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

Paper ID: 4555

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) raises new challenges for model protection. Apart from the independent verification of each author's ownership, the author collaboration should be able to recover its participant's identity against adaptive attacks and trace traitors. To meet those requirements, we propose a watermarking protocol, `Merkle-Sign`, for deep neural networks protection in FL. By incorporating state-of-the-art watermarking schemes and the cryptological primitive designed for distributed storage, this protocol meets the prerequisites for ownership verification in FL. Our work paves the way for generalizing watermark as a practical security mechanism for protecting deep learning models in distributed learning platforms.

Introduction

Deep neural networks (DNN) are intelligent systems that provide services by learning from data and consuming enormous computational resources. 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 (IP) by proving authors' ownership.

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 multimedia objects. 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.

Modern paradigms of model training and distributing such as Federated Learning (FL) (Li et al. 2020) 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, increasing 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 protocol for ownership verification, `Merkle-Sign`, is proposed to meet all the requirements for FL. We also design a new watermarking scheme, `ATGF`, to increase the applicability of `Merkle-Sign`.
- Experiments demonstrate the utility of the proposed framework and examine the capability of established watermarking schemes in FL.

Backgrounds and Preliminaries

DNN watermark

Let M_{clean} be a model trained to fulfil a primary task $\mathcal{T}_{\text{primary}}$. The author embeds its identity information into the model, which can later be revealed. A watermarking scheme $\texttt{WM} = \{\texttt{Gen}, \texttt{Embed}\}$ consists of one module for identification key generation, $\texttt{key} \leftarrow \texttt{Gen}(1^N)$, and one for key embedding $(M_{\text{WM}}, \texttt{verify}) \leftarrow \texttt{Embed}(M_{\text{clean}}, \texttt{key})$, where N is the security parameter. When the author finds that its model has been stolen, it provides $\{\texttt{key}, \texttt{verify}\}$ as its evidence. If `verify` agrees with `key` concerning the suspicious model then the author's ownership is proven. Such proof can be conducted and announced by a trusted third party, yet it introduces extra security risks and a tremendous burden to the center. Instead, as (Mengelkamp et al. 2018), the proof as a service can be done in a public and decentralized manner so any party agreeing a consensus protocol can participate (Li, Wang, and Alan 2021). The legitimate OV outcome is voted across the entire community to defend against potential conspiracy and perjury.

Watermarks are usually backdoors (Zhang et al. 2018; Adi et al. 2018) for the black-box setting. Adversarial samples (Le Merrer, Perez, and Trédan 2020) and out-of-range samples (Li et al. 2019), have been adopted to generate backdoor triggers. The white-box watermarking schemes are pioneered by (Uchida et al. 2017), which embeds the owner's digital signature into the model's weight (Liu, Weng, and Zhu 2021). `Deep-Sign` (Darvish, Chen, and Koushanfar 2019) builds the watermark on the intermediate output of the DNN on certain samples. `MTL-Sign` (Li and Wang 2021) models watermarking embedding as an extra learning task. Yet most proposals of DNN watermarking overlooked the top-level protocol for key generation and verification. A practical *watermarking protocol* must include both the bottom-level watermarking scheme and the top-level OV protocol.

A DNN watermarking scheme has to satisfy the following basic security requirements.

Correctness The author can prove its ownership correctly:

$$\Pr\{\texttt{verify}(M_{\text{WM}}, \texttt{key}) = 1\} \geq 1 - \epsilon,$$

where ϵ is a negligible function of the security parameter N. The following unambiguity condition also needs to hold:

$$\Pr\{\texttt{verify}(M_{\text{WM}}, \texttt{key}') = 1\} \leq \epsilon, \qquad (1)$$

in which $\texttt{key}' \neq \texttt{key}$ is the adversary's key.

Functionality-preserving The watermarking cannot substantially decrease the model's functionality, i.e., the performance of M_{WM} should be similar to that of M_{clean}.

Security against tuning The adversary can tune a model on its local dataset or prune neurons. Such tuning could ruin schemes like the reversible watermark (Guan et al. 2020). A watermarking scheme is secure against tuning if tuning cannot invalidate the verification.

Covertness The watermark should be hidden from the adversary. A necessary condition for the watermark's covertness is: the parameters of M_{WM} should not deviate from those of M_{clean} too much (Wang and Kerschbaum 2019).

Security against overwriting An adversary can embed its watermark into the model and *redeclare* the overwritten model as its product. The redeclaration can only be solved by authorizing the time-stamp correlated with the DNN architecture and the identification under a high-level OV protocol (Li, Wang, and Alan 2021).

Federated learning

Federated Learning (FL) (Yang et al. 2019) associates a broader range of data sources while preserving the privacy of each participant. A collection of K participants $\mathcal{U} = \{u_k\}_{k=1}^{K}$ and an aggregator $\mathcal{A}$ cooperate to train a model. During the entire procedure, each author only communicates with the aggregator through an encrypted channel. The aggregator distributes the model M to all authors. Each author u_k independently computes the direction to which the model should evolve w.r.t. its local dataset $\mathcal{D}_k$ and transmits the gradient ΔM_k back. The aggregator collects the feedback from all authors, updates the model direction $\sum_{k=1}^{K} |\mathcal{D}_k| \cdot \Delta M_k$, and starts the next round. Model aggregation can also be homomorphic encryption-based weighted average, etc (Bonawitz et al. 2017).

Current studies on the security of FL focus on data protection against curious or malicious aggregators or authors (Wei et al. 2020). Literatures on model protection in FL are relatively scanty besides the persistency against malicious authors during watermarking (Atli et al. 2020).

Security Requirements

Requirements for OV in FL

Apart from the ordinary demands, the FL setting gives rise to extra security requirements as Fig. 1 and Fig. 2.

