SCIENCE CHINA
Information Sciences

• Supplementary File •

TraceDFL: Towards Practical Model Confirmation and Traitor Tracking for Decentralized Federated Learning

Shuai Wang[1], Youliang Tian[2*], Jinbo Xiong[3*], Axin Xiang[2*], Renwan Bi[4] & Jianfeng Ma[5]

[1]*the State Key Laboratory of Public Big Data, College of Computer Science and Technology, Guizhou University, Guiyang,* 550025*, China*
[2]*College of Big Data and Information Engineering, Guizhou University, Guiyang* 550025*, China.*
[3]*School of Computer Science and Mathematics, Fujian University of Technology,* 350118*, Fuzhou , China.*
[4]*School of Cyber Science and Engineering, Minjiang University, Fuzhou* 350108*, China.*
[5]*School of Cyber Engineering, Xidian University,* 710071*, Xi'an, China.*

Appendix A Preliminaries

Appendix A.1 Model Watermark

We abstract the federated model watermarking algorithm as a triple $(Gen(), Embed(), Verify())$. $Gen(1^{\lambda}) \rightarrow mk$: Generates a watermarking key mk that uniquely represents the client's identity. $Embed(\mathbf{w}, mk) \rightarrow \mathbf{w}_{wm}$: Embeds the watermarking key mk into the federated model $\mathbf{w}$, producing the watermarked model $\mathbf{w}_{wm}$. $Verify(\mathbf{w}, \hat{\mathbf{w}}_{wm}, mk) \rightarrow 0/1$: Extracts the watermark information from the suspicious model $\hat{\mathbf{w}}_{wm}$ and compares it with mk. If the similarity falls below the predefined thresholds ϵ_K and ϵ_T, the algorithm outputs a verification result of 0 or 1. Below, we define several key concepts related to the federated model watermarking algorithm.

- **Validity**: A watermarking algorithm is considered validity if, when embedding a watermark using either black-box or white-box methods, the extracted watermark $\hat{mk}$ from a suspicious model $\hat{\mathbf{w}}_{wm}$ satisfies the following conditions:

$$\begin{cases} d(mk, \hat{mk}) \leqslant \epsilon_K \\ d(mk, \hat{mk}) \leqslant \epsilon_T, \end{cases}$$

  where $d()$ denotes the similarity measurement function, and ϵ_T, ϵ_K are predefined accuracy thresholds for black-box and white-box verification, respectively.

- **Fidelity**: A watermarking algorithm is considered fidelity if the classification accuracy of the watermarked model $\mathbf{w}_{wm}$ is equivalent to that of the original unwatermarked model $\mathbf{w}$, i.e.,

$$\begin{aligned} &\Pr_{x \in \mathcal{D}}\big[\mathbb{N}(x, \mathbf{w}) = y\big] \\ &\approx \Pr_{x \in \mathcal{D}}\big[\mathbb{N}(x, \hat{\mathbf{w}}_{wm}) = y\big]. \end{aligned}$$

  where $\mathbb{N}()$ denotes the model inference API interface.

- **Robustness**: A federated model watermarking algorithm is considered robust if the extracted watermark can still successfully verify model ownership after the watermarked model undergoes attacks such as fine-tuning, pruning, or overwriting. Formally, let $\mathbf{w}'_{wm}$ be the attacked version of the watermarked model $\mathbf{w}$, the algorithm satisfies robustness if:

$$\begin{aligned} &\Pr_{x \in \mathcal{D}}\big[\mathbb{N}(x, \mathbf{w}) = y\big] \\ &\approx \Pr_{x \in \mathcal{D}}\big[\mathbb{N}(x, \mathbf{w}'_{wm}) = y\big]. \end{aligned}$$

  and $Verify(mk, \mathbf{w}, \mathbf{w}'_{wm}) = 1$.

- **Unforgeability**: A watermarking algorithm is considered unforgeable if an adversary cannot claim ownership of a federated model by arbitrarily generating a watermark key mk' and embedding it into the model. Informally, if there exists a probabilistic polynomial time (PPT) algorithm $\mathcal{A}$ that can win the following game with probability at most ϵ, where ϵ is a negligible value, then the watermarking algorithm satisfies unforgeability.

$$\Pr\left[Verify(mk', \mathbf{w}, \mathbf{w}'_{wm}) = 1\right] \leqslant \epsilon,$$

* Corresponding author (email: youliangtian@163.com, jbxiong@fjut.edu.cn, axxiang@gzu.edu.cn)

- **Security**: A watermarking algorithm is considered secure if an adversary cannot obtain any meaningful information about the embedded watermark key mk from the publicly available parameters of the algorithm or the watermarked model $\mathbf{w}_{wm}$ itself.

- **Exclusivity**: In the DFL framework, no single participant in the federated training process can unilaterally claim exclusive ownership of the watermarked model $\hat{\mathbf{w}}_{wm}$. Formally, a watermarking algorithm satisfies exclusivity if, for any malicious internal client executing a PPT algorithm $\mathcal{A}$, the probability of successfully claiming exclusive ownership of the model is at most a negligible value ϵ:

$$\Pr\left[Verify(mk_{\text{mal}}, \mathbf{w}, \hat{\mathbf{w}}_{wm}) = 1\right] \leqslant \epsilon,$$

where mk_{mal} denotes the identity watermark of the malicious client.

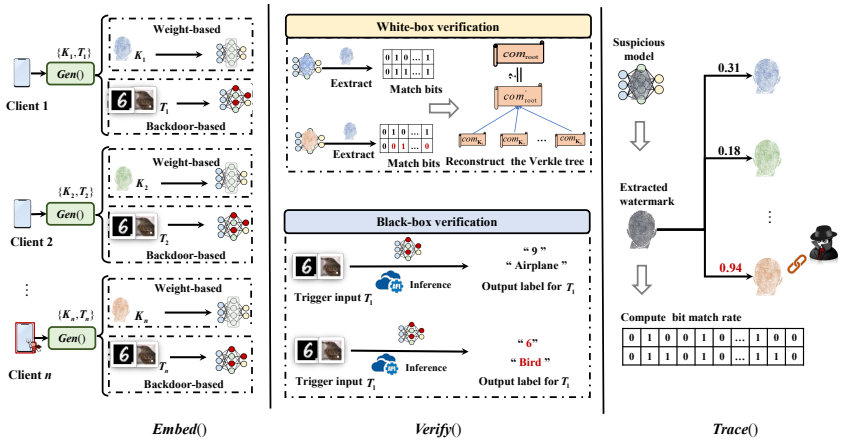


Fig. A1. The illustration of TraceDFL.

Appendix B Problem Formulation

Appendix B.1 Overview of TraceDFL

The main workflow of the TraceDFL framework is illustrated in Fig. A1. After local training, each client generates the identity watermark and initializes the backdoor trigger set, embedding the watermark into the local model. TraceDFL enables model ownership confirmation and traitor tracing through four key phases, watermark and key generation *Gen*(), embedding watermarks *Embed*(), ownership confirmation *Verify*(), and traitor tracking *Trace*().

- **Watermark and key generation** *Gen*(). Each client u locally generates a unique identity watermark $\mathbf{K}_u$ and receives a backdoor trigger set $\mathbf{T}_u$ distributed by the trusted authority.

- **Embedding watermarks** *Embed*(). Each client u embeds its identity watermark $\mathbf{K}_u$ into the regularization layer of the federated model using a weight-based embedding approach, and incorporates the trigger set $\mathbf{T}_u$ via backdoor embedding to generate a backdoored model.

- **Ownership confirmation** *Verify*(). During the verification phase, each client performs white-box verification by computing the bitwise matching rate of the identity watermark and verifying the Verkle tree root commitment to confirm model ownership. Additionally, black-box verification is conducted by inputting trigger samples and checking whether the model returns the expected backdoor labels.

- **Traitor tracking** *Trace*(). Each client u computes the matching rate between the extracted watermark and its identity watermark, and publishes the result to the bulletin board. The trusted authority then traces the malicious forwarder by identifying the client with the highest matching rate.

For clarity of scope, TraceDFL is formulated under the assumption that, in each training round, a subset of clients with stable device-to-device (D2D) communication links is randomly selected to participate. These clients are assumed to remain online and connected throughout the execution of the four phases, ensuring that the distributed backdoor trigger sets are consistently generated for the actual participants. In practice, only a moderate number of clients are required per round, as long as the D2D connectivity within the selected subset can support trigger distribution. While more realistic DFL scenarios may involve intermittent connectivity, client drop-in/drop-out, or time-varying topologies, we defer the design of adaptive mechanisms (e.g., dynamic trigger reallocation or topology-aware watermark embedding) to future work. This assumption of stable subsets allows us to focus on demonstrating the feasibility and effectiveness of decentralized model ownership confirmation and traitor tracing.

It is worth noting that although the white-box watermark already provides both ownership confirmation and traitor tracing, its verification requires parameter-level access, which is not always available in practice. The black-box watermark, by contrast, can be verified through simple API queries, making it suitable for ownership confirmation in scenarios where only black-box access to the suspicious model is possible. Therefore, TraceDFL incorporates both mechanisms: the white-box watermark serves as the primary

means for traitor tracing, while the black-box watermark complements ownership verification under restricted-access settings. This dual deployment strengthens the robustness of model ownership protection in decentralized federated learning. The important variables used in the paper and their interpretations are listed in Table B1.

Table B1 Notations and Descriptions

Notations	Descriptions	Notations	Descriptions
$\mathbf{w}$	Unwatermarked federated model	$\mathbf{w}_{wm}$	Watermarked federated model
$\hat{\mathbf{w}}_{wm}$	Suspicious models with watermarks	mk	Watermark Key
ϵ_K	Preset minimum acceptable matching rates for white-box watermarks	ϵ_T	Preset minimum acceptable matching rates for black-box watermarks
N	Length of the identity watermark bit-string for each client	μ	Margin hyperparameter in the hinge-like embedding loss for the local identity watermark
η	Steepness parameter for estimating the confidence of bitwise matching.	ξ	Activation threshold for matched watermark bits and unmatched watermark bits
$\mathcal{D}_u$	Client u's local dataset	$\mathbf{T}_u$	Client u's local trigger set
$\mathbf{K}_u$	Client u's unique identity key	$com_{\mathbf{K}_u}$	Client identity key $\mathbf{K}_u$'s commitment
$com_{\mathbf{w}_G}$	The commitment of the global model $\mathbf{w}_G$	$\mathbf{w}_i$, $\mathbf{x}_i$	Client i's local model
τ_u^T	Actual match rate of trigger sets	τ_u^K	Actual match rate of identity keys
com_{root}	Root commitment of the Verkle tree	$\mathbb{G}$	Multiplicative cyclic group

Appendix B.2 Threat Model

In a DFL setting, we assume that $|\mathcal{U}| = n$ clients collaboratively train a federated model, with each client sharing co-ownership of the resulting model. Clients are assumed to be semi-honest, meaning they honestly participate in model training and watermark embedding, but may also attempt to infer other clients' private data or identity watermarks. Additionally, external adversaries may compromise clients to launch attacks such as watermark overwriting or watermark removal.

