


# A White-Box Watermarking Modulation for Encrypted DNN in Homomorphic Federated Learning

Mohammed Lansari<sup>1,2</sup> <sup>a</sup>, Reda Bellafqira<sup>1</sup>, Katarzyna Kapusta<sup>2</sup>, Vincent Thouvenot<sup>2</sup>, Olivier Bettan<sup>2</sup> and Gouenou Coatrieux<sup>1</sup>

<sup>1</sup>IMT Atlantique, Inserm UMR 1101, 29200 Brest, France

<sup>2</sup>ThereSIS, Thales SIX, 91120 Palaiseau, France

**Keywords:** DNN Watermarking, Federated Learning, Homomorphic Encryption, Intellectual Property Protection.

**Abstract:** Federated Learning (FL) is a distributed paradigm that enables multiple clients to collaboratively train a model without sharing their sensitive local data. In such a privacy-sensitive setting, Homomorphic Encryption (HE) plays an important role by enabling computations on encrypted data. This prevents the server from reverse-engineering the model updates, during the aggregation, to infer private client data, a significant concern in scenarios like the healthcare industry where patient confidentiality is paramount. Despite these advancements, FL remains susceptible to intellectual property theft and model leakage due to malicious participants during the training phase. To counteract this, watermarking emerges as a solution for protecting the intellectual property rights of Deep Neural Networks (DNNs). However, traditional watermarking methods are not compatible with HE, primarily because they require the use of non-polynomial functions, which are not natively supported by HE. In this paper, we address these challenges by proposing the first white-box DNN watermarking modulation on a single homomorphically encrypted model. We then extend this modulation to a server-side FL context that complies with HE's processing constraints. Our experimental results demonstrate that the performance of the proposed watermarking modulation is equivalent to the one on unencrypted domain.


## 1 INTRODUCTION

Federated Learning (FL) enables multiple data owners to collaboratively train machine learning or deep learning models without sharing their private data (McMahan et al., 2017). In a FL round, each client trains the shared model with its own data. Subsequently, the server selects a subset of clients, gathers and aggregates their model updates, and forms what is known as the global model for the current round. This aggregated model is then distributed back to the clients for further rounds of training, a process repeated until the global model converges. Despite the advantages of data privacy, the integrity of the server and clients cannot always be ensured, leading to several security challenges.

The server may exhibit an honest-but-curious behavior, meaning that while it follows to the federated learning (FL) protocol, it might attempt to infer information about client data based on the updates received in each round. Common inference attacks include inversion attacks (He et al., 2019) and member-

ship inference attacks (Hu et al., 2022; Shokri et al., 2017), aiming to reconstruct training data from model parameters. A solution to this issue is the use of Homomorphic Encryption (HE) (Benaissa et al., 2021), which encrypts client updates before they are sent to the server for aggregation. HE allows the server to perform aggregation on encrypted updates without decryption, thus preventing the server or external adversaries from conducting inference attacks (Zhang et al., 2020; Xiong et al., 2024).

The second concern involves the potential for model redistribution by malicious participants, either during or after the training process, which could lead to unauthorized profits. Protecting the intellectual property (IP) of the model is important in this context. Watermarking (Uchida et al., 2017; Fan et al., 2019; Darvish Rouhani et al., 2019; Bellafqira and Coatrieux, 2022; Kallas and Furon, 2023; Pierre et al., 2023), which embeds a secret watermark into the model's parameters or behavior for later extraction by the owner, has emerged as a promising solution. This technique can be applied either client-side or server-side in FL context.

<sup>a</sup>  <https://orcid.org/0009-0004-8025-5587>

Although several solutions propose watermarking the model from the client-side (Liu et al., 2021; Liang and Wang, 2023; Yang et al., 2023; Li et al., 2022; Yang et al., 2022), which are compatible with clients' update encryption, no existing solution combines server-side watermarking with homomorphic encryption in a Federated Learning (FL) context. This is primarily because embedding the watermark server-side, when updates are encrypted, poses challenges due to the complexity of the watermarking process under HE constraints. Furthermore, server-side watermarking has been shown to be more robust and efficient than client-side watermarking (Shao et al., 2024). More clearly, if each client watermarks its local model, the aggregation stage may lead to watermark collusion, and consequently, some clients' watermarks may not be effectively inserted.