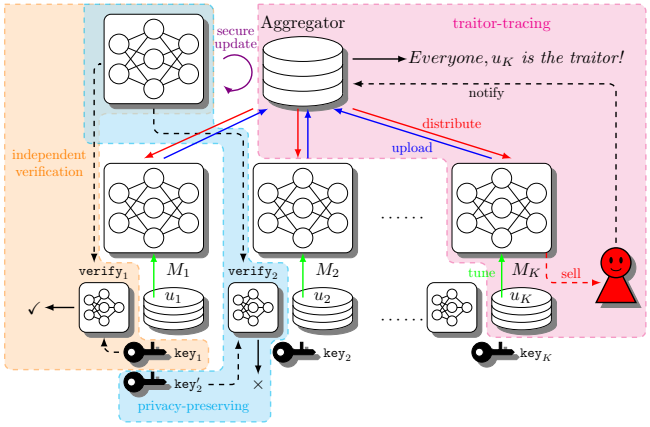


Figure 1: Security requiremetns of OV in FL: independency, privacy-preserving, and traitor-tracing.

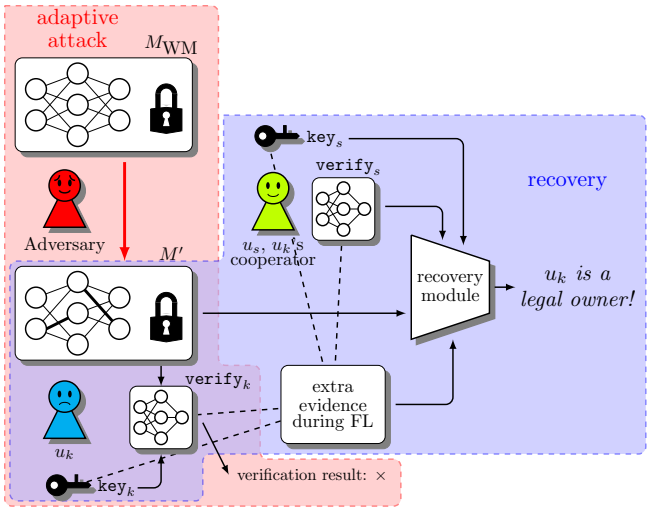


Figure 2: Security requiremetns of OV in FL: recovery.

Independency When FL terminates, each author u_k holds an evidence pair $(\texttt{key}_k, \texttt{verify}_k)$. After the trained model

Table 1: Dependence of the advanced requirements on basic security requirements.

Advanced requirements	Basic security requirements				
	Correctness.	Functionality-preserving.	Security against tuning.	Covertness.	Security against overwriting.
Independency.	Relevant	Relevant	Irrelevant	Irrelevant	Relevant
Privacy-preserving.	Relevant	Irrelevant	Irrelevant	Relevant	Irrelevant
Traitor-tracing.	Relevant	Relevant	Relevant	Relevant	Irrelevant
Recovery.	Relevant	Relevant	Irrelevant	Irrelevant	Irrelevant

M is published, it is necessary that:

$$\Pr\{\texttt{verify}_k(M, \texttt{key}_k) = 1\} \geq 1 - \epsilon.$$

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

Privacy-preserving For author $u_s \neq u_k$, no information about u_k's identity, especially $\texttt{key}_k$, appears in u_s's perspective when interacting with the aggregator. Formally, u_s cannot succeed in decepting $\texttt{verify}_k$.

Traitor-tracing Watermark can trace unauthorized reselling in DNN commercialization (Xu et al. 2020). In FL, a traitor may participate in a few epochs of training, obtain the intermediate model, then claim the current model as its product. The aggregator supervising the FL training process and the collaboration of honest authors in the decentralized setting should be able to identify the traitor.

Recovery Once u_k proves its ownership over M, an eavesdropping adversary might conduct the spoil attack (Li, Wang, and Alan 2021) 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 removal u_k's watermark is insufficient to disprove u_k's ownership.

The dependency of these four advanced requirements on the basic security properties is presented in Table. 1.

Requirements for the watermarking scheme in FL

The underlying watermarking scheme for DNN IP protection protocol should have: (i) large capacity to incorporate the evidence for each author, (ii) efficient embedding so leaving evidence for traitor-tracing is feasible. A DNN model's *watermark capacity* w.r.t. $\texttt{WM}$ and the decline of its performance δ, $\texttt{cap}^{\delta}_{\texttt{WM}}$ is defined as follows:

Definition 1: *$\texttt{cap}^{\delta}_{\texttt{WM}}$ is the maximal number of keys that can be correctly embedded by $\texttt{WM}$ into the model until the model's performance drops by δ w.r.t. the metric $\mathcal{E}$ defined by its corresponding primary task.*

$\texttt{cap}^{\delta}_{\texttt{WM}}$ measures the upper bound of the number of correctly embedded watermarks inside a model. Formally, $\texttt{cap}^{\delta}_{\texttt{WM}}$ is the maximal q satisfying the following conditions:

$$\begin{gathered}(M_1, \texttt{verify}_1) \leftarrow \texttt{Embed}(M_{\text{clean}}, \texttt{key}_1),\\ (M_2, \texttt{verify}_2) \leftarrow \texttt{Embed}(M_1, \texttt{key}_2),\\ \dots\\ (M_q, \texttt{verify}_q) \leftarrow \texttt{Embed}(M_{q-1}, \texttt{key}_q),\\ \mathcal{E}(M_q) \geq \mathcal{E}(M_{\text{clean}}) - \delta,\end{gathered}$$

where all q watermarks can be correctly verified. Deeper DNNs model should have the larger capacity since the watermark usually modifies only a small number of parameters.

The Merkle-Sign Framework

Motivation

To reduce the cost of authorization and traffic, we take the merit of Merkle-tree, a data structure for integrity verification in cloud storage (Li et al. 2013). Merkle-tree allows partial authentication of a subset of its input, this property forms the basis for independent verification and recovery.

`Merkle-Sign` for FL

Our protocol, `Merkle-Sign`, operates as Algo. 1 and is visualized in Figure 3. In which `Merkle` is a Merkle-tree combinator. The 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 δ, it is necessary that $\texttt{cap}^{\delta}_{\texttt{WM}} \geq (K+1)$.

To trace the traitor, the intermediate model $M_{\text{WM},k}$ sent to u_k contains a surveillance key, $\texttt{key}^{\dagger}_k$. This key is only known to $\mathcal{A}$, from which it can identify the traitor.

