Merkle-Sign: Watermarking Framework for Deep Neural Networks in Federated Learning

Fang-Qi Li
solour_lfq@sjtu.edu.cn
Shanghai Jiao Tong University, School of Electronic Information and Electrical Engineering
Shanghai, China

Shi-Lin Wang
wsl@sjtu.edu.cn
Shanghai Jiao Tong University, School of Electronic Information and Electrical Engineering
Shanghai, China

Alan Wee-Chung Liew
a.liew@griffith.edu.au
Griffith University, School of Information and Communication Technology
Gold Coast Campus, Australia

ABSTRACT

With the wide application of deep learning models, it is important to verify an author’s possession over a deep neural network model, e.g. by embedding watermarks and protect the model. The development of distributed learning paradigms such as federated learning (FL) leverages new challenges for model protection. Apart from the independent verification of all authors’ ownership, the author collaboration should be able to recover its participant’s identity against adaptive attacks and trace traitors. To meet those requirements, in this paper, we demonstrate a watermarking framework, Merkle-Sign, for protecting deep neural networks in FL. By incorporating state-of-the-art watermarking schemes and the cryptological primitive designed for distributed storage, the framework meets the prerequisites for ownership verification in FL. This work paves the way for generalizing watermark as a practical security mechanism for protecting deep learning models in distributed learning platforms.

CCS CONCEPTS

• Security and privacy → Digital rights management; Domain-specific security and privacy architectures.

KEYWORDS

deep learning model protection, federated learning, machine learning security

1 INTRODUCTION

Deep neural networks (DNN) are intelligent systems that provide services by learning from data and consuming enormous computational resources. DNN models have been successfully applied in internet of things [6, 19], intrusion detection [21, 29], etc. With more emerging applications, their reliability and security are gaining more attention. A crucial task in artificial intelligence security is to protect DNN models as intellectual properties by proving authors’ ownership. Mechanisms designed for the integrity of traditional communication are inappropriate for DNN for two reasons: (1) Unlike traditional communication where information is passed from one agent to another, DNN models are usually uploaded onto public websites for credit, visibility, and peer review. So schemes designed for identity verification in bidirectional channels are unsuitable for DNN models. (2) It is easy to modify a published model (e.g., changing the last bits of one parameter) without significantly damaging its commercial utility. This fact challenges the direct application of cryptological primitives under which changing a single bit within the protected object results in a completely different signature.

Having observed that DNN models share similar properties with multi-media objects (broadcast transmission and semantic invariance under slight modifications), researchers resorted to watermark, which protects the integrity for multi-media objects. The metrics for evaluating a DNN watermarking scheme are similar to those for the multi-media setting. Different kinds of redundancy within DNN models, such as parameter representation, outputs of intermediate layers, and backdoor triggers can be exploited to encode the author’s identity as watermarks. There have been various watermarking schemes designed to satisfy these security requirements. Modern paradigms of model training and distributing such as Federated Learning (FL) [16] raise new challenges for DNN watermarking. In FL, authors sharing different local datasets collaborate to train a model. As a result, the final product contains the contribution of all the authors. Such an organization results in a different threat model and ownership verification requirements from those in the traditional setting where one party completes the training of the entire DNN model.

In FL, a watermarking scheme should encode all authors’ identity information into the model while: (1) Each agent participating in the training can prove its ownership without informing its co-authors or the aggregator server. (2) One agent participating in the training cannot falsify itself as its co-author. (3) If one agent undergoes several adaptive attacks so its watermark is erased then its co-authors can recover its identity proof. (4) If a traitor within
the author union pirates the model then its identity can be correctly tracked. It is hard to fulfill all these requirements without a sophisticated verification protocol. Meanwhile, such protocol would raise additional demands for the underlying watermarking scheme for identity encoding. These concerns increase the difficulty in model protection for FL.

Being confronted with all these challenges, we propose a unified DNN watermarking framework that meets the practical security requirements for FL. The proposed framework adopts a watermarking scheme and a public verification protocol to ensure unforgeable and robust ownership proof. We prove that such a scheme meets the requirements for FL listed above. Apart from the top-level protocol, our design puts forward more requirements for the underlying watermarking schemes. To the best of our knowledge, this is the first proposal for provable ownership protection in the FL scenario. The contributions of this paper are:

- We formulate security requirements for FL, a practical scenario for secure machine learning.
- A framework for ownership verification, Merkle-Sign, is proposed to meet all the requirements for FL. We also propose an underlying watermarking scheme, ATGF, to increase the applicability of Merkle-Sign in real-world settings.
- Experiments demonstrate the utility of the proposed framework and examine the capability of established watermarking schemes in FL.

2 BACKGROUNDS AND PRELIMINARIES

2.1 DNN watermark

Let \( M_{\text{clean}} \) be a model trained to fulfill a primary task \( T_{\text{primary}} \). The author embeds its identity information, key \( \epsilon \mathcal{K} \), into the model, where \( \mathcal{K} \) is the space of legal keys. Such embedding can be revealed later by a module verify, a probabilistic algorithm with binary output. A watermarking scheme \( \mathbb{W} = \{ \text{Gen}, \text{Embed} \} \) consists of one module for key generation and one for key embedding:

\[
\text{key} \leftarrow \text{Gen}(1^N), \quad (M_{\mathbb{WM}}, \text{verify}) \leftarrow \text{Embed}(M_{\text{clean}}, \text{key}),
\]

where \( N \) is the security parameter. When the author finds that its model has been stolen, it provides \{key, verify\} as its evidence. If verify agrees with key concerning the suspicious model \( M \) then the author’s ownership is proven. Such proof can be conducted and announced by a trusted third party. This centralized setting introduces extra security risks and a tremendous burden to the center. Instead, as [10, 18, 20], the proof as a consensus service can be done in a public and decentralized manner so any party can participate in it. In particular, we assume that there exists a community of public verification agents. Each individual within the community examines the watermark independently and provides its result. The ownership verification outcome is voted across the entire community to defend against potential conspiracy and perjury. This community is motivated by the necessity of protecting DNN intellectual property. An agent correctly participating in verification is going to be assigned corresponding credits with which it can initialize a verification process for its product.

Current watermarking schemes mainly focus on embedding information into the DNN models. Zhang et al. proposed a backdoor-based watermarking scheme [31] for the black-box setting using triggers. Backdoor is an attack against DNN by triggering unexpected results. In backdoor-based watermarking, an author conducts the backdoor attack on its model and claims the exceptional output as its identity proof. Schemes ensuring the secure generation and transmission of backdoors provide better security [1, 32]. Moreover, variational autoencoder [17], adversarial samples [12] and out-of-range samples [15] in \textit{W\#onder Filter}, have been adopted as candidates for the backdoor.