For these reasons, in this paper, we explore the possibility of server-side watermarking in FL using HE as a privacy-preserving mechanism. Our contributions are as follows:

- We leverage existing FL White-Box watermarking algorithms (Shao et al., 2024; Li et al., 2022) along with homomorphic encryption in order to watermark a model with minimal modifications of the FL procedure. Specifically, we address the challenge posed by non-polynomial functions in the embedding process, which can be approximated by low-degree polynomial functions.
- We then implement this scheme within a client-server FL framework, demonstrating that it is possible to protect both the confidentiality of model parameters and intellectual property from an honest-but-curious server tasked with watermarking the model.
- Our experimental results indicate that embedding the watermark using approximated functions does not detrimentally impact the primary learning task or the effectiveness of the watermark, even when subjected to removal attacks such as fine-tuning and pruning.

## 2 BACKGROUND & RELATED WORKS

In this section, we give a brief overview of HE. Then we define FL and how HE can be used as a security mechanism against an honest-but-curious server. Finally, we introduce FL watermarking to position our work among existing solutions.

### 2.1 Homomorphic Encryption

Homomorphic Encryption (HE) is a form of encryption that allows computations to be carried out on ciphertexts, generating an encrypted result that, when decrypted, matches the outcome of operations performed on the plaintext. This capability enables secure processing of encrypted data without giving access to the underlying data, thus preserving the privacy and confidentiality of data during processing (Bouslimi et al., 2016; Baumstark et al., 2023; Wang et al., 2024). HE can be categorized into three types: Partially Homomorphic Encryption (PHE), which supports a single type of operation (either addition or multiplication) unlimited times (e.g., Paillier cryptosystem (Paillier, 1999)); Somewhat Homomorphic Encryption (SHE), allowing both addition and multiplication but with a limited computation depth, including early versions of FHE schemes (e.g., BFV scheme (Fan and Vercauteren, 2012)); and Fully Homomorphic Encryption (FHE), enabling unlimited operations of both addition and multiplication on ciphertexts. FHE includes schemes like Gentry's original construction, BGV (Brakerski et al., 2014), and TFHE (Chillotti et al., 2021). BGV and TFHE expand the capabilities of FHE, with BGV allowing computations only over integer arithmetic and TFHE enabling a broader range of operations, including non-polynomial functions, but they are unpractical for various cases regarding the time consumption (Clet et al., 2021).

The CKKS scheme (Cheon et al., 2017) is particularly used for enabling operations on encrypted floating-point numbers, indispensable for applications requiring high precision such as machine learning and statistical analysis. Its scalability and efficiency are advantageous for computational tasks involving vectors, making CKKS suitable for large-scale applications. Despite its advantages, CKKS is primarily designed to support the computation over polynomial functions.

### 2.2 Secure Federated Learning

Federated Learning (FL) is a machine learning framework that enables  $K \in \mathbb{N}^*$  participants to collaboratively train a model  $M^G$  across  $R$  rounds of exchange while maintaining the privacy of their data  $D^k$ . In the client-server model of FL (McMahan et al., 2017), the server initializes the global model  $M_0^G$ . At each round  $t$ , the global model  $M_t^G$  is distributed to a subset  $S_t \subseteq \{1, \dots, K\}$  consisting of  $C \times K$  randomly selected clients, where  $C \in (0, 1]$ . Each client  $k \in S_t$  trains the model locally using their private dataset  $D^k$  and

sends their updated model  $M_{t+1}^k$  back to the server. The server then aggregates these updates to construct the new global model  $M_{t+1}^G$ . This process repeats until  $R$  rounds are completed ( $t = R$ ).

The primary vulnerability of FL lies in potential privacy attacks, such as membership inference (Hu et al., 2022; Shokri et al., 2017) and model inversion attacks (He et al., 2019). Specifically, in client-server FL, the server aggregates the parameters of received models at each round. However, the server may be honest but curious, attempting to infer information about the client’s data from their parameters without violating the FL protocol. This concern highlights the necessity for Secure FL. HE is a prevalent method employed to mitigate this privacy issue (Zhang et al., 2020).

Several FL frameworks already support fully homomorphic encryption. For instance, the NVFLARE FL framework (NVIDIA, 2023) developed by NVIDIA implements the CKKS cryptosystem using the TenSEAL library (Benaissa et al., 2021).