In `Merkle`, $\texttt{h}_1$ maps any legal `key`, `verify`, or `info` into $\{0,1\}^r$, while $\texttt{h}_2$ is a collision resistant hash function that maps $\{0,1\}^{2r}$ into $\{0,1\}^r$. Operator `Merkle` maps each component of its input by $\texttt{h}_1$ then organizes the mapped values into a binary tree with $\texttt{h}_2$ as illustrated in Fig. 4. Concretely, $\texttt{h}_1$ for $\texttt{key}_k$ in the 7th and the 13th line in Algo. 1 is a digital signature scheme, $\texttt{Enc}_k(\texttt{h}_0(\cdot))$, where $\texttt{Enc}_k$ is an encryption module using u_k's private key, and $\texttt{h}_0$ is a preimage resistant hash function. A third party can examine whether a given $\texttt{key}_k$ corresponds to $w = \texttt{h}_1(\texttt{key}_k)$, which are part of the evidence submitted by u_k during OV. It decrypts w using u_k's public key and compares the plaintext with $\texttt{h}_0(\texttt{key}_k)$.

Algorithm 1: `Merkle-Sign` protocol for FL.

Require: A watermarking scheme `WM`, security parameters N and L, hash functions $\mathtt{h}_0$, $\mathtt{h}_1$ and $\mathtt{h}_2$. A verification community regulated by a consensus protocol.

Ensure: A watermarked model $M_{\mathcal{A}}$ and evidence.

1: $\mathcal{A}$ generates $\mathtt{key}_0 \leftarrow \mathtt{Gen}(1^N)$.
2: Each $u_k \in \mathcal{U}$ generates $\mathtt{key}_k \leftarrow \mathtt{Gen}(1^N)$, submits it and $\mathtt{hash}_1(\mathtt{key}_k|u_k)$ to $\mathcal{A}$.
3: For each u_k, $\mathcal{A}$ generates $\mathtt{key}_k^\dagger \leftarrow \mathtt{Gen}(1^N)$.
4: $\mathcal{A}$ initializes a clean model.
5: **while** not terminate **do**
6: For each u_k, $\mathcal{A}$ embeds $\mathtt{KEYs}_k = (\mathtt{key}_0, \mathtt{key}_k^\dagger)$ into the current model as $M_{\mathrm{WM},k}$, obtains $\mathtt{VERs}_k = (\mathtt{verify}_0, \mathtt{verify}_k^\dagger)$.
7: $\mathcal{A}$ signs and broadcasts to the community:

$$\langle \mathtt{time} \| \mathtt{Merkle}(\mathtt{KEYs}_k, \mathtt{VERs}_k, \mathtt{info}) \rangle.$$

8: $\mathcal{A}$ transmits $M_{\mathrm{WM},k}$ 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 $\mathtt{KEYs} = \{\mathtt{key}_k\}_{k=0}^K$ into the final model $M_{\mathcal{A}}$, obtains $\mathtt{VERs} = \{\mathtt{verify}_k\}_{k=0}^K$.
13: $\mathcal{A}$ signs and broadcasts to the community:

$$\langle \mathtt{time} \| \mathtt{Merkle}(\mathtt{KEYs}, \mathtt{VERs}, \mathtt{info}) \rangle.$$

14: $\mathcal{A}$ sends the intermedia of `Merkle(KEYs, VERs, info)` and $\mathtt{verify}_k$ to u_k.
15: $\mathcal{A}$ publishes $M_{\mathcal{A}}$.

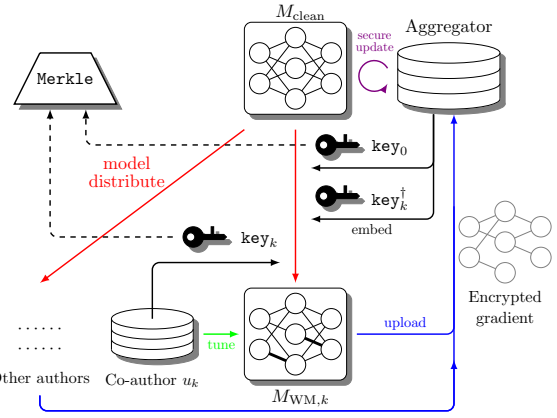


Figure 3: The `Merkle-Sign` watermarking framework for centralized FL.

Meanwhile, $\mathtt{h}_1$ for the surveillance keys and all verifiers in Algo. 1 is the encryption module using $\mathcal{A}$'s private key combined with $\mathtt{h}_0$.

The output of `Merkle` is its root $T_{\text{root}} \in \{0,1\}^r$, the intermedia are the values of all the remaining nodes. An instance for `Merkle` for $K = 2$ authors is demonstrated in Fig. 4. To justify the ownership, an author, e.g., u_1 submits

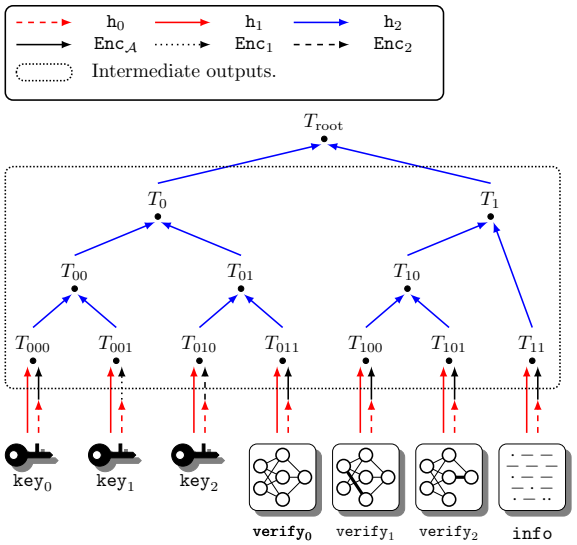


Figure 4: `Merkle` on three keys.