Uchida et al. [22] proposed the pioneering white-box watermarking scheme. Their scheme embeds the author’s digital signature into the model’s weight. In [8], Guan et al. inserted a reversible watermark into the model’s parameter. Deep-Sign proposed by Davish et al. [4] builds the watermark on the intermediate output of the DNN model on certain samples instead of the weights. The scheme MTL-Sign proposed by Li et al. in [13] models watermark embedding as an extra learning task. Since regulating the intermediate output of the DNN model is more difficult than tuning weights, these hybrid schemes are more robust and secure than weight-based schemes. Zhang et al. [30] adopted deep watermarking to protect DNN models specialized for image processing.

Despite all the recent works, DNN watermarking is still far from practical application. Most of the proposals overlooked the top-level protocol for key generation, key distribution, and verification. A practical watermarking framework must include both the bottom-level watermarking scheme and the top-level protocol. The authorship verification of DNN models should be conducted publicly in a decentralized manner. However, the public verification process itself raises additional risks as we will elaborate on later.

A DNN watermarking scheme \( \mathbb{W} \) has to satisfy the following basic security requirements.

2.1.1 Correctness. The author can prove its ownership correctly:

\[
\Pr(\text{verify}(M_{\mathbb{WM}}, \text{key}) = 1) \geq 1 - \epsilon,
\]

where \( \epsilon \) is a negligible function of the security parameter \( N \). For an adversary with the incorrect evidence, verify is not going to mistake it as the authentic author:

\[
\Pr_{\text{key} \leftarrow \mathcal{K}} \{\text{verify}(M_{\mathbb{WM}}, \text{key}') = 1\} \leq \epsilon,
\]

in which \( \text{key}' \) is a uniformly sampled instance from \( \mathcal{K} \).

2.1.2 Functionality-preserving. Let \( \mathcal{E} \) denote a task-dependent metric that evaluates a model’s performance, e.g., the accuracy on the test set for classification. It is desirable that watermarking does not substantially decrease the model’s functionality, so:

\[
\mathcal{E}(M_{\mathbb{WM}}) \geq \mathcal{E}(M_{\text{clean}}) - \epsilon
\]

holds almost always, where \( \epsilon \) measures the author’s tolerance on the watermark’s side-effect.

2.1.3 Security against tuning. The adversary can fine-tune a model on a smaller dataset similar to that of \( T_{\text{primary}} \). Such tuning could ruin some weight-based watermarks like the reversible system in [8]. A watermarking scheme is secure against tuning if tuning
cannot invalidate the ownership verification, which can be formulated as:

\[
\Pr \{\text{verify}(M_{\text{tuned}}, \text{key}) = 1\} \geq 1 - \epsilon,
\]

where \(M_{\text{tuned}}\) is obtained by having an adversary tune \(M_{\text{WM}}\).

2.1.4 **Covertness.** The watermark should be hidden from the adversary. A necessary condition for the watermark’s covertness is: the parameters of \(M_{\text{WM}}\) should not deviate from those of \(M_{\text{clean}}\) too much. This deviation can be measured by the statistics of the two models’ parameters as in the property inference [23].

2.1.5 **Privacy-preserving.** In scenarios where anonymity is necessary such as model competitions, the watermarked model must not reveal the author’s identity. A watermarking scheme is privacy-preserving [13] if no efficient adversary, given \(\text{key}_1\) and \(\text{key}_2\), can distinguish \(M_{\text{WM},1}\) from \(M_{\text{WM},2}\) where:

\[
\begin{align*}
(M_{\text{WM},1}, \text{verify}_1) &\leftarrow \text{Embed}(M_{\text{clean}}, \text{key}_1) , \\
(M_{\text{WM},2}, \text{verify}_2) &\leftarrow \text{Embed}(M_{\text{clean}}, \text{key}_2).
\end{align*}
\]

2.1.6 **Security against overwriting.** An adversary can embed its watermark into the model and **redeclare** the model with its watermark as its product to the verification community. Such overwriting must not invalidate the original watermark. The redeclaration can be solved by either using an authorized time server or a decentralized consensus protocol with which an author authorizes its

time-stamp to the verification community before publishing the DNN model [13].

2.1.7 **Security against piracy.** For watermarking schemes with structural flaws or whose key space is too small (as in the cases of most backdoor-based schemes), an adversary can trivially claim the ownership by forging the evidence. A necessary condition for the security against ownership piracy is to incorporate cryptological primitives as in [32].

2.1.8 **Security against the spoil attack.** An author might have to prove its possession over a DNN model for **multiple** times. For example, it discovers that its model has been stolen, tuned, and sold to multiple parties. If an adversary having obtained the evidence can spoil the watermark and invalidate the verification afterward then the watermarking scheme is insecure. We define this attack as the **spoil attack**. The procedure of a spoil attack is illustrated in Fig. 1. An eavesdropping adversary firstly participates in the proof for a author, during which process it obtains \(\text{key, verify}\) for model \(M\). It then tunes \(M\) into \(M_{\text{spoiled}}\) in a reverse learning manner so that the original verify cannot recognize \((M_{\text{spoiled}}, \text{key})\). Finally, the adversary sells \(M_{\text{spoiled}}\).

To the best of our knowledge, no existing watermarking scheme is secure against the spoil attack. For backdoor-based watermarking schemes, the adversary can fit the model to randomly shuffled
labels on the backdoor triggers. For weight-based watermarking schemes, the adversary can replace the watermarked layers. The concurrent watermarking schemes and their security level w.r.t. all requirements are listed as in Table 1.

2.2 Federated learning

To associate a broader range of data sources while preserving the privacy of each data provider, many distributed learning frameworks have been proposed. The representative example is Federated Learning (FL) [16], where a collection of \( K \) participants \( \mathcal{U} = \{ u_k \}_{k=1}^K \) and optionally an aggregator server \( \mathcal{A} \) cooperate to train a model. FL can be classified into horizontal FL and vertical FL. In horizontal FL, each party holds different samples with an identical set of features. This paradigm has been applied in medical and financial settings [28]. In the vertical FL, the feature spaces of different parties are different. As illustrated in Fig. 2, the centralized client-server setting of FL consists of many epochs, during the \( i \)-th epoch the aggregator distributes the current model \( M^{(i)} \) to all authors. Each author \( u_k \) independently computes the direction to which the model should evolve w.r.t. its local dataset \( D_k \) and transmits the gradient \( \Delta M^{(i)}_k \) back. Finally, the aggregator collects the feedback from all authors, updates the model, then starts another round. Possible ways of model aggregation are model average, gradient average as in (1), secure aggregation, homomorphic encryption-based weighted average, etc [3].

\[
M^{(i+1)} = M^{(i)} + \alpha \cdot \sum_{k=1}^{K} |D_k| \cdot \Delta M^{(i)}_k. \tag{1}
\]

During the entire procedure, each author communicates with the aggregator through an encrypted channel: it downloads the current model and uploads the information for model updating. No data is transmitted through any public channel.

![Figure 2: The client-server architecture for FL.](image)

Current studies on the security of FL focus on privacy protection against curious or malicious aggregators or authors. It has been claimed that the gradient information \( \Delta M^{(i)}_k \) might leak information of \( D_k \) under reverse engineering [2]. Therefore differential privacy [24, 26] methods have been widely adopted to protect the encrypted gradients. For example, by perturbing the encrypted gradients, neither the curious aggregator nor its conspirators can deduce the noiseless gradient. Yet the perturbation strategy keeps the summation of gradients invariant and preserves the stability in convergence.