For the remainder of this paper, following the approach proposed in (Zhang et al., 2020), we consider a Fully Homomorphic Encryption (FHE) cryptosystem characterized by an encryption function  $Encrypt$ , a corresponding decryption function  $Decrypt$ , and a pair of public and private keys  $(Pub_{key}, Priv_{key})$ . Clients encrypt their models  $M_t^k$  using  $\mathcal{E}$  and the public key  $Pub_{key}$  before transmission to the server. An encrypted model is denoted as  $\mathcal{E}(M_t^k)$ , meaning all parameters  $\{w_1, \dots, w_L\}$  of  $M$  are encrypted as  $\{\mathcal{E}(w_1), \dots, \mathcal{E}(w_L)\}$ . Leveraging HE, the server can aggregate these encrypted models, for instance, using FedAvg (McMahan et al., 2017). When clients receive the encrypted model  $\mathcal{E}(M_{t+1}^G)$ , they decrypt it using  $Decrypt$  and the private key  $Priv_{key}$  to continue training on  $M_{t+1}^G$ . The CKKS cryptosystem is particularly suitable for FL due to its computational and communication efficiency, making it an optimal choice for addressing the challenges of FL (Miao et al., 2022).

### 2.3 FL Watermarking

Inspired by multimedia watermarking (Bas et al., 2016), DNN watermarking is a promised solution to prove ownership of ML models (Sun et al., 2023; Boenisch, 2021; Bellafqira and Coatrieux, 2022; Li et al., 2021b; Lukas et al., 2022; Xue et al., 2021). This technique involves introducing a secret modification into the model’s parameters or behavior to embed a watermark (a secret message), which the owner can later use to verify the presence of the embedded watermark. We distinguish two types of watermark-

ing according to the setting: Black-Box and White-Box. Black-Box watermarking consists of embedding the secret into the behavior of the model. The verification process can then be performed without having full access to the model. In this article, we focus on the White-Box setting. This latter aims to hide the secret in the model by assuming that we have access to its parameters during the verification stage. Usually, the goal is to insert a robust watermark (a binary string  $b$ ) into the model’s parameters while preserving the DNN model performance in the main task.

Despite great results in centralized training, DNN watermarking solutions are hardly utilizable in FL. Tekgul *et al.* (Tekgul et al., 2021) illustrates well the fact that using existing centralized watermarking techniques in FL is possible in two ways: embedding the watermark before the FL or at the end. However, both approaches risk the model being redistributed by a malicious client with the watermark either absent or barely present. Addressing DNN watermarking within the FL context introduces new constraints and security considerations (Lansari et al., 2023). Based on these considerations, two main strategies for embedding have been explored. The first allows each client  $k$  to watermark their local model  $M_t^k$ , treating the server as honest but curious. Various techniques supporting client-side embedding have been proposed (Li et al., 2022; Yang et al., 2023; Yang et al., 2022; Liu et al., 2021; Liang and Wang, 2023), which are compatible with encrypting updates before transmission to the server, as indicated in Table 1. Nevertheless, client-side watermarking faces several drawbacks:

1. **Client Selection:** This approach involves selecting a subset of clients in each round for communication efficiency, yet the effectiveness of the embedding process in this context remains unproven.
2. **Cross-Device Setting:** Implementing client-side watermarking can be challenging when  $K \geq 10^{10}$  and devices have limited computational power.
3. **Multiple Watermarks:** With multiple clients attempting to embed their binary strings, the technique must prevent conflicts and interference between the various watermarks.

Given these challenges, server-side watermarking presents a more viable solution. In this context, the server embeds the watermark post-aggregation. Several server-side techniques have been developed (Tekgul et al., 2021; Shao et al., 2024; Chen et al., 2023a; Li et al., 2021a; Yu et al., 2023); however, as shown in Table 1, none of the current state-of-the-art watermarking solutions incorporate HE (Homomorphic Encryption) as a privacy measure to prevent the

Table 1: Our method among existing FL White-Box watermarking techniques.

Existing Works	White-Box	Embedding	HE Compatibility
FedIPR (Li et al., 2022)	✓	Client(s)	✓
FedCIP (Liang and Wang, 2023)	✓	Client(s)	✓
FedTracker (Shao et al., 2024)	✓	Server	✗
Yu et al. (Yu et al., 2023)	✓	Server	✗
Proposed method	✓	Server	✓

server from inferring information about the clients’ datasets during the watermark embedding process.