$\mathtt{key}_1$, $\mathtt{verify}_1$, $\mathtt{info}$ and the necessary information for deriving T_{root}, i.e., T_{001}, T_{100}, T_{11}, T_{000}, T_{01}, and T_{101} to the community providing OV service. Any party given the public key of u_k and the access to $\mathtt{h}_0$, $\mathtt{h}_2$ can independently examine whether the evidence is consistent with the suspicious model and the recorded time-stamps.

Since the length of the root of the Merkle-tree is a constant, the traffic in authorization during each epoch (the 7th line in Algo. 1) does not grow with K. The burden is shifted to the identity proof, where an extra amount of information that is the same order as the height of the Merkle-tree, i.e., $\mathcal{O}(\ln K)$, needs to be submitted. We now analyze four additional requirements in FL.

Independency The aggregator has embedded $\mathtt{key}_k$ into the DNN product, transmitted $\mathtt{verify}_k$ and the intermediate outputs of runing `Merkle` to u_k. Therefore, u_k can verify its ownership over $M_{\mathcal{A}}$ by presenting $(\mathtt{key}_k, \mathtt{verify}_k)$ and a list of hashed strings, from which T_{root} can be correctly recovered. Such OV does not involve other cooperators.

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

Theorem 1: Under the `Merkle-Sign` protocol, the probability that u_s succeeds in passing $\mathtt{verify}_k$ is negligible.

Proof: We prove this statement by reduction. If u_s can use an algorithm $\mathcal{A}_{\text{falsify}}$ to generate a legal key and passes $\mathtt{verify}_k$ then an algorithm $\mathcal{A}_{\text{invert}}$ that inverts $\mathtt{h}_0$ can be built as in Algo. 2.

In Algo. 2, the environment in which $\mathcal{A}_{\text{falsify}}$ operates is identical to that of u_s who aims to breach the ownership of u_k. $\mathcal{A}_{\text{invert}}$ succeeds in inverting $\mathtt{h}_0(x)$ iff $\mathcal{A}_{\text{falsify}}$ succeeds, so the probabilities of both events are identical. The preimage resistance assumption of $\mathtt{h}_0$ and the unambiguity condition (1) indicate that such probability is negligible, hence an effective $\mathcal{A}_{\text{falsify}}$ does not exist.

Algorithm 2: $\mathcal{A}_{\text{invert}}$ that inverts $\mathtt{h}_0$.

Require: algorithm $\mathcal{A}_{\text{falsify}}$, with which u_s can pretend to be u_k with non-negligible probability, $y = \mathtt{h}_0(x)$.
Ensure: $\tilde{x}$ such that $\mathtt{h}_0(\tilde{x}) = y$.
1: Generate and distribute public and private keys for all authors.
2: Simulate Algo. 1 with $\mathtt{h}_0(\mathtt{key}_k) = y$.
3: Run $\mathcal{A}_{\text{falsify}}$ on the intermedia of the Merkle-tree and $\mathtt{verify}_k$.
4: Return whatever $\mathcal{A}_{\text{falsify}}$ returns.

Traitor-tracing Since $\mathtt{key}_0$ and $\mathtt{key}_k^\dagger$ are intractable to the traitor u_k following the covertness of the underlying watermarking scheme, u_k cannot spoil them from the distributed model. $\mathcal{A}$ can examine the surveillance key from the pirated model to locate the traitor. For example, when A finds $\mathtt{verify}_k^\dagger$ combined with the pirated model recognizes $\mathtt{key}_k^\dagger$, i.e., $\mathtt{verify}_k^\dagger(M, \mathtt{key}_k^\dagger) = 1$, then u_k is the traitor.

Recovery An adversary eavesdropping u_k's OV can spoil $\mathtt{key}_k$ so u_k's evidence is no longer related 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, u_1 can ask $\mathcal{A}$ to submit its evidence and proves that it is a legal owner of the suspicious DNN model. Then u_1's ownership is recovered given the fact that its evidence $(\mathtt{key}_1, \mathtt{verify}_1)$ is consistent with $\mathcal{A}$'s information within the Merkle-tree, in particular T_{001} and T_{100}. Therefore u_1 must have been incorporated as $\mathcal{A}$'s co-author before $\mathcal{A}$ broadcasting the message as in the 13th step in Algo. 1. Each recovery involves one extra evidence, so only one co-author is affected and malicious authors cannot actively sabotage this process. Spoiling all keys requires an adversary to eavesdrop on all OV requests of potential co-authors, which is extremely hard. `Merkle-Sign` remains secure against piracy even if the recovery mechanism is allowed.

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

Proof: The proof by reduction is similar to that of Theorem 1, see Appendix. A for details.

Generating robust triggers for `Merkle-Sign`

Current backdoor-based schemes can hardly be directly adopted in `Merkle-Sign` for two reasons: (i) The definition of `Gen` is ambigious, and the correlation between keys and triggers are ambigious. (ii) Most triggers lie in the same domain that is accessible for the adversary (e.g., white noise). Therefore, spoiling the watermark of one author brings potential harm to the ownership of other authors.

To accommodate to the black-box setting, we design a new trigger generation scheme, Autoencoder-based Trigger Generator for Federated learning (`ATGF`).

As illustrated in Fig. 5, the aggregator $\mathcal{A}$ distributes an autoencoder structure `AE`. Each author u_k then trains the autoencoder into $\mathtt{AE}_k$ on its local dataset (this dataset is not necessarily that for the FL). Then $\mathcal{A}$ averages the numerical parametes across all autoencoders and obtains:

$$\mathtt{AE}_{\mathcal{A}} = \frac{1}{K}\sum_{k=1}^{K} \mathtt{AE}_k.$$

To generate triggers, `key` is mapped into a collection of numerical vectors and fed into $\mathtt{AE}_{\mathcal{A}}$'s decoder, $\mathtt{Dec}_{\mathcal{D}}$. The corresponding labels are given by another pseudorandom mapping conditioned on `key`.

As an average of autoencoders trained on distinctive datasets, the outputs of $\mathtt{Dec}_{\mathcal{D}}$ are subject to a mixed and complex distribution. Therefore, the spoil attack against any specific trigger from `ATGF` is not going to harm other triggers significantly, hence ensuring the security of `Merkle-Sign` against adaptive attacks.