As a separate aspect for FL security, DNN model protection is confronted with distinct threat models.

3 THE THREAT MODEL AND SECURITY REQUIREMENTS

3.1 Watermark capacity

It is necessary that all authors’ identities are embedded into the DNN model, inclusively and independently. Therefore practical watermarking scheme for DNN models in the FL setting has to embed multiple pieces of identity information into the model. It is generally assumed that watermark is embedded into a DNN by utilizing the model’s redundancy in parameterization. To provide correct and independent verification for each author, it is necessary to quantitively measure this redundancy. We define a DNN model’s watermark capacity w.r.t. \( \mathcal{W} \) and the decline of its performance \( \delta \) w.r.t. the metric \( \mathcal{E} \) defined in its primary task.

The value \( \text{cap}_{\mathcal{W}}^{\delta} \) measures the upper bound of the number of successfully embedded watermarks inside a model. Formally, \( \text{cap}_{\mathcal{W}}^{\delta} \) is the maximal \( q \) satisfying the following conditions:

\[
(M_1, \text{verify}_1) \leftarrow \text{Embed}(M_{\text{clean}}, \text{key}_1),
(M_2, \text{verify}_2) \leftarrow \text{Embed}(M_1, \text{key}_2),
\ldots
(M_q, \text{verify}_q) \leftarrow \text{Embed}(M_{q-1}, \text{key}_q),
\mathcal{E}(M_q) \geq \mathcal{E}(M_{\text{clean}}) - \delta,
\]

where \( \text{key}_q \) is generated by \( \text{Gen} \) and all \( q \) watermarks can be correctly verified. A very deep DNN model naturally has a large capacity since the watermarking embedding process usually modifies only a small number of parameters. Although it is difficult to formally calculate \( \text{cap}_{\mathcal{W}}^{\delta} \) for arbitrary \( \mathcal{W} \), it can be empirically estimated from (2).

3.2 Requirements in FL ownership verification

Apart from the ordinary threats against the DNN watermark, the collaboration of authors in FL gives rise to new threats and requirements. For example, the authors must not breach each other’s privacy from the watermark. Meanwhile, we expect that authors can help each other to recover the identity proof against the spoil attack. In cases a malicious author [7] sells the intermediate model as its product, it should be correctly identified. The security requirements for FL are detailed as the following aspects:

3.2.1 Independency. When FL terminates, each author \( u_k \) holds an evidence pair \((\text{key}_k, \text{verify}_k)\), in which \( \text{key}_k \) is determined by
3.2.2 Privacy-preserving. For author $u_k \in \mathcal{U}$, $u_k \neq u_k$, no information about $u_k$’s identity, especially $key_{uk}$, appears in $u_k$’s perspective when interacting with the aggregator/being chained up in the decentralized setting. Formally, $u_k$ cannot succeed in pretending to be $u_k$.

3.2.3 Recovery. Once $u_k$ proves its ownership over $M$, an eavesdropping adversary might conduct the spoil attack to invalidate this proof. Given the shared interest and cooperation in property protection from all collaborating authors, it is expected that $u_k$’s co-authors can recover $u_k$’s ownership over $M$. Hence a spoil attack against $u_k$’s watermark is insufficient to disprove $u_k$’s authorship completely.

3.2.4 Traitor-tracing. In DNN model commercialization, tracing of unauthorized reselling can be achieved by watermarking [27]. In FL, a traitor may participate in a few epochs of training, obtain the intermediate model, then claim the current model as its product to the public, or sell the model to another party. The aggregator supervising the FL training process, or the collaboration of honest authors in the decentralized setting, should be able to correctly identify the traitor.

The dependency of these four advanced requirements on the basic security properties is presented in Table 2.

### Table 2: Dependence of the advanced requirements on basic security requirements.

| Advanced requirements | Basic security requirements |
|-----------------------|----------------------------|
|                       | Correctness | Functionality-preserving | Security against tuning | Covertness | Privacy | Security against overwriting | Security against piracy | Multiple-time verification |
| Indepedency.          | ✓           | ✓                      | ×                      | ×          | ×       | ✓                                    | ✓                  | ✓                         |
| Privacy-preserving.   | ×           | ×                      | ×                      | ✓          | ×       | ×                                    | ✓                  | ×                         |
| Recovery.             | ✓           | ✓                      | ×                      | ×          | ✓       | ×                                    | ×                  | ×                         |
| Traitor-tracing.      | ✓           | ✓                      | ✓                      | ✓          | ×       | ×                                    | ✓                  | ✓                         |

✓ means relevant. × means irrelevant.

$u_k$. After the model $M$ absorbing the contributions of all authors is published, it is necessary that:

$$\Pr\{verify_{uk}(M, key_{uk}) = 1\} \geq 1 - \epsilon.$$ 

So $u_k$ can independently prove its contribution in $M$ without informing any other parties.

In addition, the verification protocol should establish a correlation between the identity information of all authors to accomplish the recovery property.

### 4 THE MERKLE-SIGN FRAMEWORK

#### 4.1 Motivation

To meet the requirements in FL model protection, we have to use a watermarking scheme satisfying the two requirements listed in Section 3.3. To reduce the cost of public verification, we propose a public verification protocol that takes the merit of the Merkle-tree structure. Merkle-tree is a data structure for data integrity verification in cloud storage [14]. Unlike a naive hash function that simply hashes a list of files into a string, the Merkle-tree hash allows partial authentication of a subset of the input, which is necessary for distributed storage and retrieval. This property forms the basis for independent verification and recovery.

#### 4.2 Merkle-Sign for FL

We propose Merkle-Sign, an efficient and secure watermarking framework for FL model protection, which operates as Algo. 1 and is visualized in Figure 3. In which Merkle is a Merkle-tree style combinator. The idea is that an aggregator $\mathcal{A}$ is responsible for embedding the identity information of all authors into the final DNN model. To ensure that all evidence is valid and the decrease of the model’s performance is upper bounded by $\delta$, it is necessary that:

$$\text{cap}^\delta_{\text{sign}} \geq (K + 1).$$

To trace the traitor before the training terminates, the intermediate model being sent to $u_k$ contains a hook, which is a surveillance key specialized for $u_k$. $key_{uk}$ This key is only known to $\mathcal{A}$, from which it can identify the traitor.

To reduce the communication traffic between the FL participants and the verification community while enabling the independence and recovery property, we adopt the Merkle-tree combinator. A hash function hash1 and a collision resistant hash function hash2 are involved in building the Merkle-tree. hash1 maps $K$, any legal verify module or info into $\{0, 1\}^l$. hash2 maps $\{0, 1\}^{2l}$ into $\{0, 1\}^l$. The Merkle operator maps each component of its input by hash1, then organizes the mapped values into a binary tree with hash2 as the reduction operator. Concretely, hash1 for key in the
Algorithm 1 Merkle-Sign framework for centralized FL.