### 3 PROPOSED METHOD

In this section, we present an overview of our new white-box watermarking modulation in the context of Homomorphic Encrypted Federated Learning. We then detail how the embedding and extraction processes are conducted on the context of a single encrypted model before generalizing to the context of homomorphic encrypted federated learning.

#### 3.1 Overview of the Proposed Method

In this subsection, we introduce the first watermarking modulation compatible with homomorphically encrypted DNN models in the context of Federated Learning (FL). Our approach relies on existing FL White-Box watermarking algorithms, adapting them to function with encrypted parameters using Fully Homomorphic Encryption (FHE). Our contributions include the redesign of the embedding processes by approximating non-polynomial functions with low-degree polynomials.

Figure 1 provides an overview of our proposed method when deployed in a client-server FL framework. At the initialization phase (Step 1), the server has unencrypted access to the initial global model  $M_0^G$  and watermarks it using our watermarking modulation until the watermark  $b$  is successfully embedded. This initial step does not compromise the privacy of client data since the global model is not yet trained on clients’ data.

Once the model is watermarked, it is scattered to clients for training on their private data (Step 2 in Figure 1). After the clients’ models are updated, each client encrypts its model and sends it back to the server, which gathers/aggregates them to form the new global model (Step 3 in Figure 1). The server then watermarks the aggregated encrypted model using our proposed modulation (Step 4 in Figure 1) and scatters it back to the clients for a new round of training (Step 5 in Figure 1). This cycle of scattering, local

training by clients, gathering, and watermarking by the server is repeated until the global model achieves convergence.

For the sake of simplicity, we first explain how to embed our White-Box watermark in an encrypted model that is supposed to be in a centralized setting. Then we generalize the algorithm to  $K$  clients in a client-server FL framework. The following embedding technique is inspired by the White-Box insertion proposed by (Li et al., 2022; Shao et al., 2024) and can be viewed as an extension of this type of technique in the encrypted domain.

#### 3.2 Watermark an Encrypted Model

The White-Box watermarking process aims to embed a watermark  $b \in \{-1, 1\}^N$ , coded into  $N$  bits, into the model’s parameters. To do so, we define an extraction function  $Ext(\cdot)$  that will extract a subset of the model parameters, based on a secret key  $K_{ext}$ , in which we embed the watermark. Then we define a projection function  $Proj(\cdot)$  that will project the selected parameters into the watermark space using a secret key  $K_{proj}$ . During the watermark verification stage, these two functions will be used to recover the embedded watermark  $b$ .

In the sequel, we note by  $\mathcal{E}(M)$  ( $\mathcal{E}(b)$ ), the homomorphic encrypted version of the model  $M$  (the watermark  $b$ ) where all its parameters (components) are encrypted element-wise by an FHE, respectively. To embed the watermark  $b$  in the model  $M$  from its encrypted version  $\mathcal{E}(M)$  without decrypting it. And as stated previously, the first step consists of defining where we want to insert  $b$  in  $M$ . To do so, we define  $Ext(\mathcal{E}(M), K_{ext})$  as the function that secretly returns, based on the secret key  $K_{ext}$ , the encrypted parameters  $\mathcal{E}(w)$  in which the watermark will be embedded, the  $K_{ext}$  could be the positions of the parameters selected to carry the watermark:

$$Ext(\mathcal{E}(M), K_{ext}) = \mathcal{E}(w) \quad (1)$$

The second step consists on defining  $Proj(\mathcal{E}(w), K_{proj})$ , a projection function that maps the extracted parameters into the watermark space “ $\{-1, 1\}^N$ ”. This function is parameterized by a secret key  $K_{proj}$ . Most of White-Box watermarking