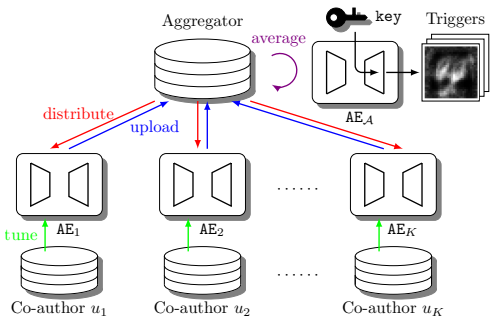


Figure 5: Building the trigger generator in `ATGF`.

Experiment and Discussions

Settings

To study the performance of `Merkle-Sign` and the applicability of current watermarking schemes in the FL scenario, we selected five tasks: MNIST (Deng 2012), Fashion (Xiao, Rasul, and Vollgraf 2017), CIFAR10, and CIFAR100 (Krizhevsky, Hinton et al. 2009) with DNN architectures ResNet-18 and ResNet-50 (He et al. 2016) for evaluation. All experiments were conducted in `PyTorch`. We adopted Adam optimizer with a three-stage learning rate decreasing schedule.

Revisiting watermarking schemes

To test the adapticity of established watermarking schemes with `Merkle-Sign` protocol, we evaluated six SOTA candidates, including: Uchida (Uchida et al. 2017), random trigger (Zhu et al. 2020), `Wonder Filter` (`WF`) (Li et al. 2019), `MTL-Sign` (`MS`) (Li and Wang 2021), `WAFFLE` (Atli et al. 2020), and `ATGF`. Note that privacy-preserving, traitor-tracing, and recovery have been secured by Theorem 1, the discussion in the section before, and Theorem 2 respectively. Therefore, we are then interested in the upper bound of authors assuming reliable independency OV and the operating time for traitor-tracing and recovery.

In Uchida, the key generation process selects 20 parameters from the DNN model architecture and 20 digits in

Table 2: Evaluation of watermarking capacity on ResNet-18 (left) and ResNet-50 (right).

Dataset	Watermarking schemes					
	Uchida	Rand	WF	MS	WAFFLE	ATGF
MNIST	8750	373	453	9503	312	350
Fashion	≥10000	513	588	≥10000	520	519
CIFAR10	≥10000	572	663	≥10000	543	582
CIFAR100	≥10000	610	797	≥10000	642	612

Dataset	Watermarking schemes					
	Uchida	Rand	WF	MS	WAFFLE	ATGF
MNIST	≥10000	411	494	≥10000	398	417
Fashion	≥10000	540	669	≥10000	565	580
CIFAR10	≥10000	612	773	≥10000	598	600
CIFAR100	≥10000	712	779	≥10000	744	710

the range $[-0.5, 0.5]$ as the watermark. In the random trigger scheme, WF, and WAFFLE, we generated 10 images for each author as backdoor triggers. For MS, the key generation process selected a random set of 20 digits from $[1, 8000]$, mapped them into QRcodes, and assigned them with binary labels. As for ATGF, we adopted an autoencoder with ReLU activation functions whose intermediate number of neurons are: 784, 256, 32, 256, 784. Each local autoencoder is trained on $\frac{1}{K}$ of the MNIST dataset. For each author, 10 triggers were generated using ATGF.

The capacity of independent OV We computed the watermark capacity of all schemes as defined by (2), which measures the upper bound of K that ensures independent OV for each author. For a given model and a given dataset, the threshold of performance decline δ is set as the classification error rate of the original clean model. The results are collected in Table 2, the maximal capacity was set to 10,000. We observed that the more complicated model have a larger capacity as expected, while simple models' capacity is poor apart from some vanilla cases, more details are provided in Appendix. . Uchida and MS had larger capacity since these weight-based schemes have a less impact on the DNN model. Among backdoor-based schemes, WF had the largest capacity, since triggers adopted in WF deviate from ordinary images significantly. For these schemes, the capacity becomes a bottleneck when $K \geq 300$.

Efficiency of traitor-tracing and recovery In Merkle-Sign, traitor-tracing is done by examing all surveillance keys, and recovery is done by examing one key and reconstructing the Merkle-tree, so their respective complexity is $\mathcal{O}(K)$ and $\mathcal{O}(\ln K)$. We recorded the time of watermark embedding for the five watermarking schemes for ResNet-50, $K = 200$ in Table. 3. One tuning epoch for ResNet-50 took 44s to 200s averagely, it is observed that the burden of watermarking is at most upper bounded by 1.6% during the normal training process and is uniformly negligible.

Convergence of FL under Merkle-Sign

Although the security of Merkle-Sign has been guaranteed in theory, it remains unclear whether the DNN can correctly converge or not when being watermarked intermediately. To measure the impact of watermarking on the convergence, we varied the number of authors K in $\{50, 100, 200\}$, adopted model averaging for aggregation, and conducted FL. The decline of loss during the FL training and the final classification loss are illustrated in Fig. 6 and Fig. 7. The scheme of Uchida has the smallest impact on the convergence, that of the other white-box scheme, MS, is also very small. For the three backdoor-based watermarking schemes, the convergence is at risk when K approaches the watermarking capacity. However, in all cases, the decline of the final model's classification accuracy is upper bounded by 3.1% and is tolerable. Therefore, applying Merkle-Sign does not significantly threaten the FL system.

Table 3: Time consumption for security functions.