To bootstrap the system, a trusted authority is introduced during initialization to generate watermark keys and construct the Verkle tree root. In practice, such an authority may be instantiated by an independent and jointly recognized third-party (e.g., industry consortia or arbitration organizations). To further relax this trust assumption, blockchain infrastructures can be leveraged as a decentralized alternative, whereby watermark key commitments are registered via smart contracts and the underlying consensus protocol ensures immutability and auditability [25]. In addition, distributed key generation (DKG) techniques [27–29] allow multiple clients to collaboratively derive the watermark keys without reliance on a single entity. We stress, however, that the concrete realization of blockchain-based deployment or DKG protocols falls beyond the scope of this work; we only note that these well-established primitives can be seamlessly integrated to alleviate the reliance on a single trusted authority in practice.

A particularly critical challenge arises from the mutual distrust among participants in DFL. Internal clients who obtain the identity watermarks of others may remove them from the model and selfishly claim exclusive ownership, violating the co-authorship rights of honest clients. We refer to this selfish behavior as a Model Exclusive Attack (MEA). In such attacks, a malicious client falsely claims the model as solely its own and may resell it, resulting in model misuse. As described in Fig. A1, a selfish client could erase the watermarks of others, submit only its watermark key mk_{mal}, and pass the ownership verification algorithm *Verify*(), thereby falsely asserting sole ownership of the model.

Therefore, a watermarking scheme suitable for DFL must be designed to accommodate the decentralized, independent, and mutually untrusted nature of participating entities. It must ensure that each client can independently verify model ownership, while also preventing any single party from exclusively claiming ownership of the collaboratively trained model.

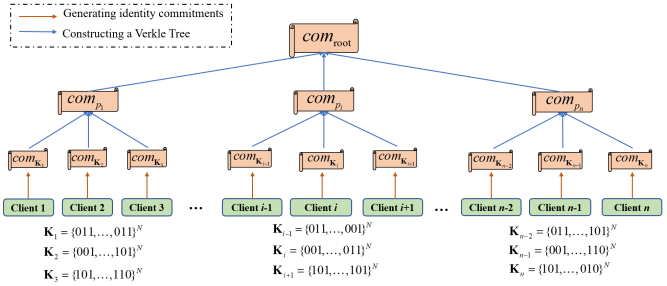


Fig. B1. 3-ary Verkle Tree Construction.

Appendix C Implementation of TraceDFL

In this section, we comprehensively portray how to implement model confirmation and traitor tracing in the DFL architecture. Concretely, the TraceDFL is defined formally as a tuple (*Gen*, *Embed*, *Verify*, *Trace*).

Appendix C.1 Watermark and Key Generation

For simplicity, we utilize a trusted authority to bootstrap the construction of watermark keys and Verkle tree roots. Once construction is complete, the trusted authority is taken offline. The watermark and key generation process can be abstracted as $Gen(1^{\lambda}) \rightarrow (\mathbf{T}_u, \mathbf{K}_u)$, where $\mathbf{T}_u = \left\{(x^1_{\mathbf{T}_u}, y^1_{\mathbf{T}_u}), \cdots, (x^n_{\mathbf{T}_u}, y^n_{\mathbf{T}_u})\right\}$ is a set of triggers based on a distributed backdoor, and $\mathbf{K}_u = \text{sign}(\mathbf{A}_u \mathbf{w}^{\gamma})$ is the unique key of client u. The detailed watermarks and keys generation are shown in Algorithm C1.

Inspired by distributed backdoor attacks [30], the trusted authority decomposes the trigger $\mathbf{T}$ into sub-triggers $|\mathcal{U}|$ and broadcasts them to the corresponding clients after associating them with client IDs, each client $u \in \mathcal{U}$ utilizes the local trigger $\mathbf{T}_u$ to embed a backdoor watermark to the local model during the training phase, and all local triggers are aggregated into global triggers $\mathbf{T}$ in the aggregation phase. After trigger segmentation is complete, the trusted authority will go offline. Note that the trigger $\mathbf{T}$ can denote regular white rectangular shapes, meaningful fonts, or logos, e.g., Google's logos/fonts. In the implementation, we set the trigger $\mathbf{T}$ to be a rectangular pattern of white pixels.

In the unique key $\mathbf{K}_u$ of client u, the trusted authority randomly generates the key matrix $\mathbf{A}_u \in \mathbb{R}^{M \times N}$ to satisfy the standard Gaussian distribution, performs matrix multiplication between the secret matrix $\mathbf{A}_u$ and the weights of the model layer to be embedded $\mathbf{w}^{\gamma}$, and obtains the unique bit string $\mathbf{K}_u \in \{0,1\}^N$ of the client u after the sign function sign(). In the implementation, we specify the normalization layer of global model as the model layer to be embedded.

Algorithm C1 Watermark and Key Generation *Gen*()

Require: Embedding-layer weight size M of $\mathbf{w}^{\gamma}$; local data $\{\mathcal{D}_u\}_{u=1}^{\mathcal{U}}$; unique key length N; number of clients $\mathcal{U}$.
Ensure: Distributed backdoor trigger set $\{\mathbf{T}_u\}_{u=1}^{\mathcal{U}}$; client unique key set $\{\mathbf{K}_u\}_{u=1}^{\mathcal{U}}$.
1: Initialize trigger $\mathbf{T}$ as a white rectangular patch on 10% of training samples $\mathbf{X}$.
2: **for** $u = 1$ **to** $\mathcal{U}$ **do**
3: Generate local trigger set $\mathbf{T}_u$ according to the location mapped from client ID u;
4: $\mathbf{A}_u = \text{randn}(M, N)$; // Gaussian sampling
5: $\mathbf{K}_u = \text{sign}(\mathbf{A}_u\, \mathbf{w}^{\gamma})$;
6: **end for**
7: **return** $\{\mathbf{T}_u\}_{u=1}^{\mathcal{U}}$, $\{\mathbf{K}_u\}_{u=1}^{\mathcal{U}}$.

Appendix C.2 Inner Weight-based Watermark Embedding and Verification

To embed identity information into the local model, each client $u \in \mathcal{U}$ incorporates a weight watermark $\mathbf{K}_u$, representing its identity, into the parameters $\mathbf{w}^{\gamma}$ of the regularization layer. The weight regularization term is computed as follows:

$$\begin{aligned} \mathcal{L}_{\mathbf{K}_u}(\mathbf{w}^{\gamma}) &= \mathcal{L}_{\mathbf{K}_u}(\mathbf{K}_u, \hat{\mathbf{K}}_u) \\ &= \sum_{i=1}^{N} \max(\mu - t_i e_i, 0). \end{aligned} \tag{C1}$$

where $\hat{\mathbf{K}}_u = \mathbf{w}^{\gamma}\mathbf{A}_u = (e_1, \cdots, e_N) \in \{0,1\}^N$ denotes the watermark extracted from the local update, $\mathbf{K}_u = (t_1, \cdots, t_N) \in \{0,1\}^N$ represents the target watermark embedded into the local model by client u, and μ is the hyperparameter of the hinge-like loss function. We instantiate the embedding process of the weight watermark using a hinge-like loss function, iteratively minimizing the loss between the target watermark $\hat{\mathbf{K}}_u$ and the extracted watermark $\mathbf{K}_u$. The hinge-like error emphasizes boundary separation characteristics, making it particularly suitable for binary decision tasks (similar/dissimilar). It is widely used to optimize the embedding process of weight watermarks [14, 15].

In TraceDFL, a Verkle tree is constructed to enable each client $u \in \mathcal{U}$ to publicly and independently verify model ownership (or co-authorship) locally. The commitment representation of each client u's unique identity $\mathbf{K}_u$ serves as a leaf node of the Verkle tree, while the root hash of the tree represents the unique identifier of the federated model $\mathbf{w}_{wm}$ collaboratively trained by the client set $\mathcal{U}$. During the verification phase, each client u downloads the identity commitments of other clients from a public bulletin board and locally reconstructs the Verkle tree to verify the consistency of the root commitment. After receiving the set of public commitments $\{com_{\mathbf{K}_v}\}_{v \in \mathcal{U}}$ from the bulletin board, each client u reconstructs the Verkle tree deterministically, the identity commitments are inserted into fixed leaf positions, and all internal node commitments are recomputed by applying the vector-commitment aggregator to their children. Since vector commitments are binding and deterministic, the root commitment $com^{(u)}_{\text{root}}$ computed locally by client u must match the globally published com_{root} if and only if all identity commitments are valid and untampered. Through this procedure, TraceDFL enables every client to independently reproduce the Verkle commitment structure without needing a server or trusted intermediary. The information published on the bulletin board, along with the unique identifier of the federated model $\mathbf{w}_{wm}$, is maintained by a trusted authority. In the event of disputes over model ownership, clients can declare co-authorship to the trusted authority to prevent model misuse. As a highly reliable authority, the trusted authority also enhances the credibility of ownership verification.