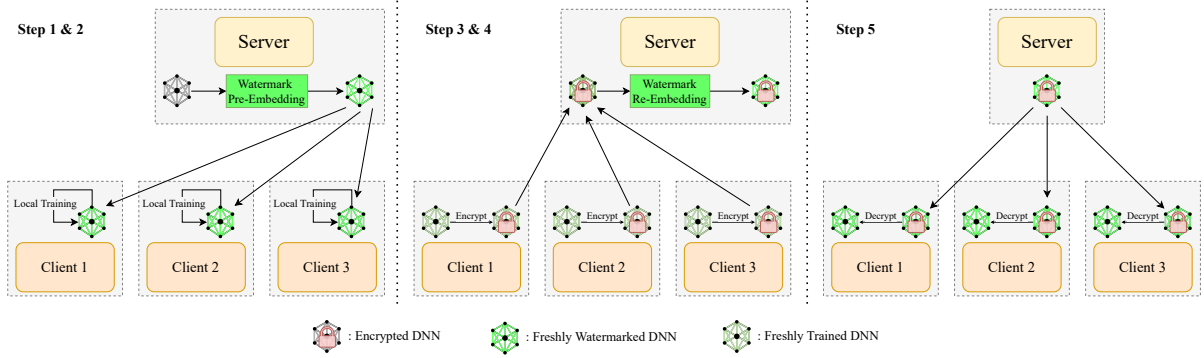


Figure 1: High-level view of the proposed method. (Step 1) The server initializes the global model, performs a pre-embedding to watermark it, and distributes it to the clients; (Step 2) Each client trains the model on its private dataset; (Step 3 & 4) Each client encrypts and sends its updated model to the server which aggregates them to get the new global model. The server then embeds the watermark using the method described in Section 3.2; (Step 5) The server distributes the global model to each client, which decrypts it; Steps 2 to 5 are repeated until the global model converges.

schemes (Uchida et al., 2017; Li et al., 2022; Shao et al., 2024) consider  $K_{proj}$  as random matrix of size  $m \times N$  where  $m = |\mathcal{E}(w)|$  is the size of the extracted parameters and  $N = |b|$  is the size of the watermark. The projection function is defined as follows:

$$\begin{aligned} Proj(\mathcal{E}(w), K_{proj}) &= \mathcal{E}(w)K_{proj} \\ &= \mathcal{E}(b^{proj}) \end{aligned} \quad (2)$$

Note that the multiplication in Equation 2 is possible because the model parameters  $\mathcal{E}(w)$  are homomorphically encrypted. Once the extraction and projection functions are defined, along with their corresponding secret keys, the embedding process involves minimizing the distance between  $b^{proj}$  and  $b$  in their homomorphically encrypted forms,  $\mathcal{E}(b^{proj})$  and  $\mathcal{E}(b)$ , using a distance metric  $d$ , given as follows:

$$E = \min_w d(\mathcal{E}(b), \mathcal{E}(b^{proj})) \quad (3)$$

In our work, we utilize the Hinge-loss (Fan et al., 2019) as a distance function. This is used to measure the distance between  $b$  and  $b^{proj}$  as follow:

$$d(\mathcal{E}(b), \mathcal{E}(b^{proj})) = \sum_{i=1}^N \text{ReLU}(\mu - \mathcal{E}(b_i)\mathcal{E}(b_i^{proj})) \quad (4)$$

Where  $\mu$  is set to 1. The ReLU (Li and Yuan, 2017) function is not polynomial and therefore cannot be efficiently computed using state-of-the-art FHE (some of the recent works on FHE focuses on solving this problem, ex. using programmable bootstrapping (Chillotti et al., 2020; Chillotti et al., 2021)). This issue is well-known in the field of secure encrypted neural networks (Chen et al., 2018) (Bellafqira et al.,

2019). The problem leveraged by ReLU (or any non-polynomial function) can be solved by approximating this latter by its  $a$ -degree polynomial approximation. In this context, we take the second-degree polynomial approximation from Gottemukkula *et al.* (Gottemukkula, 2019) which is represented in Figure 2 and is defined as:

$$\sigma(x) = 0.09x^2 + 0.5x + 0.47 \quad (5)$$

Replacing ReLU by  $\sigma$  results in the following distance for our loss term :

$$d(\mathcal{E}(b), \mathcal{E}(b^{proj})) = \sum_{i=1}^N \sigma(\mu - \mathcal{E}(b_i)\mathcal{E}(b_i^{proj})) \quad (6)$$

$\sigma$  is a good approximation for  $x \in [-5; 5]$ . To have  $x$  as close as possible to this interval, we add an  $L^2$ -norm regularisation term on the parameters  $\mathcal{E}(w)$

$$\begin{aligned} \mathcal{E}_{L^2}(w) &= \sum_{h=0}^m \mathcal{E}(w_h)^2 \\ &= \mathcal{E}\left(\sum_{h=0}^m w_h^2\right) \end{aligned} \quad (7)$$