Input: A one-time watermarking scheme \( \mathcal{WM} \), security parameters \( N \) and \( L \), hash functions \( \text{hash}_1 \) and \( \text{hash}_2 \).

Output: A watermarked model \( M_{\mathcal{A}} \) and evidence.

1. \( \mathcal{A} \) generates \( \text{key}_0 \leftarrow \text{Gen}(1^N) \).
2. Each \( u_k \in \mathcal{U} \) generates \( \text{key}_k \leftarrow \text{Gen}(1^N) \), submits it and \( \text{hash}_1(\text{key}_k) \) to \( \mathcal{A} \).
3. For each \( u_k \), \( \mathcal{A} \) generates \( \text{key}_k^\dagger \leftarrow \text{Gen}(1^N) \).
4. \( \mathcal{A} \) initializes a clean model.
5. While not terminate do
6. For each \( u_k \), \( \mathcal{A} \) embeds \( \text{KEYs}_k = (\text{key}_0, \text{key}_k^\dagger) \) into the current model as \( M_{\mathcal{WM},k} \), obtains \( \text{VERs}_k = (\text{verify}_0, \text{verify}_k^\dagger) \).
7. \( \mathcal{A} \) signs and broadcasts:
   \[
   (\text{time}||\text{Merkle} (\text{KEYs}_k, \text{VERs}_k, \text{info})).
   \]
8. \( \mathcal{A} \) transmits \( M_{\mathcal{WM},k} \) to \( u_k \).
9. Each \( u_k \) uploads the encrypted gradient to \( \mathcal{A} \).
10. \( \mathcal{A} \) averages the gradients and updates the clean model.
11. End while
12. \( \mathcal{A} \) embeds \( \text{KEYs} = \{\text{key}_k\}_{k=0}^K \) into the final model \( M_{\mathcal{A}} \), obtains \( \text{VERs} = \{\text{verify}_k\}_{k=0}^K \).
13. \( \mathcal{A} \) signs and broadcasts:
   \[
   (\text{time}||\text{Merkle} (\text{KEYs}, \text{VERs}, \text{info})).
   \]
14. \( \mathcal{A} \) sends the intermedia of \( \text{Merkle} (\text{KEYs}, \text{VERs}, \text{info}) \) and \( \text{verify}_k \) to \( u_k \).
15. \( \mathcal{A} \) publishes \( M_{\mathcal{A}} \).

The output of \( \text{Merkle} \) is the value of its root node \( T_{\text{root}} \in \{0, 1\}^7 \), the intermedia are the values of all the remaining nodes. An instance for \( \text{Merkle} \) on three keys is illustrated in Fig. 4. The underlying setting is \( K = 2 \) authors. To justify the ownership, an author, e.g., \( u_1 \) submits \( \text{key}_1, \text{verify}_1, \text{info} \) and the necessary information for deriving \( T_{\text{root}} \), i.e., \( T_{00}, T_{01}, T_{10}, T_{11}, T_{000}, T_{010}, T_{101}, \) and \( T_{111} \) to the verification community. Any member of the community, given the public key of \( u_k \) and the access to \( \text{hash}_0, \text{hash}_2 \) can independently examine whether the submitted evidence is consistent with the suspicious model and the root node.
This scheme is more space-friendly since the length of the root of the Merkle-tree is a constant $r$. Moreover, the traffic burden for verification is replaced by the identity proof cost, as only an extra amount of information that is the same order as the height of the Merkle-tree, i.e., $O(\ln K)$, needs to be transmitted. If the aggregator simply hashes the keys of all authors as one string then independent verification is impossible. While hashing and broadcasting each key multiplies the communication cost by a factor $O(K)$.

The basic security of this framework can be reduced to the security of the underlying watermarking scheme. The functionality-preserving property, and the capacity of the model brace the performance of the watermarked model. The security against tuning and piracy, and the covertness for each key ensure the robustness of each author’s ownership. We now proceed to analyze four additional requirements in the FL setting.

4.2.1 Independence. Since the aggregator has embedded $u_k$’s key set $\mathcal{K}_k$ into $M_A$, transmitted verify$_k$ and the intermediate outputs of running Merkle to $u_k$, $u_k$ can verify its ownership over $M_A$ independently. To prove its ownership, $u_k$ only needs to submit (key$_k$, verify$_k$) and a list of hashed strings to the public, from which $T_{root}$ can be correctly computed, and the ownership is proven. Such verification does not involve other co-authors or the aggregator server.

4.2.2 Privacy-preserving. The privacy-preserving property can be formulated as the following theorem:

**Theorem 1.** Under the Merkle-Sign framework, the probability that $u_k$ succeeds in forging $u_\ast$’s key is negligible.

**Proof.** We prove this statement by reduction. If an author $u_k$ can use a PPT algorithm $A_{\text{faulsify}}$ to generate a legal key and falsify itself as another author $u_\ast$ then a PPT algorithm $A_{\text{invert}}$ that inverts hash$_0$ can be built as in Algo. 2.

**Algorithm 2** PPT $A_{\text{invert}}$ that inverts hash$_0$.

**Input:** PPT algorithm $A_{\text{faulsify}}$, with which $u_k$ can falsify itself as $u_\ast$ with non-negligible probability, $y = \text{hash}_0(x)$.

**Output:** $\hat{x}$ such that $\text{hash}_0(\hat{x}) = y$.

1. $A_{\text{invert}}$ generates and distributed public and private keys for all authors.
2. $A_{\text{invert}}$ receives the key set from $u_k$ running $A_{\text{faulsify}}$.
3. $A_{\text{invert}}$ simulates Algo. 1, sets $\text{hash}_0(\text{key}_k)$ as $y$ and builds a Merkle-tree.
4. $A_{\text{invert}}$ runs $A_{\text{faulsify}}$ on the intermedia of this Merkle-tree.
5. $A_{\text{invert}}$ returns whatever $A_{\text{faulsify}}$ returns.

In Algo. 2, the environment in which $A_{\text{faulsify}}$ operates is identical to that of $u_k$ who aims to breach the privacy of $u_\ast$. $A_{\text{invert}}$ succeeds in inverting $\text{hash}_0(x)$ iff $A_{\text{faulsify}}$ successfully helps $u_k$ in falsification, so the probabilities of both events are identical. The preimage resistance assumption of $\text{hash}_0$ indicates that such probability is negligible, hence an effective $A_{\text{faulsify}}$ does not exist. □

4.2.3 Recovery. Consider the case where an adversary eavesdrops all $t$ times of $u_k$’s verifications and spoiling all keys. In this case, $u_k$ can only prove that it has successfully inverted many nodes within the Merkle-tree, yet it is unrelated to the pirated model. To recover its ownership, $u_k$ requests its neighbor w.r.t. the Merkle-tree to submit its evidence to the verification community. For example, in Figure. 4, if $u_1$ has been spoiled from the model then $u_1$ can ask $A$ to submit its key and verifier. Given this evidence, $A$ proves that it is a legal owner of the suspicious DNN model. Then $u_1$’s ownership can be recovered given the fact that its evidence $(\text{key}_1, \text{verify}_1)$ is consistent with $A$’s information within the Merkle-tree, in particular $T_{\text{root}}$ and $T_{100}$. Therefore $u_1$ must have been incorporated as $A$’s co-author before $A$ broadcasting the message as in the thirteenth step in Algo. 1. Each recovery involves proving one extra key that is consistent with the Merkle-tree shared by the victim’s co-author, so only one co-author/aggregator is involved. The recovery property is not contradictory to the privacy-preserving property as all keys are assumed to be one-time against potential spoil attacks. Spoiling all keys requires an adversary to eavesdrop on all potential co-authors, which is extremely hard. The Merkle-Sign is secure against adversarial piracy even if recovery verification is allowed.