Each client $u \in \mathcal{U}$ computes a commitment $com_{\mathbf{K}_u} = Commit(\mathbf{K}_u; r)$ of its identity information $\mathbf{K}_u$ and publishes the commitment on a bulletin board, preserving the privacy of $\mathbf{K}_u$. Based on the commitments on the bulletin board, the trusted authority constructs a Verkle tree and generates a root commitment com_{root}, which represents the unique identifier of the federated model $\mathbf{w}_{wm}$. The node information in the Verkle tree captures the relationships between the identity information of the participating training clients. A Verkle

Algorithm C2 Embedding Watermarks *Embed*()

Require: Local data $\{\mathcal{D}_u\}_{u=1}^{\mathcal{U}}$; distributed backdoor trigger sets $\{\mathbf{T}_u\}_{u=1}^{\mathcal{U}}$; unique identities $\{\mathbf{K}_u\}_{u=1}^{\mathcal{U}}$; learning rate λ.
Ensure: Model with embedded watermark $\mathbf{w}_{wm}$.
Construct a Verkle Tree:
for $u = 1$ to $\mathcal{U}$ **do**
  Compute commitment $com_{\mathbf{K}_u} = \text{Commit}(\mathbf{K}_u; r_u)$ and publish $com_{\mathbf{K}_u}$ to the bulletin board;
  The trusted authority constructs the Verkle tree for the set of local training clients.
end for
Local Training:
for $u = 1$ to $\mathcal{U}$ **do**
  Sample a mini-batch $\mathbf{X}_u = \{x_u^1, \ldots, x_u^t\}$ and labels $\mathbf{Y}_u = \{y_u^1, \ldots, y_u^t\}$ from $\mathcal{D}_u$;
  if embed distributed backdoor watermarks **then**
    Augment with triggers: $\mathbf{X}_u = \mathbf{X}_u \cup \{x_{\mathbf{T}_u}^1, \ldots, x_{\mathbf{T}_u}^n\}$, $\mathbf{Y}_u = \mathbf{Y}_u \cup \{y_{\mathbf{T}_u}^1, \ldots, y_{\mathbf{T}_u}^n\}$;
  end if
  Compute cross-entropy loss $\mathcal{L}_m$ on $(\mathbf{X}_u, \mathbf{Y}_u)$, $\mathcal{L}_m = \mathcal{L}_{\text{master}} + \mathcal{L}_{\mathbf{T}_u}$;
  Compute hinge-like loss $\mathcal{L}_{\mathbf{K}_u}$ using identity $\mathbf{K}_u$, $\mathcal{L}_e \Leftarrow \mathcal{L}_m + \mathcal{L}_{\mathbf{K}_u}$;
  Update local model: $\mathbf{w}_u = \mathbf{w}_u - \lambda \nabla \mathcal{L}_e(\mathbf{w}_u)$;
  Compute and broadcast model commitment: $com_{\mathbf{w}_u} = \text{Commit}(\mathbf{w}_u; r_u)$;
end for
return $\mathbf{w}_{wm}$.

tree is a vector-commitment-based authenticated data structure that generalizes Merkle trees by replacing hash nodes with polynomial or Pedersen-style commitments. Each internal node commits to a large vector of child values, enabling shorter membership proofs: a verifier only needs the commitment opening of the selected child rather than all sibling hashes. This reduces proof size from $\mathcal{O}(\log n)$ in Merkle trees to $\mathcal{O}(\log_k n)$, making Verkle trees significantly more bandwidth-efficient for decentralized verification. Fig.B1 illustrates the construction process of a 3-ary Verkle tree.

Unlike server-centric aggregation mechanisms, TraceDFL must achieve global model synchronization in a fully decentralized environment without relying on trusted coordinators. Inspired by VerifyDFL [31], once local training and watermark embedding are completed, each client u computes a commitment $com_{\mathbf{w}_u}$ to its updated local model $\mathbf{w}_u$ and posts this value to a publicly accessible bulletin board together with its identifier. The bulletin board enforces accountability by binding each commitment to the client identity.

Due to the additive homomorphic property of the underlying vector commitment scheme [32], the commitments published by all participants can be multiplicatively combined to obtain a commitment to the aggregated model, i.e.,

$$com_{\mathbf{w}_G} = \prod_{u \in \mathcal{U}} com_{\mathbf{w}_u} = \mathbf{g}^{\sum_{u \in \mathcal{U}} \mathbf{w}_u} h^{\sum_{u \in \mathcal{U}} r_u}, \tag{C2}$$

where $com_{\mathbf{w}_G}$ denotes the global model commitment and r_u is the client-specific randomness. Each client can independently retrieve all commitments from the bulletin board, compute $com_{\mathbf{w}_G}$ locally, and thus reach consensus on the global commitment without iterative gossip communication. In the subsequent opening phase, every client reveals its randomness r_u, which enables the group to jointly reconstruct and verify the global model.

Remarks. Merkle-sign [25] incorporates Merkle-tree structures into federated watermarking to enable ownership verification among multiple participants. While Merkle-Sign demonstrates how cryptographic primitives can strengthen model protection in FL, it fundamentally assumes a server-centric training architecture where the aggregator embeds and distributes watermarks, limiting its applicability in decentralized settings. In addition, Merkle proofs grow logarithmically with the number of participants, creating scalability bottlenecks. In contrast, TraceDFL adopts Verkle trees, where internal nodes commit to large vectors rather than pairwise hashes, enabling shorter membership proofs and significantly reduced verification bandwidth. Each client locally reconstructs the Verkle tree from publicly posted identity commitments and recomputes the root commitment. Because vector commitments are deterministic and binding, any tampering or exclusion of other clients? identities results in a mismatched root, enabling clients to detect inconsistencies without server involvement. Moreover, this authenticated structure fundamentally prevents MEA. A malicious client cannot remove other participants? identities without breaking the Verkle-commitment binding, and honest clients can always open their Verkle-paths to cryptographically prove inclusion in the collaboratively trained model. These membership proofs, requiring only $\mathcal{O}(\log_k n)$ commitment openings, enable efficient and public co-authorship verification, ensuring that no participant can falsely claim sole ownership of the global model.

Appendix C.3 Distributed Backdoor-Based Watermark Embedding and Verification

During the local training phase, each client $u \in \mathcal{U}$ embeds the corresponding local trigger $\mathbf{T}_u$ into its local model $\mathbf{w}_u$ and computes the cross-entropy loss $\mathcal{L}_{\mathbf{T}_u}$.

$$\mathcal{L}_{\mathbf{T}_u} = l(y_{\mathbf{T}_u}, \mathbf{w}_u(x_{\mathbf{T}_u})). \tag{C3}$$

where $l()$ is the cross-entropy loss function. This process adheres to the training protocol for distributed backdoor tasks [30], wherein all participating clients strive to enhance the performance of the backdoor task without compromising the primary task's efficacy. Specifically, each client $u \in \mathcal{U}$ trains its local model $\mathbf{w}_u$ to maximize the probability of predicting the target label on data samples

containing the trigger, while simultaneously maximizing the probability of predicting the true label on clean data samples.

$$\mathbf{w}_u^* = \arg\max_{\mathbf{w}_u} \Bigg(\sum_{i \in \mathcal{D}_u^{\text{clean}}} \Pr(\mathbf{w}_G(x_u^i) = y_u^i) + \sum_{i \in \mathcal{D}_u^{\text{poison}}} \Pr(\mathbf{w}_G(x_{\mathbf{T}_u}^i) = y_{\mathbf{T}_u}^i) \Bigg). \tag{C4}$$

where $\mathbf{w}_u^*$ is the iteratively optimized local model, $Pr()$ denotes the probability, $\mathcal{D}_u^{\text{clean}}$ is the set of clean local data samples, and $\mathcal{D}_u^{\text{poison}}$ is the set of poisoned local data samples.

Appendix C.4 Ownership Confirmation and Traitor Tracking

When unauthorized forwarding, tampering, or forgery of federated learning-generated models occurs during data circulation or model trading, TraceDFL conducts ownership verification and traitor tracing through a dual-layer watermarking design. The internal weight watermark (white-box) enables fine-grained ownership confirmation and precise traitor identification when parameter-level access to the suspicious model is available, while the distributed backdoor watermark (black-box) supports lightweight verification through inference-only access in scenarios where only API-level interaction is possible. By integrating these two complementary mechanisms, TraceDFL provides a unified protection framework that remains effective across diverse deployment environments. It ensures no malicious actor can deploy models without authorisation, impersonate identities, or falsely claim exclusive ownership of collaboratively trained federated models. The specific process of ownership verification and traitor tracing is illustrated in Fig. A1.

Appendix C.4.1 *Ownership Confirmation*

In TraceDFL, the trusted authority invokes the watermark verification algorithm *Verify* to extract the watermarks in the suspicious model to confirm the ownership of the model. Each client $u \in \mathcal{U}$ extracts the embedded identity watermark $\hat{\mathbf{K}}_u$ from a suspicious model using a white-box approach. In TraceDFL, we discard the use of Hamming distance to evaluate the difference between the extracted suspicious watermark $\hat{\mathbf{K}}_u$ and the target watermark $\mathbf{K}_u$. This decision is made because Hamming distance can produce ambiguous results, where two different identity watermarks may have the same distance to the target watermark, i.e., $hamd(\mathbf{K}_u, \hat{\mathbf{K}}_1)$ and $hamd(\mathbf{K}_u, \hat{\mathbf{K}}_2)$, while $\hat{\mathbf{K}}_1 \neq \hat{\mathbf{K}}_2$, leading to potential misidentification. We model the identity watermark matching problem, assuming that the target watermark $\mathbf{K}_u$ and the extracted watermark $\hat{\mathbf{K}}_u$ are both bitstrings of length N. Let τ be the probability of each bit matching. For the i-th bit, we define the indicator function as follows:

$$X_i = \begin{cases} 1, & \text{if } \mathbf{K}_{u,i} = \hat{\mathbf{K}}_{u,i}, \\ 0, & \text{otherwise.} \end{cases} \tag{C5}$$

assuming that X_i follows an independent distribution for each bit, the likelihood function of the entire extracted watermark can be expressed as:

$$L(\tau) = \prod_{i=1}^{N} \tau^{X_i} (1-\tau)^{1-X_i} = \tau^m (1-\tau)^{N-m}, \tag{C6}$$

where $m = \sum_{i=1}^{N} X_i$ represents the number of matching bits. Compute the log-likelihood:

$$\log L(\tau) = m \log \tau + (N-m) \log(1-\tau). \tag{C7}$$

take the derivative with respect to τ and set $\tau = 0$ to obtain the maximum likelihood estimate $\hat{\tau} = \frac{m}{N}$. However, since $\hat{\tau}$ takes discrete values, it cannot clearly distinguish the bit-wise differences between the suspicious watermark $\hat{\mathbf{K}}_u$ and the target watermark $\mathbf{K}_u$. In TraceDFL, we precompute the confidence probability p_i for each bit of the extracted identity watermark $\mathbf{K}_u$.

$$p_i \leftarrow \frac{1}{1 + \exp(-\eta(\hat{\mathbf{K}}_{u,i} - \xi))}, \tag{C8}$$

then, based on the maximum likelihood estimation, we compute a continuous estimate of the overall matching rate.

$$\tau_u^K = \frac{1}{N} \sum_{i=1}^{N} \Big(p_i \cdot \mathbb{1}\{\mathbf{K}_{u,i} = 1\} + (1 - p_i) \cdot \mathbb{1}\{\mathbf{K}_{u,i} = 0\} \Big). \tag{C9}$$

the above steps soften the judgment of each bit, obtaining continuous values and overcoming the discreteness issue of Hamming distance.