Finally, we have the following loss function to embed the watermark:

$$\begin{aligned} \mathcal{L} &= d(\mathcal{E}(b), \mathcal{E}(b^{proj})) + \frac{\lambda}{2} \mathcal{E}_{L^2}(w) \\ &= \sum_{i=1}^N \sigma(\mu - \mathcal{E}(b)_i \mathcal{E}(b^{proj})_i) + \frac{\lambda}{2} \mathcal{E}_{L^2}(w) \end{aligned} \quad (8)$$

where  $\lambda$  is a hyper-parameter to control the impact of the  $L^2$ -norm on the model parameters (aka the weight decay parameter (Loshchilov and Hutter, 2017)).

To optimize the equation 8, and update the extracted parameters, we use the gradient descent (Du et al., 2019) which is given as:

$$\mathcal{E}(w^{t+1}) = \mathcal{E}(w^t) - \alpha \frac{\partial \mathcal{L}}{\partial w^t} \quad (9)$$

where  $\alpha$  is the learning rate and  $w^{t+1}$  represents the updated parameters from  $w^t$ . The partial derivative in equation 9 is feasible because the loss function  $\mathcal{L}$  is homomorphically encrypted.

The watermark retrieval process is straightforward and it requires the knowledge of the secret keys ( $K_{ext}$ ,  $K_{proj}$ ). Let us consider a plain-text suspicious model  $M^*$ , to extract the watermark, the first step consists of computing:

$$b^{proj} = Proj(Ext(M^*, K_{ext}), K_{proj}) \quad (10)$$

where  $b^{proj}$  is the reconstructed watermark from  $M^*$  where its components are not necessarily belongs to  $\{-1, 1\}$ . To get a watermark with the same range as the embedded watermark, we threshold  $b^{proj}$  based on the sign of its components using the function  $sgn(b^{proj}) = b^{ext}$  where

$$sgn(x) = \begin{cases} 1 & \text{if } x > 0 \\ -1 & \text{otherwise} \end{cases} \quad (11)$$

Then we can compare  $b$  and  $b^{ext}$  to evaluate the embedding. To quantify this we use a common metric for static White-Box watermarking which is the Watermark Detection Rate (WDR). This metric is expressed using the following formula

$$WDR(b, b^{ext}) = 1 - \frac{1}{N} \times H(b, b^{ext}) \quad (12)$$

where  $b$  is the message that we want to insert,  $b^{ext}$  the reconstructed binary string (see Equation 10) and  $H$  is the distance defined, in the case of two binary sequences  $a$  and  $c$  of  $N$  bits, as

$$H(a, c) = |\{i \in [1; N] | a_i \neq c_i\}| \quad (13)$$

Where  $|\cdot|$  is the cardinal of the set. If  $WDR(b, b^{ext})$  is near to 1 or 0,  $b$  and  $b^{ext}$  are highly correlated which correspond to a good embedding of  $b$  in  $M$  while  $WDR(b, b^{ext})$  close to 0.5 correspond to a wrong embedding.

### 3.3 FL Encrypted Watermarking

Performing DNN watermarking from the server side consists of embedding the watermark  $b$  just after the aggregation of the clients' updates following the extraction and projection steps described in 3.2. In the literature, we generally distinguish two steps: a pre-embedding and a re-embedding phases (Tekgul et al.,

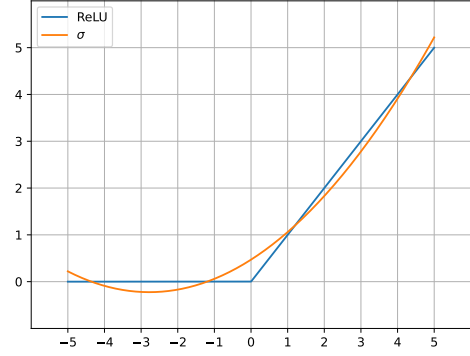


Figure 2: ReLU and its second degree approximation  $\sigma$  in the  $[-5; 5]$  interval.