**Theorem 2.** Under the Merkle-Sign framework, the probability of pirating a published model is negligible.

This theorem can be proven by reduction as for Theorem 1. Details are left in the appendices.

4.2.4 Traitor-tracing. A traitor $u_k$ can publish the model distributed by $A$ as its product before the training terminates. Under our configuration, $K_0$ and key$_k^\dagger$ are intractable to $u_k$, following the discussion in Section 4.2.2 and the privacy-preserving of the underlying watermarking scheme so $u_k$ cannot spoil them from the distributed model. Hence, $A$ can always successfully declare ownership over the pirated model afterward. After doing so, $A$ can examine which of the surveillance key set is in the suspicious model $M$ to locate the traitor. For example, when $A$ finds verify$_k^\dagger$ combined with the suspicious model correctly recognizes key$_k^\dagger$, i.e.:

$$\Pr \left\{ \text{verify}_k^\dagger(M, \text{key}_k^\dagger) = 1 \right\} \geq 1 - \epsilon.$$

then $u_k$ is the traitor.

4.3 Generating robust triggers for Merkle-Sign

Current black-box watermarking schemes, especially backdoor-based schemes can hardly be directly adopted in Merkle-Sign for two reasons:

- The definition of $Gen$ is ambiguous. The keyspace and the mapping from a single key to triggers are unclear.
- Most triggers lie in the same domain that is accessible for the adversary (e.g., triggers in [31, 32] are white noise). Therefore, spoiling one backdoor-back watermark brings potential harm to other watermarks. An adversary can spoil multiple watermarks once for all.

To cope with these problems and accommodate Merkle-Sign to the black-box setting, we propose a new scheme for trigger generation, Autoencoder-based Trigger Generator for Federated learning (ATGF). We resort to the autoencoder structure [17] to generate triggers, whose intermediate embedding can encode an author’s key. Since the aggregator does not possess any data in advance and lacks enough pseudorandomness to generate triggers, ATGF absorbs
patterns from authors by running the FL for an extra round as Fig. 5.

![Diagram of trigger generation process](image)

**Figure 5: Building the trigger generator in ATGF.**

Concretely, the aggregator $\mathcal{A}$ broadcasts an autoencoder structure to authors. Each author $u_k$ then trains the autoencoder into $AE_k$ on its local dataset with the legitimate domain (for privacy concerns, this dataset is not necessarily that for the FL task). Then the aggregator averages the numerical parameters across all autoencoders and obtains $AE_{\mathcal{A}} = \frac{1}{|\mathcal{A}|} \sum_{k=1}^{K} AE_k$. To generate a trigger from key, we map key into a numerical vector and feed it to $AE_{\mathcal{A}}$'s decoder. The corresponding label is obtained by running another AE decoder. The resulting trigger is given to encoders and obtains $D_{\mathcal{A}} = AE_{{\mathcal{A}}} \circ \sigma \circ \epsilon$. To generate a trigger, we map key into a numerical vector and feed it to $AE_{\mathcal{A}}$'s decoder. The corresponding label is obtained by running another AE decoder. The resulting trigger is given to encoders and obtains $D_{\mathcal{A}} = AE_{{\mathcal{A}}} \circ \sigma \circ \epsilon$.

The pseudorandomness within $AE_{\mathcal{A}}$ is attributed to all participating authors. An adversary oblivious to any evidence cannot infer the patterns of triggers from scratch. Meanwhile, since $AE_{\mathcal{A}}$ is an average of autoencoders trained on distinctive datasets, its decoder's output on different keys are subject to a mixed and complex distribution. Therefore, the spoil attack against one specific trigger in ATGF is not going to harm other triggers significantly. Such decoupling between watermarks of different authors is a necessary prerequisite for the independency, recovery, and traitor-tracing properties for Merkle-Sign against the adaptive spoil attack.

5 EXPERIMENT AND DISCUSSIONS

5.1 Settings

To evaluate the performance of Merkle-Sign and the applicability of current watermarking schemes in the FL scenario, we selected five tasks and four DNN structures for comparison.

We experimented with datasets across many domains. Apart from MNIST [5] where many studies on the security on FL have been tested, we included Fashion [25], a more complicated dataset, CIFAR10, and CIFAR100 [11], two real-world large scale datasets. Four candidate DNN structures were adopted as the backbone classifiers: an elementary multiple-layer perceptron (MLP) with cross-entropy loss, a shallow convolutional neural network (CNN) with five layers, ResNet-18, and ResNet-50 [9]. All experiments were conducted using PyTorch framework. In all scenarios, we adopted Adam optimizer with a three-stage learning rate decreasing schedule.

5.2 Revisiting watermarking schemes

To choose the appropriate watermarking scheme for Merkle-Sign, we evaluated five candidates w.r.t. the two requirements in Section 3.3, including: Uchida's [22], random trigger [31, 32], Wonder Filter (WF) [15], MTL-Sign [13], and ATGF. The scheme in [8] is insecure against tuning, while Deep-Sign [4] is slow regarding watermark embedding.

In Uchida’s, the key generation process selects $N_{\text{Uchida}} = 20$ parameters from the DNN model architecture and $N_{\text{Uchida}}$ digits in the range $[-0.5, 0.5]$ as the watermark. Watermark embedding for Uchida’s involved replacing these $N_{\text{Uchida}}$ parameters with the chosen digits. For the two established backdoor-based schemes, the key generation process randomly selects patterns as the triggers and assigns them with randomly chosen labels as in [32]. In the random trigger scheme, a random pattern with the same size as the image was generated as the key. In WF, we generated a random stamp and set its pixel value into ±2000. Watermark embedding is tantamount to having the DNN model learn the backdoor triggers.

For MTL-Sign, the key generation process selected a random set of $N_{\text{MTL}} = 20$ digits from $[1, N_{\text{MTL}}]$, mapped them into images as QR-code, and assigned them with binary labels. Watermark embedding for MTL-Sign involves minimizing the binary classification loss on a watermarking backend classifier, whose inputs are the intermediate outputs of the backbone classifier. As for ATGF, we adopted an autoencoder with ReLU activation functions whose intermediate number of neurons are: 784, 256, 32, 256, 784. For each author, five triggers were generated using ATGF.