For black-box verification, the trigger samples $\mathbf{X}_u$ are fed into the suspicious model $\hat{\mathbf{w}}_{wm}$ to obtain the predicted labels $\hat{\mathbf{Y}}_u$, and the probability $\Pr(\hat{\mathbf{Y}}_u = \mathbf{Y}_u)$ is evaluated to determine the trigger set matching rate τ_u^T. Importantly, when both the trigger set matching rate τ_u^T and the identity matching rate τ_u^K are higher than the predefined thresholds ϵ_T and ϵ_K, each client computes the commitment $com_{\hat{\mathbf{K}}_u}$ of the extracted watermark $\hat{\mathbf{K}}_u$ and uploads it to the public bulletin board. Simultaneously, clients download identity commitments from other clients $v \in \mathcal{U} \setminus u$ to construct a Verkle tree. By comparing the root commitment com'_{root} of the newly constructed Verkle tree, model ownership can be verified. The detailed calculation procedure is shown in Algorithm C3.

Algorithm C3 Ownership Confirmation *Verify*()

Require: Trigger sets $\{\mathbf{T}_u = (\mathbf{X}_u, \mathbf{Y}_u)\}_{u=1}^{\mathcal{U}}$; suspicious model $\hat{\mathbf{w}}_{wm}$ with embedding-layer weights $\hat{\mathbf{w}}^{\gamma}$; secret matrices $\{\mathbf{A}_u\}_{u=1}^{\mathcal{U}}$; client identities $\{\mathbf{K}_u\}_{u=1}^{\mathcal{U}}$; segmentation threshold ξ and steepness η; identity-match threshold ϵ_K; trigger-match threshold ϵ_T; key length N.
Ensure: **true** if ownership holds; otherwise **false**.

for $u = 1$ to $\mathcal{U}$ **do**
  Extract watermark bits: $\hat{\mathbf{K}}_u = \text{sign}(\mathbf{A}_u \hat{\mathbf{w}}^{\gamma})$;
  Compute bit-wise confidence scores $\{p_i\}_{i=1}^{N}$:
  for $i = 1$ to N **do**
    $p_i = \left(1 + \exp(-\eta(\hat{\mathbf{K}}_{u,i} - \xi))\right)^{-1}$;
  end for
  Compute MLE-based identity match rate: $\tau_u^K \leftarrow \frac{1}{N}\sum_{i=1}^{N}\left(p_i \cdot \mathbb{1}\{\mathbf{K}_{u,i} = 1\} + (1 - p_i) \cdot \mathbb{1}\{\mathbf{K}_{u,i} = 0\}\right)$;
  Query suspicious model on triggers: $\hat{\mathbf{Y}}_u \leftarrow \text{Pred}_{\hat{\mathbf{w}}_{\text{wm}}}(\mathbf{X}_u)$; Compute trigger-label match rate: $\tau_u^T \leftarrow \Pr(\hat{\mathbf{Y}}_u = \mathbf{Y}_u)$;
  if $\tau_u^K \geqslant \epsilon_K$ **and** $\tau_u^T \geqslant \epsilon_T$ **then**
    Compute and publish commitment: $com_{\hat{\mathbf{K}}_u} = \text{Commit}(\hat{\mathbf{K}}_u; r_u)$;
    Download other clients' commitments $v \in \mathcal{U} \setminus \{u\}$ and reconstruct Verkle tree to obtain root com'_{root};
    if $com'_{\text{root}} \neq com_{\text{root}}$ **then**
      return **false**;
    end if
  end if
end for
return **true**.

Appendix C.4.2 *Traitor Tracking*

Consistent with the identity watermark extraction process in ownership verification, each client calculates the identity watermark matching rate τ_u^K and uploads it to the public bulletin board. To trace malicious clients within the FL group who unauthorizedly forward or leak the model, the trusted authority collects the matching rates $\{\tau_u^K\}_{u=1}^{N}$ from all clients, forming a vector $\vec{\tau}$. By sorting this vector, the client with the highest identity watermark matching rate is identified, and its corresponding client-ID is returned.

In TraceDFL, each client embeds a unique identity watermark within the federated model. The distributed backdoor watermark is interpreted using maximum-likelihood estimation, which models watermark matching as a continuous probability measure and enhances the stability of traitor identification under black-box access. Meanwhile, the Verkle-based white-box verification anchors every client's identity through binding vector commitments, enabling cryptographically verifiable co-authorship and defending against MEA. In short, these two mechanisms form a complete verification pipeline. The black-box watermarking provides practical detectability of model leakage in untrusted environments, while the white-box commitment structure ensures authoritative ownership validation whenever internal model information is available. This layered design ensures that ownership confirmation and traitor tracing remain reliable across heterogeneous deployment conditions, and that no participant can erase, counterfeit, or monopolize model ownership during or after federated training.

Algorithm C4 Traitor Tracking *Trace*()

Require: Suspicious model $\hat{\mathbf{w}}_{wm}$ with embedding-layer weights $\hat{\mathbf{w}}^{\gamma}$; secret matrices $\{\mathbf{A}_u\}_{u=1}^{\mathcal{U}}$; client identities $\{\mathbf{K}_u\}_{u=1}^{\mathcal{U}}$; segmentation threshold ξ; steepness η; key length N; target Verkle-tree root commitment com_{root}.
Ensure: Malicious client identity v_{mal}.

for $u = 1$ to $\mathcal{U}$ **do**
  Extract watermark bits: $\hat{\mathbf{K}}_u = \text{sign}(\mathbf{A}_u \hat{\mathbf{w}}^{\gamma})$;
  Compute bit-wise confidence scores $\{p_i\}_{i=1}^{N}$: $p_i = \left(1 + \exp(-\eta(\hat{\mathbf{K}}_{u,i} - \xi))\right)^{-1}$;
  Compute MLE-based identity match rate: $\tau_u^K \leftarrow \frac{1}{N}\sum_{i=1}^{N}\left(p_i \cdot \mathbb{1}\{\mathbf{K}_{u,i} = 1\} + (1 - p_i) \cdot \mathbb{1}\{\mathbf{K}_{u,i} = 0\}\right)$;
  Publish τ_u^K to the bulletin board;
end for
Authority collects $\vec{\tau} = \{\tau_u^K\}_{u=1}^{\mathcal{U}}$ from the bulletin board;
Identify the most likely malicious client: $v_{\text{mal}} \Leftarrow \arg\max_{v \in \{1,\ldots,\mathcal{U}\}} \vec{\tau}$;
return v_{mal}.

Appendix D Theoretical Analysis

In this section, we theorize the security and effectiveness of TraceDFL at a high level, and we prove that TraceDFL satisfies the following properties:

Validity. For black-box verification, TraceDFL computes the matching rate τ_u^T between the model's output labels $\hat{\mathbf{Y}}_u$ and the target labels $\mathbf{Y}_u$. For white-box verification, it estimates the matching rate τ_u^K of the claimed identity watermark using maximum likelihood estimation. *Verify*() outputs true if $\tau_u^T \geqslant \epsilon_T$ and $\tau_u^K \geqslant \epsilon_K$, and the root commitment $com'_{\text{root}} = com_{\text{root}}$ of the reconstructed Verkle.

Fidelity. TraceDFL instantiates the concept of distributed backdoor attacks to embed backdoor watermarks. Prior work [30] has demonstrated that distributed backdoors can serve as effective watermarking mechanisms without degrading the performance of the

primary learning task, i.e.

$$\Pr_{x\in\mathcal{D}^{\text{clean}}}\big[\mathbb{N}(x,\mathbf{w})=y\big] \approx \Pr_{x\in\{\mathcal{D}^{\text{clean}}\cup\mathcal{D}^{\text{poison}}\}}\big[\mathbb{N}(x,\hat{\mathbf{w}}_{\text{wm}})=y\big].$$

Furthermore, prior work [9] has shown that deep neural network models embedded with white-box watermarks satisfy:

$$\Pr_{x\in\mathcal{D}}\big[y\neq\mathbb{N}(x,\mathbf{w})\big]\leqslant\epsilon,$$

where ϵ is a negligible value.

Robustness. Assume there exists a probabilistic polynomial-time (PPT) adversary $\mathcal{A}$ that attempts to compromise the watermark by applying common model perturbation techniques such as fine-tuning, pruning, or parameter overwriting, without access to the watermark secret key mk. Suppose $\mathcal{A}$ produces an attacked model w such that the classification accuracy on the main task is preserved, i.e.,

$$\Pr_{x\in\mathcal{D}}\big[\mathbb{N}(x,\mathbf{w}'_{wm})=y\big]\approx\Pr_{x\in\mathcal{D}}\big[\mathbb{N}(x,\mathbf{w})=y\big],$$

but the watermark verification fails $\mathit{Verify}(mk,\mathbf{w},\mathbf{w}'_{wm})=0$.

To prove the robustness of our watermarking scheme, we adopt a hybrid argument. We construct a sequence of hybrid games where in each hybrid, one bit of the watermark secret mk is gradually replaced, resulting in a corresponding modified watermark embedding in the model weights. Due to the binding and hiding properties of the commitment scheme used in watermark construction, adjacent hybrids remain statistically indistinguishable.

Therefore, if adversary $\mathcal{A}$ can succeed in generating such a $\mathbf{w}'$ without access to mk, it implies that $\mathcal{A}$ is capable of distinguishing between hybrids, which contradicts the security guarantees of the commitment scheme. Moreover, the decentralized backdoor mechanism employed in TraceDFL ensures that the watermarks are embedded across multiple feature dimensions. This design inherently provides statistical robustness against adversarial perturbations, thus enhancing the probability that the watermark remains verifiable even after model modification.