Task	Uchida	M-S	Rand	WF	WAFFLE	ATGF
Traitor-tracing (s)	2.1	26.2	27.0	62.5	24.3	25.4
Recovery (ms)	121	362	369	717	334	353

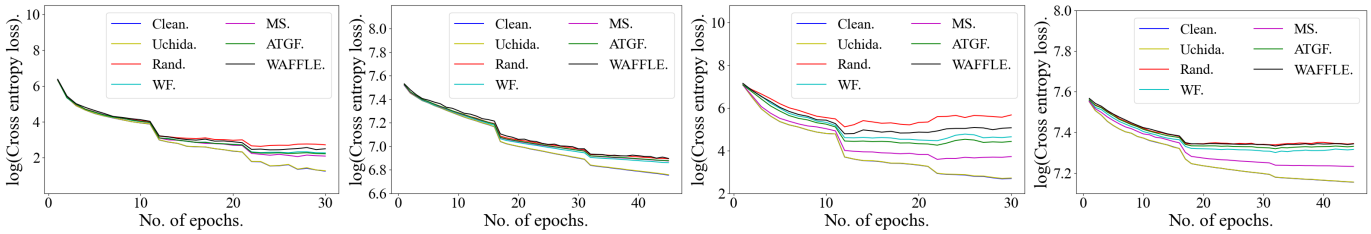
Security against the spoil attack

Under Merkle-Sign, the cooperating authors can recover a spoiled identity proof by putting forward new evidence and reconstructing the Merkle-tree. However, this defense might be breached since the spoil attack against one watermark may erase the watermarks of other authors (unrevealed to the adversary) at the same time. This phenomenon, especially evident in backdoor-based schemes, is a threat to the recovery property.

To evaluate the security against the spoil attack under different configurations, we measured the percentage of watermarks that can be correctly verified after undertaking the spoil attack against one watermark. This metric reflects the correlation between different watermarks within the model. The lower this percentage is, the less correlated are the watermarks and the less effective the spoil attack is.

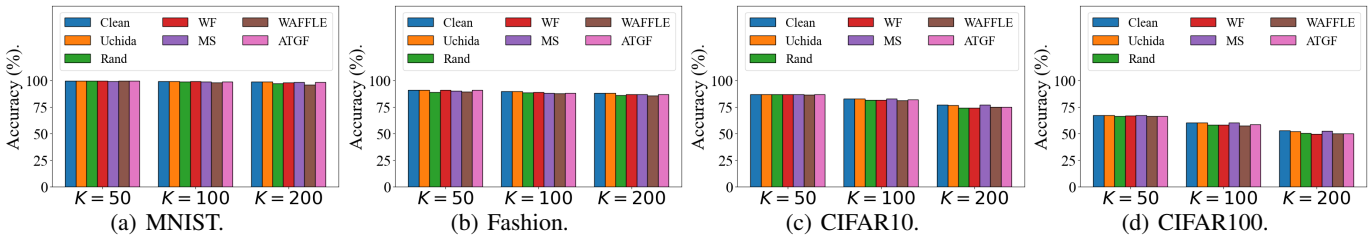
We adopted CIFAR10 for evaluation (it was empirically observed that this metric was almost invariant under different datasets). For Uchida, the spoil attack simply replaced the watermarked parameters with random numbers. For the spoil attack against MS, we fixed the watermarking backend and tuned the DNN model until its watermarking branch failed to work. For the backdoor-based schemes, we tuned the DNN model to fit arbitrary labels on one set of triggers. The results are shown in Table 4.

Table 4 indicates that white-box schemes are more ro-



(a) MNIST, $K = 50$. (b) CIFAR100, $K = 50$. (c) MNIST, $K = 200$. (d) CIFAR100, $K = 200$.

Figure 6: The validation loss of ResNet-50 in FL under `Merkle-Sign`.



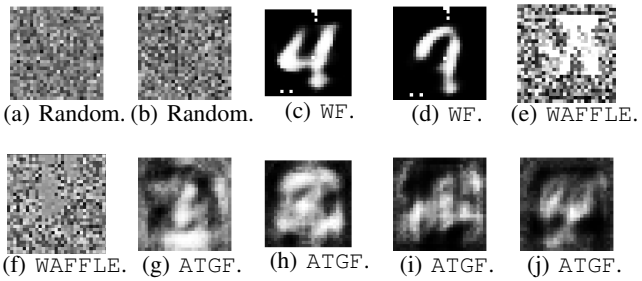
(a) MNIST. (b) Fashion. (c) CIFAR10. (d) CIFAR100.

Figure 7: The classification accuracy of ResNet-50 in FL under `Merkle-Sign`.

Table 4: Percentage of correctly verified watermarks (in %).

K	White-box schemes		Black-box schemes			
	Uchida	M-S	Rand	WF	WAFFLE	ATGF
50	100	85	61	68	71	**100**
100	100	80	66	64	69	**98**
200	100	73	59	64	69	**99**

bust against the spoil attack since spoiling one watermark has little impact on others. For schemes as the random trigger, `WF`, and `WAFFLE`, spoiling one single watermark can simultaneously invalidate many others. Under these configurations, the security of innocent authors and the recovery property are at risk. Instead, triggers generated from `ATGF` are subject to a more diversified distribution and resist such correlated spoil, as demonstrated in Fig. 8. Despite



(a) Random. (b) Random. (c) `WF`. (d) `WF`. (e) `WAFFLE`.
(f) `WAFFLE`. (g) `ATGF`. (h) `ATGF`. (i) `ATGF`. (j) `ATGF`.

Figure 8: Triggers in different watermarking schemes.

the privileges of Uchida, it is also the scheme that is the easiest to spoil, which costs approximately 21ms. Meanwhile, white-box watermark schemes require the author to obtain and transmit the entire pirated DNN model to the verification community for proof. In contrast, the black-box setting where the author only has to publicize the link to 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` (when $K \leq 300$), under which a high level of security is achieved.

Conclusion

This paper presents `Merkle-Sign`, a protocol for DNN model protection in FL by watermarking. After formulating the security requirements, we combine the watermarking schemes 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 `ATGF` that generates triggers robust against the spoil attack. Experimental results indicated that `Merkle-Sign` can provide the desired security in FL. Moreover, extra requirements introduced by `Merkle-Sign` point the direction of designing new watermarking schemes.

`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., watermarks that are robust under the aggregation operator so the embedding can be conducted in a completely decentralized manner.

References

Adi, Y.; Baum, C.; Cisse, M.; Pinkas, B.; and Keshet, J. 2018. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*, 1615–1631.

Atli, B. G.; Xia, Y.; Marchal, S.; and Asokan, N. 2020. WAFFLE: Watermarking in Federated Learning. *CoRR*, abs/2008.07298.