5.2.1 Capacity. We computed the watermark capacity of the five baseline models as defined by (2). For a given model and a given dataset, the threshold of performance decline $\delta$ is set as the classification error rate of the clean model. That is to say, if the clean model achieves classification accuracy of $(1 - \delta)$ then we continue embedding watermarks into it until the accuracy declines to $(1 - 2\delta)$. The results are collected in Table 3, the maximal capacity was manually set to 10,000. We observed that complicated models have a larger capacity as expected, while simple models can hardly fit real-world datasets apart from some vanilla cases. Therefore, in the following experiments, we only adopted ResNet-50 as the backbone classifier. The weight-based and hybrid schemes Uchida’s and MTL-Sign had a larger capacity, suitable for FL and multiple-time verification. This is because these two methods, unlike backdoor-based methods, have a less negative impact on the DNN model. For ResNet-18 and ResNet-50, Uchida’s only modified at most 4.6% and 1.9% of all parameters respectively. Meanwhile, MTL-Sign, at its extreme configuration, has almost no impact on the model to be watermarked.

Among backdoor-based schemes, Wonder Filter deviate from ordinary images significantly. As a result, these triggers have less influence on the DNN’s ordinary functionality.

---

1The source code will be available at https://github.com/A_Temporary_User/MklSgn.
Table 3: Evaluation of watermarking capacity, the maximal number of correctly embedded watermarks when the classification error rate doubles. WF represents Wonder Filter, M-S represents MTL-Sign.

| Dataset  | MLP Accuracy | Uchida’s | ATGF | Rand | WF | M-S |
|----------|--------------|----------|------|------|----|-----|
| MNIST    | 91.92%       | 36       | 23   | 17   | 43 | 65  |
| Fashion  | 82.90%       | 45       | 75   | 50   | 66 | 95  |
| CIFAR10  | 25.92%       | —        | —    | —    | —  | —   |
| CIFAR100 | 13.26%       | —        | —    | —    | —  | —   |

| Dataset  | ResNet-18 Accuracy | Uchida’s | ATGF | Rand | WF | M-S |
|----------|---------------------|----------|------|------|----|-----|
| MNIST    | 99.60%              | 8,750    | 350  | 373  | 453| 9,503|
| Fashion  | 93.81%              | ≥10,000  | 519  | 513  | 588| ≥10,000|
| CIFAR10  | 89.10%              | ≥10,000  | 582  | 572  | 663| ≥10,000|
| CIFAR100 | 62.59%              | ≥10,000  | 612  | 610  | 797| ≥10,000|

5.2.2 Embedding efficiency. We recorded the time of watermark embedding for the five watermarking schemes in ResNet-50. The time consumptions for Uchida’s, random trigger, WF, MTL-Sign and ATGF trigger are respectively: 20.6ms, 262.2ms, 268.7ms, 617.3ms, and 253.2ms. Since one training epoch for ResNet-50 on average took 44s to 200s, the burden of watermark embedding in any scheme is at most upper bounded by 1.4% of training time and is uniformly negligible.

5.3 Convergence of FL under Merkle-Sign

Hitherto, the formal analysis in Section 4 has guaranteed the security of Merkle-Sign for FL. However, it remains unclear whether the DNN model can correctly converge or not when being watermarked immediately. A large capacity is necessary yet insufficient for model protection in FL.

To measure the impact of watermarking on the convergence of FL, we varied the number of authors $K$ in $\{50, 100, 200\}$. Model averaging in (1) was adopted for aggregation in the centralized setting. The decline of loss during the FL training is illustrated in Fig. 6. The final classification accuracy of different configurations of Merkle-Sign is demonstrated in Fig. 7. It can be observed that larger $K$ resulted in slower convergence, reflecting the trade-off between performance and security. The scheme of Uchida’s has the smallest impact on the convergence. The impact of the other white-box scheme, MTL-Sign, is also very small. For the three black-box/backdoor-based watermarking schemes, the convergence is at risk when $K$ approaches the watermarking capacity. Such approaching implies that watermarking exert a large impact on the model and the normal training is disturbed. However, in all cases, the decline of the final model’s classification accuracy is upper bounded by 3.1% according to Fig. 7 and is tolerable. Therefore, applying Merkle-Sign does not significantly threaten the FL system.

Figure 6: The convergence of DNN models in FL under Merkle-Sign.
5.4 Security against the spoil attack

Under Merkle-Sign, the cooperating authors are able to defend against the spoil attack by putting forward new evidence and reconstructing the Merkle-tree. However, this defense might be breached due to the flaw within the watermarking scheme: the spoil attack against one watermark may erase other watermarks at the same time. This phenomenon, especially evident in backdoor-based schemes, remains a threat to the recovery property.

To evaluate the framework’s robustness against the spoil attack under different configurations, we measured the percentage of watermarks that can be correctly verified after the spoil attack against one watermark. This metric reflects the synchronism between different watermarks within the model for a given watermarking scheme. The lower this percentage is, the less synchronism exists between watermarks and the less effective the spoil attack is.

We adopted ResNet-50 for the evaluation, it was empirically observed that this metric was almost invariant under different datasets. For Uchida’s, the spoil attack simply replaced the watermarked parameters with random numbers. For the spoil attack against MTL-Sign, we fixed the watermarking backend and tuned the DNN model until its watermarking branch failed to work. For the backdoor-based schemes, we tuned the DNN model to fit arbitrary labels on the triggers. The results are shown in Table 4.

| No. of authors | White-box scheme | Black-box scheme |
|----------------|------------------|------------------|
|                | Uchida’s | M-S | Rand | WF | ATGF | Uchida’s | M-S | Rand | WF | ATGF |
| K = 50         | 100   | 85  | 61   | 68  | 100  | 100   | 85  | 61   | 68  | 100  |
| K = 100        | 100   | 80  | 66   | 64  | 98   | 100   | 80  | 66   | 64  | 98   |
| K = 200        | 100   | 73  | 59   | 64  | 99   | 100   | 73  | 59   | 64  | 99   |

We observe from Table 4 that Uchida’s and ATGF are more robust against the spoil attack since spoiling one watermark has little impact on others. For schemes as the random trigger and WF, spoiling one single watermark can simultaneously invalidate many other watermarks. Some instances of triggers are illustrated in Fig. 8. It can be observed that triggers generated from ATGF were subject to a more diversified distribution. Concretely, the spoil attack against one author in FL compromises the identity proof of 36% and 41% oblivious authors when Merkle-Sign adopts these two schemes.

Figure 7: The convergence of DNN models in FL under Merkle-Sign.

Figure 8: Triggers generated from different watermarking schemes.

Under these configurations, the security of innocent authors and the recovery property are at risk.