Unforgeability. To prove the unforgeability of the watermarking scheme in TraceDFL, we apply a hybrid argument. Suppose a probabilistic polynomial-time adversary $\mathcal{A}$ can generate a forged identity mk' and model $\mathbf{w}_{wm}$ such that $\mathit{Verify}()$. We construct a sequence of hybrid games, where in each step we replace a bit of mk' with the corresponding bit in the true mk. Due to the hiding and binding properties of the commitment scheme, the output distributions of successive hybrids are statistically indistinguishable. Thus, any adversary that distinguishes between the first (forged) and last (real) hybrid would break the hiding property. Consequently, the probability that $\mathcal{A}$ successfully forges a valid watermark identity mk' is negligible.

Security. To prove the security of TraceDFL, we assume TraceDFL uses a statistically hiding and computationally binding commitment scheme (e.g., Pedersen Commitment), and the discrete logarithm (DLOG) assumption holds in the underlying a multiplicative cyclic group $\mathbb{G}$, then the watermarking scheme is secure against any PPT adversary $\mathcal{A}$. We prove security through two components:

- *Hiding.* For the sake of brief description, let $\mathbf{K}_u$ be abbreviated as mk and $com_{\mathbf{K}_u}$ as com_{mk}. The vector Pedersen commitment scheme $com_{mk}=g^{mk}h^r\in\mathbb{G}$, where g,h are public group generators and r is sampled randomly, satisfies statistical hiding. That is, for any two keys mk_1,mk_2, the distributions of com_{mk_1} and com_{mk_2} are statistically indistinguishable. Hence, any adversary $\mathcal{A}$, even with access to $\mathbf{w}_{wm}$, cannot infer any partial or full bit of mk.

- *Binding.* Suppose there exists a PPT adversary $\mathcal{A}$ that outputs a key $mk'\neq mk$ such that $com_{mk'}=com_{mk}$. This implies

$$g^{mk}h^r=g^{mk'}h^{r'}\Rightarrow g^{mk-mk'}=h^{r'-r}.$$

  But this contradicts the binding property of the commitment scheme unless $\mathcal{A}$ can solve the discrete logarithm problem in $\mathbb{G}$, i.e., compute $\log_g h$, which is assumed to be hard. Therefore, under the DLOG assumption and the commitment properties, any PPT adversary cannot extract mk or forge a different valid key mk', which completes the proof.

Exclusivity. Let $\mathcal{T}$ be a Verkle tree constructed from identity keys $\{\mathbf{K}_1,\ldots,\mathbf{K}_n\}$ corresponding to the clients participating in the federated training of model $\hat{\mathbf{w}}_{wm}$. Suppose that each client's fingerprint $\mathbf{K}_u$ is committed into a Verkle tree leaf via a binding commitment scheme. Then, for any malicious internal client with key $\mathbf{K}_{\text{mal}}\notin\mathcal{T}$, the probability that the verification function accepts their exclusive claim of ownership is negligible.

We argue that exclusivity holds in TraceDFL by leveraging the structure and cryptographic properties of the Verkle tree. The TraceDFL framework constructs a Verkle tree whose leaves are derived from the committed fingerprints $\mathbf{K}_u$ of all participating clients. The tree root com acts as the global model ownership commitment. During white-box verification, the function $\mathit{Verify}(mk,\mathbf{w},\hat{\mathbf{w}}_{wm})$ checks whether a fingerprint extracted from $\hat{\mathbf{w}}_{wm}$ can be proven to belong to the Verkle tree rooted at com_{root}. The Verkle tree uses a binding commitment scheme for each leaf, so a malicious client cannot forge an inclusion proof for a $\mathbf{K}_{\text{mal}}\notin\mathcal{T}$. Furthermore, without modifying the root com_{root} (which would alter the public state of the model), the malicious client cannot generate a valid path proof under the original Verkle tree structure.

Therefore, assuming the binding property of commitments and collision-resistance of the Verkle tree hashing, any attempt to pass verification with an identity $mk_{\text{mal}}\notin\mathcal{T}$ will fail with overwhelming probability.

Appendix E Numerical Result

In this section, we conduct extensive experimental evaluation of TraceDFL in terms of fidelity, robustness, verifiability, and traceability.

Appendix E.1 Settings

We implement the relevant components of TraceDFL using Pytorch on the platform with Intel(R) Xeon(R) W-1390 @ 2.80GHz 2.81 GHz, 64GB of RAM. To simulate the operation of DFL, we set the total number of clients to $|\mathcal{U}| = 100$, with 10% of clients randomly selected in each round to perform 3 local training iterations. For black-box watermark embedding, we follow prior work [9,13] and implement the WafflePattern approach to generate the subsequent trigger set. For white-box watermark embedding, we assume a default identity watermark length of 128 bits.

To evaluate the performance of TraceDFL under multi-task settings, we conduct experiments across different domains: AlexNet [33] is used for image classification on the CIFAR-10 dataset, ResNet-34 [34] on the CIFAR-100 and Tiny-imagenet datasets, and StackedLSTM [35] for text generation on the Shakespeare dataset.

Metrics. For fidelity, we use test set accuracy as the evaluation metric. For ownership verification, we assess whether the trigger set accuracy and the reconstructed Verkle tree root commitment are consistent. For traceability, we define the traceability rate (TR) as the proportion of maximum likelihood matches that correctly identify the true model owner.

Baselines. For watermarking performance, we use standard federated learning with embedded watermarks and FedAvg aggregation as the baseline, comparing the effectiveness of TraceDFL against FedIPR [14] and FedTracker [15]. For traceability performance, we employ watermark accuracy (WM acc) and maximum likelihood-based matching rate (MLR) to evaluate the efficiency of TraceDFL.

Appendix E.2 Performance under IID Settings

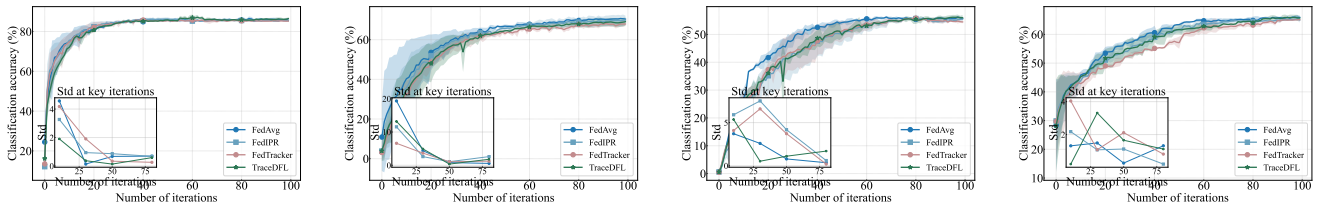
We first evaluate TraceDFL's performance under independent and identically distributed (IID) settings across aspects such as fidelity, robustness, and traceability. The IID configuration aligns with the standard settings for traditional FL.

Appendix E.2.1 *Fidelity Performance*

To evaluate the fidelity of TraceDFL, we examine whether jointly embedding distributed backdoor watermarks and weight-based watermarks preserves the utility of the federated model. For a fair and reliable comparison, each experiment is independently repeated three times, and we report both the mean performance and the corresponding variance. As shown in Fig. E1, the solid curves represent the mean classification accuracy, while the shaded regions indicate the standard deviation (std) across repeated runs, reflecting training stability under watermark embedding.

Overall, TraceDFL demonstrates near-equivalent convergence behavior and final accuracy compared with the unwatermarked FedAvg baseline and representative federated watermarking methods, including FedIPR [14] and FedTracker [15]. Across all evaluated datasets, the classification accuracy of TraceDFL deviates by no more than 1% from the clean model after 60 rounds of federated training. Importantly, the std values at key training iterations are consistently small, indicating that the introduction of both black-box and white-box watermarks does not amplify training variance or destabilize optimization. Compared with FedIPR and FedTracker, TraceDFL exhibits comparable or lower variance in most settings, suggesting that the proposed joint embedding strategy introduces negligible interference to the primary learning task and thus achieves high fidelity.

Fig. E2 further provides a direct comparison of watermark accuracy among TraceDFL, FedIPR, and FedTracker under the same experimental protocol. Similar to Fig. E1, each result is averaged over three independent runs, with shaded regions denoting the standard deviation and inset plots highlighting std values at representative iterations. TraceDFL consistently achieves high and stable watermark accuracy across all datasets, rapidly converging to near-perfect detection rates. In contrast, FedIPR and FedTracker exhibit larger performance fluctuations, particularly in early training stages, as reflected by wider shaded regions and higher std at key iterations. These results indicate that TraceDFL not only preserves model utility but also provides more robust and stable watermark extraction, even under the combined embedding of distributed backdoor and weight-based watermarks. The reduced variance in watermark accuracy demonstrates that TraceDFL yields reliable ownership verification signals across repeated runs, thereby offering clearer advantages in terms of stability and practical traceability compared with existing approaches.



(a) Classification accuracy on CIFAR-10 dataset with AlexNet.

(b) Classification accuracy on CIFAR-100 dataset with ResNet-34.

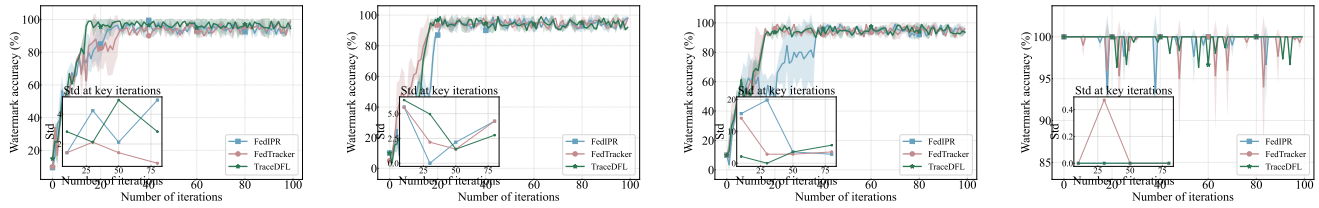
(c) Classification accuracy on Tiny-imagenet dataset with ResNet-34.

(d) Classification accuracy on Shakespeare dataset with StackedLSTM.

Fig. E1. Model performance of TraceDFL compared to FedAvg without embedded watermark, existing work FedIPR [14] and FedTracker [15].

Appendix E.2.2 *Robustness Performance*

- *Robust Performance at Different Pruning Rates Under IID Settings.* To assess robustness against pruning attacks, we simulate parameter pruning by removing weights with the lowest L1 norm and evaluate TraceDFL on multiple datasets. As shown in Fig. E3, even under aggressive pruning (e.g., 50% on Tiny-imagenet), where main accuracy falls below 50%, TraceDFL preserves around 90% watermark accuracy and over 75% MLR. This demonstrates that its watermarking mechanism remains reliable even when the model itself becomes ineffective.