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

Darvish, R. B.; Chen, H.; and Koushanfar, F. 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.

Deng, L. 2012. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6): 141–142.

Guan, X.; Feng, H.; Zhang, W.; Zhou, H.; Zhang, J.; and Yu, N. 2020. Reversible Watermarking in Deep Convolutional Neural Networks for Integrity Authentication. In *Proceedings of the 28th ACM International Conference on Multimedia*, 2273–2280.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images.

Le Merrer, E.; Perez, P.; and Trédan, G. 2020. Adversarial frontier stitching for remote neural network watermarking. *Neural Computing and Applications*, 32(13): 9233–9244.

Li, F.; and Wang, S. 2021. Secure Watermark for Deep Neural Networks with Multi-task Learning. *arXiv preprint arXiv:2103.10021*.

Li, F.; Wang, S.; and Alan, W.-C. L. 2021. Regulating Ownership Verification for Deep Neural Networks: Scenarios, Protocols, and Prospects. *IJCAI Workshop*.

Li, H.; Lu, R.; Zhou, L.; Yang, B.; and Shen, X. 2013. An efficient merkle-tree-based authentication scheme for smart grid. *IEEE Systems Journal*, 8(2): 655–663.

Li, H.; Willson, E.; Zheng, H.; and Zhao, B. Y. 2019. Persistent and unforgeable watermarks for deep neural networks. *arXiv preprint arXiv:1910.01226*.

Li, T.; Sahu, A. K.; Talwalkar, A.; and Smith, V. 2020. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3): 50–60.

Liu, H.; Weng, Z.; and Zhu, Y. 2021. Watermarking Deep Neural Networks with Greedy Residuals. In *International Conference on Machine Learning*, 6978–6988. PMLR.

Mengelkamp, E.; Notheisen, B.; Beer, C.; Dauer, D.; and Weinhardt, C. 2018. A blockchain-based smart grid: towards sustainable local energy markets. *Computer Science-Research and Development*, 33(1): 207–214.

Uchida, Y.; Nagai, Y.; Sakazawa, S.; and Satoh, S. 2017. Embedding watermarks into deep neural networks. In *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval*, 269–277.

Wang, T.; and Kerschbaum, F. 2019. Robust and Undetectable White-Box Watermarks for Deep Neural Networks. *arXiv preprint arXiv:1910.14268*.

Wei, K.; Li, J.; Ding, M.; Ma, C.; Yang, H. H.; Farokhi, F.; Jin, S.; Quek, T. Q.; and Poor, H. V. 2020. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security*, 15: 3454–3469.

Xiao, H.; Rasul, K.; and Vollgraf, R. 2017. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. .

Xu, G.; Li, H.; Zhang, Y.; Lin, X.; Deng, R. H.; and Shen, X. 2020. A Deep Learning Framework Supporting Model Ownership Protection and Traitor Tracing. In *2020 IEEE 26th International Conference on Parallel and Distributed Systems (ICPADS)*, 438–446.

Yang, Q.; Liu, Y.; Chen, T.; and Tong, Y. 2019. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2): 1–19.

Zhang, J.; Gu, Z.; Jang, J.; Wu, H.; Stoecklin, M. P.; Huang, H.; and Molloy, I. 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.

Zhu, R.; Zhang, X.; Shi, M.; and Tang, Z. 2020. Secure neural network watermarking protocol against forging attack. *EURASIP Journal on Image and Video Processing*, 2020(1): 1–12.

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 $\mathcal{A}$'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_{root} under $\texttt{hash}_2$ to forge a legal key. Hence a PPT adversary $\mathcal{A}_{\text{pretend}}$ that suceeds in pretending to be an uninformed author participating FL can be used to build a PPT adversary $\mathcal{A}_{\text{collide}}$ that finds a collision w.r.t. $\texttt{hash}_2$. Formally, $\mathcal{A}_{\text{collide}}$ operates as Algo. 3. The event that $\mathcal{A}_{\text{collide}}$

Algorithm 3: PPT $\mathcal{A}_{\text{collide}}$ that finds a collsion for $\texttt{hash}_2$.

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

Ensure: x and y such $\texttt{hash}_2(y) = \texttt{hash}_2(x)$.

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 runing $\mathcal{A}_{\text{pretend}}$.
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_{root}, especially nodes of the second level $\tilde{T}_0$ and $\tilde{T}_1$.
6: $\mathcal{A}_{\text{collide}}$ returns $T_0\|T_1$ and $\tilde{T}_0\|\tilde{T}_1$.

succeeds in finding a collision pair $x \neq y$ happens if: $\mathcal{A}_{\text{pretend}}$ succeeds in pretending to be an author and $T_0\|T_1 \neq \tilde{T}_0\|\tilde{T}_1$. Therefore:

$$\Pr([\mathcal{A}_{\text{collide}} \text{ wins}]) \geq \Pr([\mathcal{A}_{\text{pretent}} \text{ wins}] \wedge [T_0\|T_1 \neq \tilde{T}_0\|\tilde{T}_1]).$$

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

$$\Pr([\mathcal{A}_{\text{pretent}} \text{ wins}] \wedge [T_0\|T_1 = \tilde{T}_0\|\tilde{T}_1])$$
$$= \Pr([\mathcal{A}_{\text{pretent}} \text{ wins}]) - \Pr([\mathcal{A}_{\text{pretent}} \text{ wins}] \wedge [T_0\|T_1 \neq \tilde{T}_0\|\tilde{T}_1])$$

is non-negligible given the assumption that $\Pr([\mathcal{A}_{\text{pretent}} \text{ wins}])$ is non-negligible. This further implies that $\mathcal{A}_{\text{pretend}}$ can efficiently inverse $\texttt{hash}_2$ with non-negligible probability, which is contradictive to the fact that $\texttt{hash}_2$ is also preimage resistant [1]. Therefore such $\mathcal{A}_{\text{pretend}}$ does not exist.

[1] Collision resistance implies preimage resistance.

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

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

Detailed implementations for experiments

`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.