2021; Li et al., 2022; Shao et al., 2024). The first one performs embedding before the training and the other one at each round. Both are based on the same function but with different hyperparameters. In our paper, they are different with regard to the use of homomorphic encryption. Following the encryption and decryption functions defined in Section 2.2, we suppose that the server owns  $Pub_{key}$ , the public encryption key, and since it is in charge of the watermark embedding it also owns the watermarking parameter set  $\theta = \{b, K_{ext}, K_{proj}\}$  with  $b$  the watermark,  $K_{ext}$  the secret extraction feature key and  $K_{proj}$  the secret projection matrix. On their side, each client only owns  $(Pub_{key}, Priv_{key})$ , i.e., the public and private encryption keys, respectively.

In the sequel, we depict the initialization of FL watermarking in the plain-text domain, i.e., our pre-embedding phase, and by next our re-embedding phase, in the context of  $K$  clients in the encrypted domain.

#### 3.3.1 FL Watermarking Initialization

The first step of this procedure consists of randomly initializing the model  $M$  to get  $M_0^G$ . Once the model is initialized one can perform the second step which aims to embed the desired binary string  $b$  directly in the un-encrypted version of  $M_0^G$  to have parameters already optimized for the watermarking embedding later. Doing so reduces the number of epochs needed for re-embedding the watermark during the FL. It is this embedding we define as the Watermark Pre-Embedding function ( $WPE(M_0^G, \theta)$ ). It simply embeds the watermark into the initialized model  $M_0^G$  following the method described in Section 3.2 using the secret parameters  $\theta$  until the watermark is embedded resulting in a watermarked model  $\hat{M}_0^G$ , where  $\hat{\cdot}$  denotes the result of a watermark embedding procedure.

Note that since the server works with the plaintext version of  $M_0^G$ , as exchanges with clients have not yet started, it has no limitation in the number of computations nor epochs performed for the embedding contrarily to the re-embedding phase, where the server will manipulate homomorphically encrypted model parameters. Thus one can perform this embedding until the watermark is perfectly embedded. The embedding is done when the watermark is perfectly embedded i.e.,  $WDR(b, b^{ext}) = 1$ . The model is then encrypted, using *Encrypt* and  $Pub_{key}$ , and the resulting encrypted model  $\mathcal{E}(\hat{M}_0^G)$  is sent to the clients.

### 3.3.2 FL Watermarking Round

In a FL round  $t$ , the server randomly selects  $C \times K$  clients from the  $K$  clients available. This set of clients is designated by  $S_t$ . Then the encrypted global model  $\mathcal{E}(\hat{M}_{t-1}^G)$  is sent to each client  $k \in S_t$  to perform local training as follow:

- *LocalTrain*( $\hat{M}_{t-1}^G, D^k$ ): Local training of the client  $k$  on his dataset  $D^k$  using the received global model  $\hat{M}_{t-1}^G$ , decrypted using *Decrypt* and  $Priv_{key}$ . Client  $k$  send back the encrypted updated model  $\mathcal{E}(M_t^k)$ .

Once all clients send their encrypted model to the server. This latter performs aggregation. The aggregation algorithm needs to be compatible with FHE. In this case, we use FedAvg (McMahan et al., 2017). This produces the encrypted global model  $\mathcal{E}(M_t^G)$  to which the server applies the re-embedding function. This step has the aim of keeping the watermark on the model since the local training can degrade the presence of the watermark. Contrary to the embedding during initialization, this embedding is performed in the encrypted domain. This increases the computation time and leads to restrictions on the number of possible updates performed as a hyper-parameter. This re-embedding function (*WRE*) is defined as follow:

- *WRE*( $\mathcal{E}(M_t^G), \theta$ ): embedded the watermark into the encrypted global model  $\mathcal{E}(M_t^G)$  following the method described in Section 3.2 using the secret parameters  $\theta$  for a fixed number of epoch. This function returns an encrypted watermarked model  $\mathcal{E}(\hat{M}_t^G)$ .

This procedure is repeated until  $t = R$  which corresponds to the end of the FL training. We define  $\hat{M}^G = \hat{M}_R^G$  the final global model with the watermark. Algorithm 1 shows the associated pseudo code.

### 3.3.3 IP Verification

The IP verification aims to determine if a plagiarized model comes from FL. Regarding the security hy-

**Input** :  $M$  the global FL model to train;  $K$  the number of clients;  $\theta$  are the set of parameters used for the watermarking;  $D^k$  local dataset with  $n_k$  samples of the client  $k$ ;  $(Pub_{key}, Priv_{key})$  the public and private encryption keys, respectively.