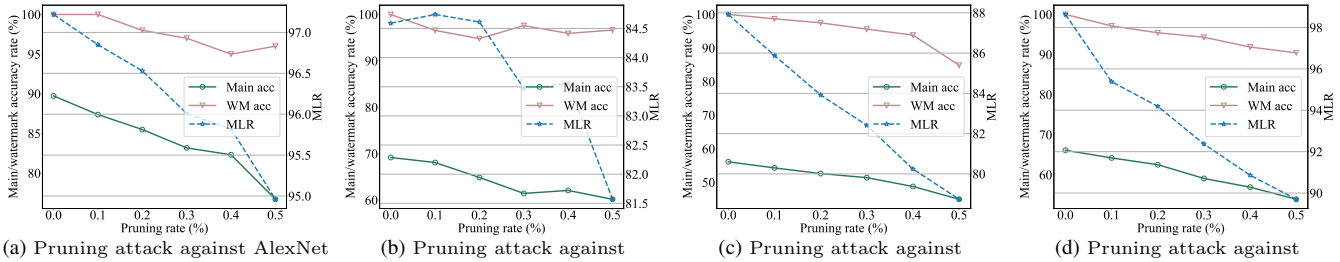
(a) Watermark accuracy on CIFAR-10 dataset with AlexNet.

(b) Watermark accuracy on CIFAR-100 dataset with ResNet-34.

(c) Watermark accuracy on Tiny-imagenet dataset with ResNet-34.

(d) Watermark accuracy on Shakespeare dataset with StackedLSTM.

Fig. E2. Watermark accuracy of TraceDFL compared to existing work FedIPR [14] and FedTracker [15].



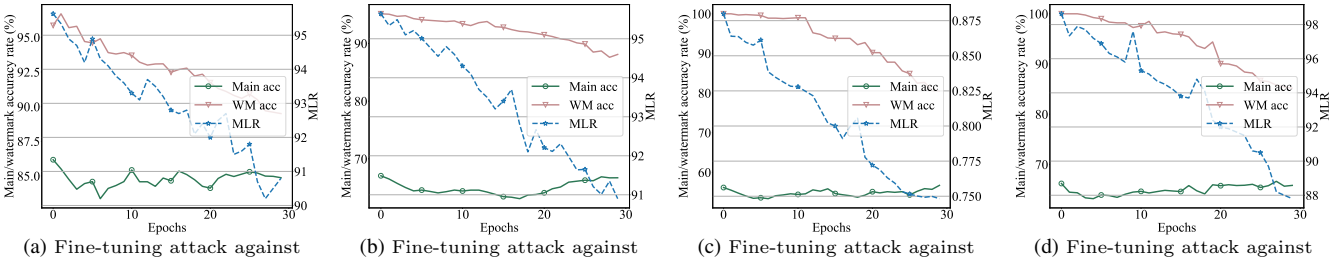
(a) Pruning attack against AlexNet models on the CIFAR-10 dataset.

(b) Pruning attack against ResNet-34 models on the CIFAR-100 dataset.

(c) Pruning attack against ResNet-34 models on the Tiny-imagenet dataset.

(d) Pruning attack against StackedLSTM models on the Shakespeare dataset.

Fig. E3. Model main task accuracy (Main acc, solid line), watermarking accuracy (WM acc, solid line) and extracted identity watermark matching rate (MLR, dashed line) under pruning attacks.



(a) Fine-tuning attack against AlexNet models on the CIFAR-10 dataset.

(b) Fine-tuning attack against ResNet-34 models on the CIFAR-100 dataset.

(c) Fine-tuning attack against ResNet-34 models on the Tiny-imagenet dataset.

(d) Fine-tuning attack against StackedLSTM models on the Shakespeare dataset.

Fig. E4. Model main task accuracy (Main acc, solid line), watermarking accuracy (WM acc, solid line) and extracted identity watermark matching rate (MLR, dashed line) under fine-tuning attacks.

- *Robust Performance at Different Fine-tuning Rates Under IID Settings.* To evaluate the impact of fine-tuning attacks on watermark embedding, we conduct short retraining of the target model using only clean local data, without relying on any trigger samples. Specifically, we perform three consecutive epochs of cross-entropy optimization, equivalent to SGD without momentum, weight decay, or learning-rate scheduling. Since the optimization objective only aligns with the main task distribution, model parameters are gradually pulled back toward the clean distribution's optimum. Meanwhile, batch normalization layers are updated under training mode, refreshing their running statistics. Together, these effects dilute or overwrite the discriminative features introduced by the trigger domain, thereby degrading the watermark detection rate. As shown in Fig. E4, the main task accuracy is largely unaffected by the implementation of this fine-tuning attack, with WM acc and MLR decreasing as the fine-tuning attack iterations progress. Nonetheless, both WM Acc and MLR remain above 85% and 75%, respectively, indicating that ownership verification and traitor tracing remain effective even under extensive post-training model modifications.

- *Robust Performance at Overwrite Attack Under IID Settings.* To illustrate the impact of overwriting attacks on watermark embedding, we first generate a trigger set with uniformly shifted labels without altering the distribution of the primary task data. During the attack, we freeze the BN running statistics to prevent distribution drift. Each attack step combines "new triggers + a small number of clean samples" into a batch, updating it 12 times under SGD using cross-entropy to shift the decision boundary away from the original trigger domain. This process overwrites the watermark information without relying on the original watermark data. Meanwhile, the inclusion of clean samples and a small deployment volume ensures controllable accuracy for the main task. As shown in Table E1, TraceDFL remains largely unaffected before and after the overwrite attack. The errors for all three metrics remain below 5%, staying within a controllable range that does not compromise ownership confirmation and traceability performance.

Table E1 Performance under overwrite attacks across different datasets under IID settings (%)

Metric	CIFAR-10		CIFAR-100		Tiny-imagenet		Shakespeare	
	Before	After	Before	After	Before	After	Before	After
Main acc	86.89	85.34 ± 0.36	70.12	68.87 ± 0.32	56.25	55.09 ± 0.74	66.74	64.69 ± 0.75
WM acc	100	98.9 ± 0.16	99.85	97.13 ± 0.32	100.0	98.24 ± 0.11	100.0	97.31 ± 0.40
MLR	93.78	92.64 ± 0.07	95.14	91.76 ± 0.58	88.23	85.68 ± 0.47	98.63	95.80 ± 0.76

Appendix E.2.3 *Traceability Performance*

We evaluate the tracing performance of TraceDFL on models trained with varying numbers of clients across four distinct datasets, with the identity watermark length fixed at 128 bits. Specifically, we assess the backdoor watermark accuracy (WM Acc) on the trigger set and the maximum likelihood-based matching rate (MLR) of the extracted identity watermark. As shown in Table E2, TraceDFL

Table E2 Watermark accuracy and MLR (%) under different numbers of clients.

Model	Metric	Number of Clients				
		10	20	30	40	50
AlexNet on CIFAR-10	WM acc	98.9	99.3	100	99.7	99.8
	MLR	93.78	93.16	94.22	95.84	92.48
ResNet-34 on CIFAR-100	WM acc	98.0	99.01	99.85	98.46	99.37
	MLR	95.14	93.84	94.54	94.72	94.66
ResNet-34 on Tiny-imagenet	WM acc	95.8	97.0	98.3	100	100
	MLR	88.23	87.85	87.71	87.92	86.5
StackedLSTM on Shakespeare	WM acc	90.07	100	100	99.67	100
	MLR	90.76	95.38	94.73	100	98.63

consistently maintains a watermark accuracy exceeding 90% across all datasets. Furthermore, as the number of clients increases, the embedded identity watermarks remain highly verifiable, with matching rates exceeding 80%, thereby supporting the effectiveness of TraceDFL in verifying model ownership under multi-client federated learning scenarios.

We further investigate the impact of varying identity watermark lengths on the tracing performance of models generated by TraceDFL. As presented in Table E3, increasing the watermark length does not negatively affect TraceDFL's ability to trace model ownership. The majority of observed MLR remains stable, confirming that the length of the identity watermark has negligible influence on tracing effectiveness.

Appendix E.3 Performance under Non-IID Settings

To address the Non-IID data encountered in real-world FL deployments, we employ a Dirichlet distribution to simulate heterogeneous client data configurations. Based on this , we evaluate TraceDFL's performance in terms of fidelity, robustness, and traceability.

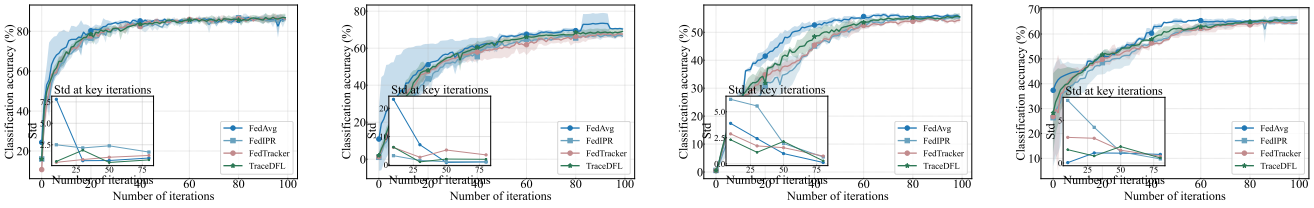
Table E3 Maximum likelihood-based matching rate (%) under different numbers of bits.

Model	Number of bits			
	64	128	256	512
AlexNet on CIFAR-10	90.84	93.78	92.48	93.82
ResNet-34 on CIFAR-100	95.69	95.14	95.85	94.48
ResNet-34 on Tiny-imagenet	86.53	88.23	86.56	87.10
StackedLSTM on Shakespeare	86.13	90.76	95.38	100.00

Appendix E.3.1 *Fidelity Performance*

We evaluate TraceDFL under non-IID settings by partitioning client datasets using a Dirichlet distribution with concentration parameter 0.6. As shown in Fig. E5, TraceDFL achieves model accuracy and convergence behavior comparable to vanilla FL and existing watermarking schemes across all evaluated datasets. Despite data heterogeneity, the introduction of joint black-box and white-box watermarks does not noticeably degrade the main task performance, and the shaded regions indicate limited variance across repeated runs, confirming stable training under non-IID conditions.