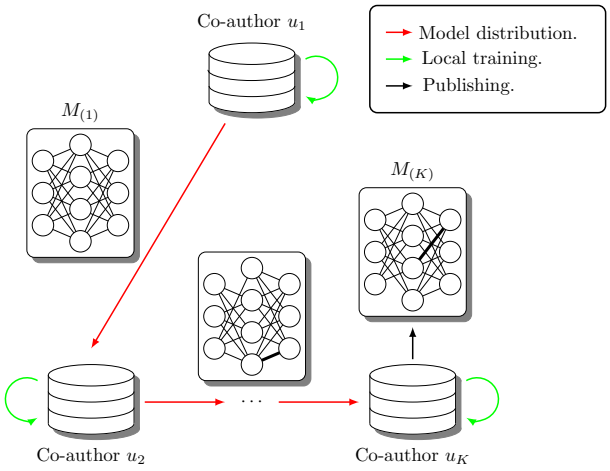


Figure 9: The architectures of P2P FL.

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

We assume that the watermarking scheme only modifies a tiny part of the model (e.g, few parameters or few trigger samples) so such embedding is not going to prevent the model from convergence. The capacity of the model w.r.t. the watermarking scheme 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 $\mathcal{U}$ follow Algo. 5. The correctness is guaranteed by the following theorem.

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

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

Algorithm 4: `Merkle-Sign` for P2P FL.

Require: A one-time watermarking scheme `WM`, security parameters N, L, functions $\texttt{hash}_1$ and $\texttt{hash}_2$.
Ensure: A watermarked model $M_{\mathcal{A}}$ and evidence for verification.

1: Each $u_k \in \mathcal{U}$ generates $\texttt{key}_k \leftarrow \texttt{Gen}(1^N)$.
2: $\mathcal{U}$ forms a schedule $(u_{(1)}, u_{(2)}, \cdots, u_{(T)})$.
3: $u_{(0)}$ initializes a clean model.
4: **for** $t = 0$ to $T-1$ **do**
5: $u_{(t)}$ generates time-dependent $\texttt{key}^{\dagger}_{(t)} \leftarrow \texttt{Gen}(1^N)$.
6: $u_{(t)}$ tunes the model, embeds $\texttt{KEYs}_{(t)} = \left\{\texttt{key}_{(t)}, \texttt{key}^{\dagger}_{(t)}\right\}$ into the model, obtains $\texttt{VERs}_{(t)} = \left\{\texttt{verify}_{(t)}, \texttt{verify}^{\dagger}_{(t)}\right\}$.
7: $u_{(t)}$ signs and broadcasts the following message to the verification community.

$$\langle \texttt{time} \| \texttt{Merkle}(\texttt{KEYs}_{(t)}, \texttt{VERs}_{(t)}, \texttt{info}) \rangle.$$

8: $u_{(t)}$ transmits the model to the $u_{(t+1)}$.
9: **end for**
10: $u_{(T)}$ tunes and embeds $\mathcal{K}_{(T)}$ into the model, obtains $\left\{\texttt{verify}_{(T)}\right\}$.
11: Each $u_k \in \mathcal{U} / \left\{u_{(T)}\right\}$ sends $\left\{\texttt{hash}_1(\texttt{key}_k)\right\}$ and $\left\{\texttt{hash}_1(\texttt{verify}_k)\right\}$ to $u_{(T)}$.
12: $u_{(T)}$ builds a Merkle-tree from the information received, signs and broadcasts:

$$\langle \texttt{time} \| \texttt{Merkle}(\texttt{KEYs}, \texttt{VERs}, \texttt{info}) \rangle.$$

13: $u_{(T)}$ publishes the model.

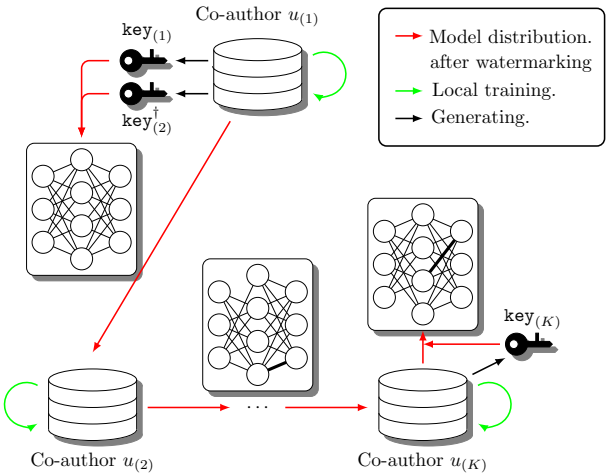


Figure 10: The `Merkle-Sign` watermarking framework for P2P FL.

Algorithm 5: Traitor-tracing under `Merkle-Sign` for P2P FL.

Require: The pirated model M.
Ensure: The index of the traitor.

1: **for** $t = 1$ to T **do**
2: $u_{(t)}$ proves its ownership over M.
3: $u_{(t)}$ presents to its co-authors: $\texttt{key}^{\dagger}_{(t)}$, $\texttt{verify}^{\dagger}_{(t)}$, and $\left\{\texttt{hash}_1(\texttt{key}_{(t)}), \texttt{hash}_1(\texttt{verify}_{(t)})\right\}$.
4: Other authors check whether the presented information is consistent with $u_{(t)}$'s historical broadcasting.
5: Other authors vote on $\texttt{verify}^{\dagger}_{(t)}\left(M, \texttt{key}^{\dagger}_{(t)}\right)$.
6: If the voted result is zero then return $(t-1)$.
7: **end for**
8: Return (T).

the model released by $u_{(t)}$ to $u_{(t+1)}$, the model in $u_{(t-1)}$'s perspective does not contain $\texttt{key}^{\dagger}_{(t)}$. The secrecy of $\texttt{key}^{\dagger}_{(t)}$ 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 $\texttt{key}^{\dagger}_{(t)}$, which happens only with negligible probability. By the definition of the correctness of `WM`, $u_{(t-1)}$ cannot forge the evidence for $\texttt{key}^{\dagger}_{(t)}$. Therefore, Algo. 5 is sufficient to identify $u_{(t-1)}$ as the traitor.