Despite the privileges of Uchida’s, it is also the scheme that is the easiest to spoil. The necessary time consumption for spoiling Uchida’s, M-S, random trigger, WF, and ATGF is 21ms, 750ms, 312ms, 320ms, and 303ms. White-box watermark schemes as Uchida’s and MTL-Sign require the author to obtain and transmit the entire suspicious DNN model to the verification community for proof. In contrast, the black-box setting where the author only has to publicize the suspicious service deployed by the adversary. Among compared schemes, ATGF is the optimal choice concerning the spoil attack under the black-box setting, this is because the distribution of triggers in ATGF is hidden from the adversary. Therefore, we suggest using the combination of Merkle-Sign and ATGF, under which a high level of security is guaranteed.

6 CONCLUSION

This paper presents Merkle-Sign, a framework for DNN model protection in FL by watermarking the model. After formulating the threat model and corresponding security requirements, we extend current watermarking schemes and combine them with a data structure for distributed storage. The security of Merkle-Sign can be reduced to that of basic DNN watermarking schemes and cryptological primitives. To generalize Merkle-Sign to the black-box
setting, we propose a watermarking scheme, ATGF, that generates triggers robust against the spoil attack.

Experimental results indicated that Merkle-Sign can provide the desired security for authors in FL. Moreover, the extra requirements introduced by Merkle-Sign point the direction of designing new watermarking schemes.

The framework proposed in this paper sheds light on the availability of model protection in distributed learning systems such as FL. Merkle-Sign can be combined with other security mechanisms in FL to simultaneously protect privacy and ownership. Our future studies are going to exploit watermarking schemes that can be more conveniently combined with distributed learning systems, e.g., watermarking schemes that are robust under the aggregation operator so the embedding can be conducted in a completely decentralized manner.

ACKNOWLEDGMENTS
This work is supported by · · ·

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] M Gethisyal Augusta and Thangairulappan Katifiravalakumar. 2012. Reverse engineering the neural networks for rule extraction in classification problems. Neural processing letters 35, 2 (2012), 131–150.
[3] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical secure aggregation for privacy-preserving machine learning. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. 1175–1191.
[4] Bita Darvish Rouhani, Huili Chen, and Farinaz Koushanfar. 2019. DeepSigns: an end-to-end watermarking framework for ownership protection of deep neural networks. In Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems. 485–497.
[5] Li Deng. 2012. The mnist database of handwritten digit images for machine learning research [best of the web]. IEEE Signal Processing Magazine 29, 6 (2012), 141–142.
[6] Mohamed Elhoseny, Mahmoud Mohamed Selim, and K Shankar. 2020. Optimal Deep Learning based Convolutional Neural Network for digital forensics Face Sketch Synthesis in internet of things (IoT). International Journal of Machine Learning and Cybernetics (2020), 1–12.
[7] Clement Fung, Chris JM Yoon, and Ivan Beschastnikh. 2018. Mitigating sybilson in federated learning poisoning. arXiv preprint arXiv:1808.04866 (2018).
[8] Xinquan Guan, Huamin Feng, Weinming Zhang, Haig Zhou, Jie Zhang, and Nenghai Yu. 2020. Reversible Watermarking in Deep Convolutional Neural Networks for Integrity Authentication. In Proceedings of the 28th ACM International Conference on Multimedia. 2273–2280.
[9] 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.
[10] Minhaj Ahmad Khan and Khaleed Salah. 2018. IoT security: Review, blockchain solutions, and open challenges. Future Generation Computer Systems 82 (2018), 395–411.
[11] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features with Multi-task Learning. arXiv preprint arXiv:2003.10021 (2021).
[12] Hongwei Li, Rongxing Lu, Liang Zhou, Bo Yang, and Xuemin Shen. 2013. An efficient merkle-tree-based authentication scheme for smart grid. IEEE Systems Journal 8, 2 (2013), 635–663.
[13] 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.
[14] Esther MengelKamp, Benedikt Notheisen, Carolin Beer, David Dauer, and Christof Weinhardt. 2018. A blockchain-based smart grid towards sustainable local energy markets. Computer Science-Research and Development 33, 1 (2018), 207–214.
[15] Han Qiu, Qinkai Zheng, Tianwei Zhu, Meikang Qiu, Gerard Memmi, and Jialiang Lu. 2020. Towards secure and efficient deep learning inference in dependable IoT systems. IEEE Internet of Things Journal (2020), preprint.
[16] Tony Q Qie, and H Vincent Poor. 2020. Federated learning with differential privacy: Algorithms and performance analysis. IEEE Transactions on Information Forensics and Security 15 (2020), 3454–3469.
[17] Han Qiu, Qinkai Zheng, Tianwei Zhu, Meikang Qiu, Gerard Memmi, and Jialiang Lu. 2020. Towards secure and efficient deep learning inference in dependable IoT systems. IEEE Internet of Things Journal (2020), preprint.
[18] Nathan Shown, Tran Nguyen Ngoc, Vu Dinh Phai, and Q Shi. 2018. A deep learning approach to network intrusion detection. IEEE transactions on emerging topics in computational intelligence 2, 1 (2018), 41–50.
[19] N. Celik, and Behnam Dastani. 2019. Embedding watermarks into deep neural networks. In Proceedings of the 2019 ACM on International Conference on Multimedia Retrieval. 269–277.
[20] Tianhao Wang and Florian Kerschbaum. 2019. Robust and Undetectable White-Box Watermarks for Deep Neural Networks. arXiv preprint arXiv:1910.14268 (2019).
[21] Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H Yang, Farhad Farokhi, Shi Jin, Tony Q Qie, and H Vincent Poor. 2020. Federated learning with differential privacy: Algorithms and performance analysis. IEEE Transactions on Information Forensics and Security 15 (2019), 911–926.
[22] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. 2019. Veri- fynet: Secure and verifiable federated learning. IEEE Transactions on Information Forensics and Security 15 (2019), 911–926.
[23] Jialong Zhang, Zhongshu Gu, Jiyong Jang, Hui Wu, Marc Ph Stoecklin, Heqing Jialiang Lu. 2020. Towards secure and efficient deep learning inference in dependable IoT systems. In Proceedings of the 2020 USENIX Conference on Principles of Distributed Systems (PODS). 438–446. https://doi.org/10.1109/ICPADS55104.2020.00084
[24] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. 2019. Federated machine learning: Concept and applications. ACM Transactions on Intelligent Systems and Technology (TIST) 10, 2 (2019), 1–19.
[25] Chuanlong Yin, Yuifei Zhu, Jilong Fei, and Xinzheng He. 2017. A deep learning approach for intrusion detection using recurrent neural networks. Jee Access 5 (2017), 21954–21961.
[26] Jie Zhang, Dongdong Chen, Jing Liao, Weinming Zhang, Huamin Feng, Gang Hua, and Nenghai Yu. 2021. Deep Model Intellectual Property Protection via Deep Watermarking. IEEE Transactions on Pattern Analysis and Machine Intelligence (2021).
[27] 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.
[28] Renjie Zhu, Xinpeng Zhang, Mengte Shi, and Zhenjun Tang. 2020. Secure neural network watermarking protocol against forgery attack. EURASIP Journal on Image and Video Processing 2020, 1 (2020), 1–12.