Fig. E6 further reports the watermark accuracy under the same non-IID settings. TraceDFL consistently attains high and stable watermark accuracy, with smaller standard deviation at key training iterations compared to FedIPR and FedTracker. These results demonstrate that TraceDFL maintains reliable ownership verification and traceability even in heterogeneous data environments, without sacrificing model utility.



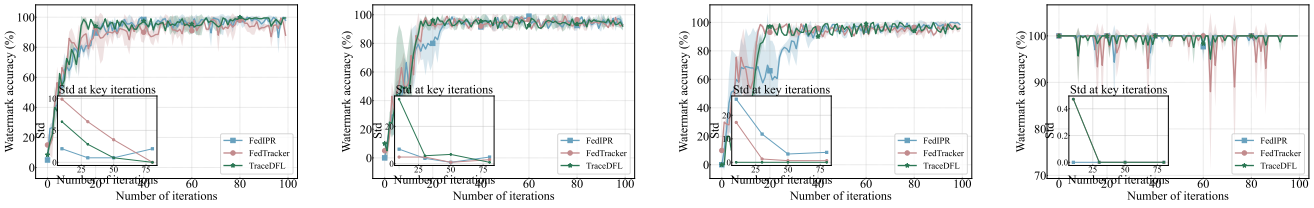
(a) Classification Accuracy for AlexNet on CIFAR-10 under Non-IID Settings.

(b) Classification accuracy for ResNet-34 on CIFAR-100 dataset under Non-IID Settings.

(c) Classification accuracy for ResNet-34 on Tiny-imagenet dataset under Non-IID Settings.

(d) Classification accuracy for StackedLSTM on Shakespeare dataset under Non-IID Settings.

Fig. E5. Performance comparison of TraceDFL models in Non-IID settings: FedAvg without embedded watermarks, existing work FedIPR [14], and FedTracker [15]



(a) Watermark Accuracy for AlexNet on CIFAR-10 under Non-IID Settings.

(b) Watermark accuracy for ResNet-34 on CIFAR-100 dataset under Non-IID Settings.

(c) Watermark accuracy for ResNet-34 on Tiny-imagenet dataset under Non-IID Settings.

(d) Watermark accuracy for StackedLSTM on Shakespeare dataset under Non-IID Settings.

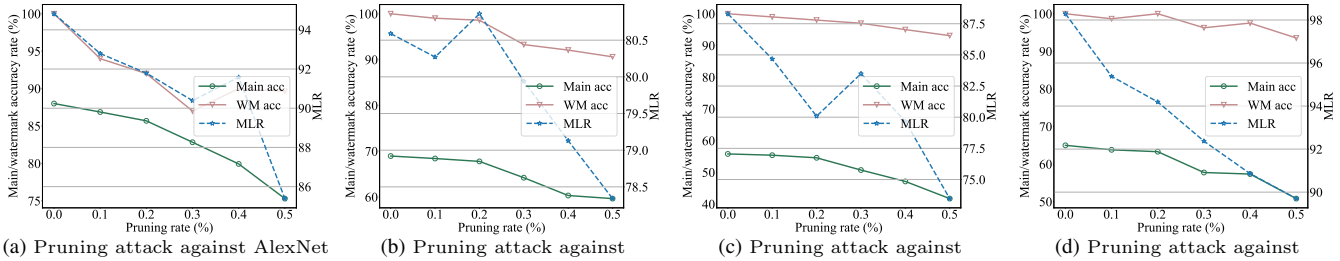
Fig. E6. Comparison of watermarking accuracy between TraceDFL and existing studies FedIPR [14] and FedTracker [15] under non-independent identically distributed settings

Appendix E.3.2 *Robustness Performance*

We adopt implementation approaches similar to pruning attacks, fine-tuning attacks, and overwriting attacks under Non-IID settings to evaluate the robustness of TraceDFL in a heterogeneous setting with a fixed Dirichlet distribution of 0.6. A smaller Dirichlet distribution indicates that the client holds a sparser dataset.

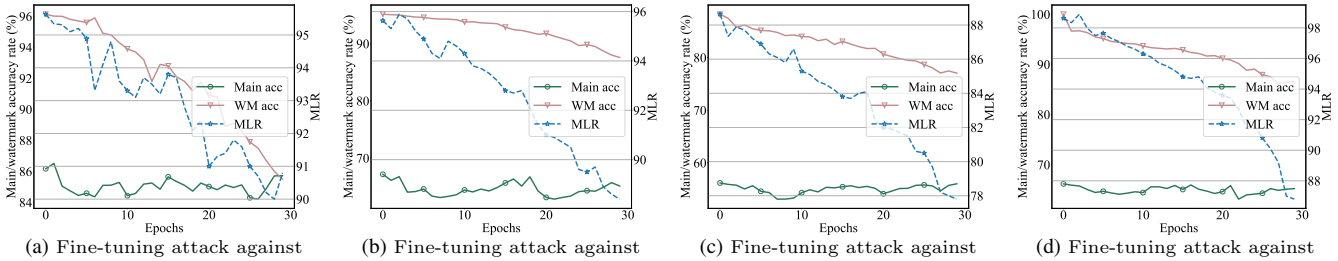
- *Robust Performance at Different Pruning Rates Under Non-IID Settings.* Fig. E7 shows that the main task accuracy, watermark accuracy, and identity-key matching rate all decrease as the pruning ratio increases. However, the variation trends of these three metrics remain close to those observed under the IID setting, indicating that heterogeneous data has little impact. Even in the worst-case scenario, TraceDFL still succeeds in safeguarding model copyright and tracing malicious leakers.

- *Robust Performance at Different Fine-tuning Rates Under Non-IID Settings.* Under Non-IID settings, fine-tuning attacks exhibit a similar impact pattern on TraceDFL as in IID scenarios. As shown in Fig. E8, with the increase of fine-tuning epochs, the main



(a) Pruning attack against AlexNet models on the CIFAR-10 dataset.
(b) Pruning attack against ResNet-34 models on the CIFAR-100 dataset.
(c) Pruning attack against ResNet-34 models on the Tiny-imagenet dataset.
(d) Pruning attack against StackedLSTM models on the Shakespeare dataset.

Fig. E7. Model performance under Non-IID settings: pruning attacks impact main task accuracy (Main acc, solid line), watermark accuracy (WM acc, solid line), and extracted identity watermark matching rate (MLR, dashed line).



(a) Fine-tuning attack against AlexNet models on the CIFAR-10 dataset.
(b) Fine-tuning attack against ResNet-34 models on the CIFAR-100 dataset.
(c) Fine-tuning attack against ResNet-34 models on the Tiny-imagenet dataset.
(d) Fine-tuning attack against StackedLSTM models on the Shakespeare dataset.

Fig. E8. Model performance under Non-IID settings: fine-tuning attacks impact main task accuracy (Main acc, solid line), watermark accuracy (WM acc, solid line), and extracted identity watermark matching rate (MLR, dashed line).

task accuracy (green solid line) remains relatively stable, while both the watermark accuracy (red solid line) and the identity watermark matching rate (blue dashed line) gradually decline. This indicates that although heterogeneous data distribution slightly accelerates the erosion of watermark features, the overall degradation trend is consistent with IID cases. Importantly, even under Non-IID conditions, TraceDFL retains sufficient robustness to preserve ownership verification and traitor tracing capability against fine-tuning attacks.

- *Robust Performance at Overwrite Attack Under Non-IID Settings.* As shown in Table E4, under the Non-IID setting, the accuracy loss of the main task remains below 2%. The minimal variation in WM acc and MLR indicates that both distributed backdoor watermark embedding and identity watermark embedding maintain strong robustness against overwriting attacks.

Table E4 Performance under overwrite attacks across different datasets under Non-IID settings (%)

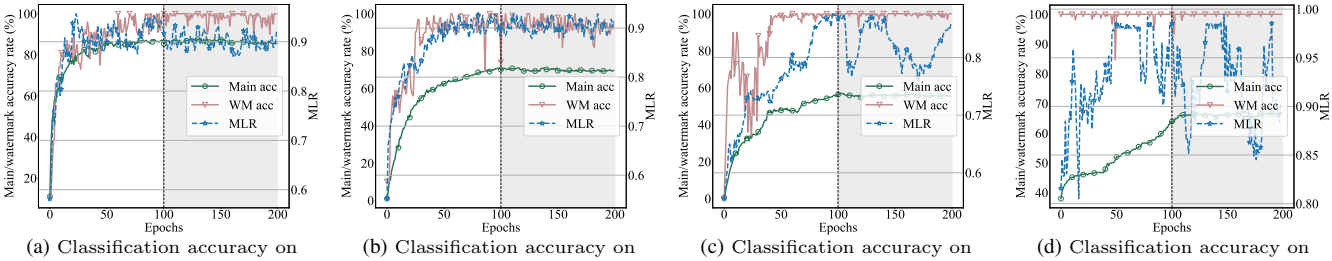
Metric	CIFAR-10		CIFAR-100		Tiny-imagenet		Shakespeare	
	Before	After	Before	After	Before	After	Before	After
Main acc	85.57	85.09 ± 0.16	69.57	67.95 ± 0.34	55.46	54.76 ± 0.47	65.83	65.19 ± 0.06
WM acc	100	97.75 ± 0.44	95.05	94.64 ± 0.13	99.0	97.36 ± 0.41	100.0	98.24 ± 0.37
MLR	93.46	91.75 ± 0.26	91.75	90.85 ± 0.23	86.47	85.08 ± 0.64	97.45	95.85 ± 0.38

Appendix E.4 Performance in Reliability and Scalability

As illustrated in Fig. E9, after 100 training epochs we stop embedding the black-box watermark and identity keys (marked by the vertical dashed line). In the subsequent stage (shaded region), both watermark accuracy (WM acc, red line) and identity-key matching rate (MLR, blue dashed line) remain consistently high, demonstrating that the watermarking signals have been firmly integrated into the model parameters and retain strong detectability even without further embedding. Meanwhile, the main task accuracy (green line) continues to improve and converges steadily across all datasets, confirming that TraceDFL achieves robust ownership verification and traitor tracing without sacrificing model utility.

