wang, Kun Sun, and Yan Zhai. 2024. Dye4AI: Assuring Data Boundary on Generative AI Services. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*. 2281–2295.

[81] Hanzhou Wu, Gen Liu, Yuwei Yao, and Xinpeng Zhang. 2020. Watermarking neural networks with watermarked images. *IEEE Transactions on Circuits and Systems for Video Technology* 31, 7 (2020), 2591–2601.

[82] Zhaohan Xi, Ren Pang, Shouling Ji, and Ting Wang. 2021. Graph Backdoor.. In *USENIX Security Symposium*. 1523–1540.

[83] Qixue Xiao, Yufei Chen, Chao Shen, Yu Chen, and Kang Li. 2019. Seeing is Not Believing: Camouflage Attacks on Image Scaling Algorithms.. In *USENIX Security Symposium*. 443–460.

[84] Weilin Xu, David Evans, and Yanjun Qi. 2017. Feature squeezing: Detecting adversarial examples in deep neural networks. *arXiv preprint arXiv:1704.01155* (2017).

[85] Mingfu Xue, Jian Wang, and Weiqiang Liu. 2021. DNN intellectual property protection: Taxonomy, attacks and evaluations. In *Proceedings of the 2021 on Great Lakes Symposium on VLSI*. 455–460.

[86] Limin Yang, Zhi Chen, Jacopo Cortellazzi, Feargus Pendlebury, Kevin Tu, Fabio Pierazzi, Lorenzo Cavallaro, and Gang Wang. 2023. Jigsaw Puzzle: Selective Backdoor Attack to Subvert Malware Classifiers. *IEEE Symposium on Security and Privacy (S&P)* (2023).

[87] Yuanshun Yao, Huiying Li, Haitao Zheng, and Ben Y Zhao. 2019. Latent backdoor attacks on deep neural networks. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 2041–2055.

[88] Sergey Zagoruyko and Nikos Komodakis. 2016. Wide residual networks. *arXiv preprint arXiv:1605.07146* (2016).

[89] Valentina Zantedeschi, Maria-Irina Nicolae, and Ambrish Rawat. 2017. Efficient defenses against adversarial attacks. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. 39–49.

[90] Tongqing Zhai, Yiming Li, Ziqi Zhang, Baoyuan Wu, Yong Jiang, and Shu-Tao Xia. 2021. Backdoor attack against speaker verification. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2560–2564.

[91] Jie Zhang, Dongdong Chen, Jing Liao, Han Fang, Weiming Zhang, Wenbo Zhou, Hao Cui, and Nenghai Yu. 2020. Model watermarking for image processing networks. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34. 12805–12812.

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

[93] Zaixi Zhang, Jinyuan Jia, Binghui Wang, and Neil Zhenqiang Gong. 2021. Backdoor attacks to graph neural networks. In *Proceedings of the 26th ACM Symposium on Access Control Models and Technologies*. 15–26.

[94] Shihao Zhao, Xingjun Ma, Xiang Zheng, James Bailey, Jingjing Chen, and Yu-Gang Jiang. 2020. Clean-label backdoor attacks on video recognition models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 14443–14452.

[95] Michael Zhu and Suyog Gupta. 2017. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878* (2017).