A THE PROOF OF THEOREM 2

Proof. To pirate a model under this verification framework, an adversary can take three possible approaches: (1) Broadcasting a fake message with an early time-stamp than $A_i$’s, writing its key into the model, and declaring ownership. (2) Pretending to be an author that has participated in FL. (3) Pretending to be an author whose key has been spoiled.

The first attack succeeds with negligible probability since the adversary has to correctly guess the structure of the DNN model to correctly provides info. Moreover, the adversary has to tune the DNN model so its watermarking branch is consistent with its key, during which process the verifier module is fixed. This process might bring damage to the model’s performance.

For the second attack, if the adversary has not eavesdropped any verification proof then it has to find the preimage of $T_{\text{root}}$ under hashb to forge a legal key. Hence a PPT adversary $A$ pretend that
Algorithm 3 PPT $\mathcal{A}_{\text{collide}}$ that finds a collision for hash$_2$.

Input: A PPT algorithm $\mathcal{A}_{\text{pretend}}$, with which an adversary can falsify itself as $u_t$, without eavesdropping any proof with non-negligible probability.

Output: $x$ and $y$ such that hash$_2(y) = \text{hash}_2(x)$.

1. $\mathcal{A}_{\text{invert}}$ generates and distributed public and private keys for all authors.
2. $\mathcal{A}_{\text{collide}}$ receives the key from the adversary running $\mathcal{A}_{\text{pretend}}$.
3. $\mathcal{A}_{\text{collide}}$ simulates Algo. 1, builds a Merkle-tree with intermediate nodes $T_0$ and $T_1$.
4. $\mathcal{A}_{\text{collide}}$ runs $\mathcal{A}_{\text{pretend}}$ on the root node of its Merkle-tree.
5. $\mathcal{A}_{\text{collide}}$ receives $\mathcal{A}_{\text{pretend}}$’s output, which contains enough information for computing $T_{\text{root}}$, especially nodes of the second level $T_0$ and $T_1$.
6. $\mathcal{A}_{\text{collide}}$ returns $T_0\|T_1$ and $T_0\|T_1$.

Therefore:

$$\Pr([\mathcal{A}_{\text{collide}} \text{ wins}] \cap \{T_0\|T_1 \neq T_0\|T_1\}) = \Pr([\mathcal{A}_{\text{collide}} \text{ wins}] \cap \{T_0\|T_1 \neq T_0\|T_1\})\cdot \Pr([\mathcal{A}_{\text{pretend}} \text{ wins}] \cap \{T_0\|T_1 \neq T_0\|T_1\}).$$

Recall that the collision resistance of hash$_2$ implies that $\Pr([\mathcal{A}_{\text{collide}} \text{ wins}])$ is upper bounded by a negligible function, which indicates that:

$$\Pr([\mathcal{A}_{\text{pretend}} \text{ wins}] \cap \{T_0\|T_1 \neq T_0\|T_1\}) = \Pr([\mathcal{A}_{\text{pretend}} \text{ wins}] \cap \{T_0\|T_1 \neq T_0\|T_1\}) - \text{non-negligible}.$$  

This further implies that $\mathcal{A}_{\text{pretend}}$ can efficiently inverse hash$_2$ with non-negligible probability, which is contradictory to the fact that hash$_2$ is also preimage resistant. Therefore such $\mathcal{A}_{\text{pretend}}$ does not exist.

If the adversary has eavesdropped a verification proof then it has to find the preimage of a string under hash$_1$ for at least once. Similar to Algo. 2, the impossibility of such an attack is reduced to the security of hash$_1$.

For the third attack, the adversary has to invert hash$_1$ for at least once, which succeeds only with negligible probability as in the second case in the second attack.

B Merkle-Sign FOR PEER-TO-PEER FL

The proposed framework can be generalized to peer-to-peer (P2P), or decentralized FL, in which no aggregator is involved in the entire process. In P2P FL, authors form a chain along with the DNN model is trained and transmitted. An illustration of the P2P FL is in Fig. 9.

To adapt to P2P FL, Merkle-Sign requires all authors $\mathcal{U}$ follow Algo. 4 as shown in Figure 10. The function hash$_1$ adopted by author $u_t$ in the seventh and the eleventh steps is $\text{Enc}_{(t)}(\text{hash}_0(\cdot))$. Finally, $u_{(T)}$ uses $\text{Enc}_{(T)}(\text{hash}_0(\cdot))$ to mask its key set and verifiers in the twelfth step.

---

2Collision resistance implies preimage resistance.
has to be larger than $T$ to ensure the model’s performance. The basic security requirements, the independency, the privacy-preserving, and the recovery properties hold with similar discussion as in the aggregator-based case. For traitor-tracing, the authors $U$ follow

\begin{algorithm}
\caption{Traitor-tracing under Merkle-Sign for P2P FL.}
\begin{algorithmic}[1]
\Input The pirated model $M$.
\Output The index of the traitor.
\For{$t = 1$ to $T$}
\State $u_{(t)}$ proves its ownership over $M$.
\State $u_{(t)}$ presents to its co-authors: $\text{key}^\dagger_{(t)}$, $\text{verify}^\dagger_{(t)}$, and \{hash1($\text{key}_{(t)}$), hash1($\text{verify}_{(t)}$)\}.
\State Other authors check whether the presented information is consistent with $u_{(t)}$’s historical broadcasting.
\State Other authors vote on $\text{verify}^\dagger_{(t)}(M, \text{key}^\dagger_{(t)})$.
\State If the voted result is zero then return $(t - 1)$.
\EndFor
\State Return $(T)$.
\end{algorithmic}
\end{algorithm}

\textbf{Theorem 3.} If $u_{(t)}$ is the traitor who sells the model, then Algo. 5 can almost always correctly identify it.

\textbf{Proof.} We assume that $u_{(t)}$ follows Algo. 4 except for its unauthorized reselling. Otherwise, it cannot verify and protect its contribution in the model, and it might be identified as the traitor from the fourth step in Algo. 5. Compared with the model released by $u_{(t)}$ to $u_{(t+1)}$, the model in $u_{(t-1)}$’s perspective does not contain $\text{key}^\dagger_{(t)}$. The secrecy of $\text{key}^\dagger_{(t)}$ is protected by the CPA-security, the covertness, and the privacy-preserving property of the underlying watermarking scheme, together with the one-wayness of Merkle. If $u_{(t-1)}$ can successfully embed this surveillance key onto the model and escape tracing then it must have obtained $\text{key}^\dagger_{(t)}$, which happens only with negligible probability. By the definition of the correctness of WM, $u_{(t-1)}$ cannot forge the evidence for $\text{key}^\dagger_{(t)}$. Therefore, Algo. 5 is sufficient to identify $u_{(t-1)}$ as the traitor. \hfill $\Box$

Figure 10: The Merkle-Sign watermarking framework for P2P FL.

Algo. 5. The correctness is guaranteed by the following theorem.