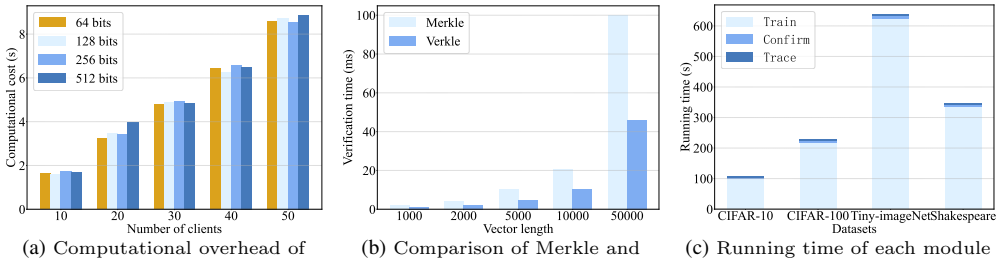
Fig. E10 reports the computational cost of TraceDFL across different modules and settings. As shown in Fig. E10(a), the reconstruction overhead of Verkle trees scales primarily with the number of clients rather than the identity key length, with even the extreme case of 50 clients requiring only about 8 seconds, which is negligible compared to the overall training cycle. We implemented Merkle trees and Verkle trees based on SHA-256, and conducted benchmark tests to evaluate the overhead associated with different vector lengths.

Fig. E10(b) further compares verification costs between Merkle and Verkle proofs, where Verkle consistently achieves lower latency and demonstrates clear advantages at large vector lengths (e.g., 45 ms vs. nearly 100 ms at 50,000). Fig. E10(c) provides a breakdown of running time on different datasets, showing that training dominates the total cost, while confirmation and tracing introduce only



(a) Classification accuracy on CIFAR-10 dataset with AlexNet.
(b) Classification accuracy on CIFAR-100 dataset with ResNet-34.
(c) Classification accuracy on Tiny-imagenet dataset with ResNet-34.
(d) Classification accuracy on Shakespeare dataset with StackedLSTM.

Fig. E9. Sustainability of indicators before and after embedding across different datasets under Non-IID settings.



(a) Computational overhead of constructing Verkle trees under different bit configurations.
(b) Comparison of Merkle and Verkle verification overheads at different vector lengths.
(c) Running time of each module across different datasets.

Fig. E10. Computational cost for various modules in TraceDFL

marginal overhead (less than 10 seconds even on Tiny-imagenet). Taken together, these results demonstrate that TraceDFL's design introduces only minimal additional overhead, even under the most challenging conditions. The Verkle-based verification mechanism not only scales efficiently with client size and vector length but also ensures that the computational cost of confirmation and tracing remains negligible compared to federated training itself. This ensures that TraceDFL is both scalable and practical for real-world decentralized federated learning deployments.

References

1 Kairouz P, McMahan H B, Avent B, et al. Advances and open problems in federated learning[J]. Found. Trends Mach. Learn, 2021, 14(1–2): 1–210.
2 Ni L Q, Qi Y F, Bao X Y, et al. Temporal-spectral correlation dynamics of Raman random fiber laser[J]. Sci China Inf Sci, 2025, 68(4).
3 Yuan W, Yang C Q, Qu L, et al. Robust federated contrastive recommender system against targeted model poisoning attack[J]. Sci China Inf Sci, 2025, 68(4).
4 Tian Y, Wang S, Xiong J, et al. Robust and privacy-preserving decentralized deep federated learning training: Focusing on digital healthcare applications[J]. IEEE/ACM Trans. Comput. Biol. Bioinform, 2023, 21(4): 890–901.
5 Dayan I, Roth H R, Zhong A, et al. Federated learning for predicting clinical outcomes in patients with COVID-19[J]. Nat. Med, 2021, 27(10): 1735–1743.
6 Miao Y B, Kuang D, Wang L H, et al. Efficient privacy-preserving federated learning under dishonest-majority setting[J]. Sci China Inf Sci, 2024, 67(5).
7 Rajbhandari S, Rasley J, Ruwase O, et al. ZeRO: Memory optimizations toward training trillion-parameter models[C]. In: Proceedings of SC20: Int. Conf. High Performance Computing, Networking, Storage and Analysis. Piscataway, NJ: IEEE, 2020: 1–16.
8 Cao X, Jia J, Gong N Z. IPGuard: Protecting intellectual property of deep neural networks via fingerprinting the classification boundary[C]. In: Proceedings of ACM Asia Conference on Computer and Communications Security (AsiaCCS). New York, NY: ACM, 2021: 14–25. New York, NY: ACM, 2021: 14–25.
9 Uchida Y, Nagai Y, Sakazawa S, et al. Embedding watermarks into deep neural networks[C]. In: Proceedings of ACM Int. Conf. Multimedia Retrieval (ICMR). New York, NY: ACM, 2017: 269–277.
10 Adi Y, Baum C, Cisse M, et al. Turning your weakness into a strength: Watermarking deep neural networks by backdooring[C]. In: Proceedings of USENIX Security Symposium (USENIX Security 2018). Berkeley, CA: USENIX Association, 2018: 1615–1631.
11 Tekgul B G A, Xia Y, Marchal S, et al. Waffle: Watermarking in federated learning[C]. In: Proceedings of Int. Symp. Reliable Distributed Systems (SRDS). Piscataway, NJ: IEEE, 2021: 310–320.
12 Darvish Rouhani B, Chen H, Koushanfar F. Deepsigns: An end-to-end watermarking framework for ownership protection of deep neural networks[C]. In: Proceedings of Int. Conf. Architectural Support for Programming Languages and Operating Systems (ASPLOS). New York, NY: ACM, 2019: 485–497.
13 Yang W, Shao S, Yang Y, et al. Watermarking in secure federated learning: A verification framework based on client-side backdooring[J]. ACM Trans. Intell. Syst. Technol, 2023, 15(1): 1–25.
14 Li B, Fan L, Gu H, et al. FedIPR: Ownership verification for federated deep neural network models[J]. IEEE Trans. Pattern Anal. Mach. Intell, 2022, 45(4): 4521–4536.
15 Shao S, Yang W, Gu H, et al. FedTracker: Furnishing ownership verification and traceability for federated learning model[J]. IEEE Trans. Dependable Secure Comput, 2024, 22(1): 114–131.
16 Fang M., Zhang Z., Hairi, et al. Byzantine-robust decentralized federated learning[C]. In: Proceedings of ACM SIGSAC Conference on Computer and Communications Security (CCS 2024). ACM, 2024: 2874–2888.
17 Beltrán E. T. M., Gómez Á. L. P., Feng C., et al. Fedstellar: A platform for decentralized federated learning[J]. Expert Syst. Appl., 2024, 242: 122861.
18 Beltrán E. T. M., Pérez M. Q., Sánchez P. M. S., et al. Decentralized federated learning: Fundamentals, state of the art, frameworks, trends, and challenges[J]. IEEE Commun. Surv. Tutor., 2023, 25(4): 2983–3013.

19 Guo Z., Gao Z., Liu Q., et al. Verkle-accumulator-based stateless transaction validation (VA-STV) scheme for the blockchain-based IoT network[J]. IEEE Internet Things J., 2023, 11(1): 543–558.
20 Wang S., Ma J., Huang Q., et al. Verkle-accumulator-based multiple state verifiable and updatable (VA-MSVU) scheme for blockchain[J]. J. Netw. Comput. Appl., 2025: 104392.
21 Nie H, Lu S. FedCRMW: Federated model ownership verification with compression-resistant model watermarking[J]. Expert Syst. Appl., 2024, 249: 123776.
22 Xu J, Hu R, Kotevska O, et al. Traceable black-box watermarks for federated learning[J]. arXiv, 2025, arXiv:2505.13651.
23 Yang Y, Li Q, Hong Y, et al. FedGMark: Certifiably robust watermarking for federated graph learning[C]. In: Proceedings of Advances in Neural Information Processing Systems (NeurIPS 2024). Red Hook, NY: Curran Associates, 2024: 48971–48995.
24 Luo Y, Li Y, Qin S, et al. Copyright protection framework for federated learning models against collusion attacks[J]. Inf. Sci., 2024, 680: 121161.
25 Li F Q, Wang S L, Liew A W C. Merkle-sign: Watermarking framework for deep neural networks in federated learning[J]. arXiv, 2021, arXiv:2105.03167.
26 Rodríguez-Lois E, Pérez-González F. Exploring federated learning dynamics for black- and white-box DNN traitor tracing[C]. In: Proceedings of Int. Conf. Federated Learning Technologies and Applications (FLTA). Piscataway, NJ: IEEE, 2024: 282–289.
27 Xiang A, Gao H, Tian Y, et al. H2CT: Asynchronous distributed key generation with high computational efficiency and threshold security in blockchain network[J]. IEEE Internet Things J., 2024, in press.
28 Xie Z., Zhang H., Duan S., et al. Everything distributed and asynchronous: A practical system for key management service[J]. IEEE Trans. Parallel Distrib. Syst., 2025.
29 Das S., Yurek T., Xiang Z., et al. Practical asynchronous distributed key generation[C]. In: Proceedings of IEEE Symposium on Security and Privacy (SP 2022). IEEE, 2022: 2518–2534.
30 Xie C, Huang K, Chen P Y, et al. DBA: Distributed backdoor attacks against federated learning[C]. In: Proceedings of Int. Conf. Learn. Representations (ICLR). Online: OpenReview.net, 2019.
31 Wang S, Tian Y, Xiong J, et al. VerifyDFL: Secure aggregation for decentralized federated learning with input validation in mobile edge intelligence[J]. IEEE Trans. Cogn. Commun. Netw., 2025, in press.
32 Libert B. Vector commitments with proofs of smallness: Short range proofs and more[C]. In: Proceedings of IACR Int. Conf. Public-Key Cryptography (PKC). Cham: Springer, 2024: 36–67.
33 Krizhevsky A, Sutskever I, Hinton G E. ImageNet classification with deep convolutional neural networks[C]. In: Proceedings of Advances in Neural Information Processing Systems (NIPS 2012). Red Hook, NY: Curran Associates, 2012: 1097–1105.
34 He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]. In: Proceedings of IEEE Conf. Computer Vision and Pattern Recognition (CVPR). Piscataway, NJ: IEEE, 2016: 770–778.
35 Cui Z, Ke R, Pu Z, et al. Stacked bidirectional and unidirectional LSTM recurrent neural network for forecasting network-wide traffic state with missing values[J]. Transp. Res. Part C Emerg. Technol., 2020, 118: 102674.