**Output**: Final watermarked model  $\hat{M}_R^G$

```

1  $M_0^G \leftarrow \text{Initialize}(M)$ ;
2  $\hat{M}_0^G \leftarrow \text{WPE}(M_0^G, \theta)$ ;
3 for each round  $t=1, \dots, R$  do
4    $S_t \leftarrow$  Randomly select  $C \times K$  clients;
5   for each client  $k \in S_t$  do
6      $\hat{M}_{t-1}^G \leftarrow \text{Decrypt}(\mathcal{E}(\hat{M}_{t-1}^G), Priv_{key})$ ;
7      $M_t^k \leftarrow \text{LocalTrain}(\hat{M}_{t-1}^G, D^k)$ ;
8      $\mathcal{E}(M_t^k) \leftarrow \text{Encrypt}(M_t^k, Pub_{key})$ ;
9   end
10   $\mathcal{E}(M_t^G) \leftarrow \sum_{k \in S_t} \frac{n_k}{\sum_{k \in S_t} n_k} \mathcal{E}(M_t^k)$ ;
11   $\mathcal{E}(\hat{M}_t^G) \leftarrow \text{WRE}(\mathcal{E}(M_t^G), \theta)$ ;
12 end
    
```

Algorithm 1: Proposed FL Watermarking technique with FHE.

pothesis, this leak can come from the clients during the FL or after when the model is deployed. To resolve this issue, a third honest party is called to get  $\theta$  from the server. This one computes  $sgn(b^{proj})$  from the decrypted suspect model  $M^*$ ;  $b^{proj}$  is defined as:

$$b^{proj} = Proj(Ext(M^*, K_{ext}), K_{proj}) \quad (14)$$

Then, if  $WDR(b, sgn(b^{proj})) \geq \mathcal{T}$ , where  $\mathcal{T}$  is the threshold from which the IP is verified, then  $M^*$  comes from the federation. This decision is simplified by the following function

$$\text{Verify}(M^*, \theta) = \begin{cases} 1 & \text{if } WDR(b, sgn(b^{proj})) \geq \mathcal{T} \\ 0 & \text{otherwise} \end{cases} \quad (15)$$

## 4 EXPERIMENTAL RESULTS

The experimental section is divided into two configurations:

**Real FHE**: The first one is conducted using a real Fully Homomorphic Encryption (FHE) library. For computational efficiency, we employ a one-layer neural network with a small message size  $b$  ( $N = 16$ ). This configuration evaluates the impact of the watermarking on the model performance (fidelity) using a subset of the public dataset MNIST. The primary aim

of this part is to demonstrate a proof of concept for our watermark modulation on homomorphically encrypted parameters.

**Simulated FHE:** The second configuration is conducted in plaintext, allowing greater flexibility regarding the number of tests and the size of  $b$  ( $N = 256$ ). We evaluate, using ResNet18 and the public dataset CIFAR10, the performance differences for both the main task and the watermark embedding, depending on the use of polynomial approximation. Additionally, we evaluate the watermark’s robustness against attacks such as fine-tuning and pruning.

**FL Setting:** In both experimental setups, we set the number of clients to  $K = 10$ . At each round, half of the clients are randomly selected by the server to contribute to the aggregation process ( $C = 0.5$ ). Additionally, each client performs five local epochs of training on their IID dataset.

**Metrics:** Model performance is evaluated using  $Acc$ , which denotes accuracy over the testing set, while WDR is used to measure the effectiveness of the watermark embedding. The accepted threshold for the watermark,  $\mathcal{T}$ , is set at 0.98, and the allowable difference between the un-watermarked and watermarked model is set at  $\epsilon = 0.05$ .

For readability purposes, the details of both settings are summarized in Table 2.

## 4.1 Real FHE Training

In this subsection, we evaluate the performance of our watermarking modulation in terms of effectiveness, fidelity, and efficiency, while also providing details on the computational overhead complexity. We use a simple one-layer neural network and the CKKS scheme as an FHE cryptosystem from TenSEAL.

### 4.1.1 Fidelity & Effectiveness

Fidelity requires that the performance of the model with a watermark ( $\hat{M}^G$ ) should be as close as possible to that of the model without a watermark ( $M^G$ ). We assess this by comparing the accuracy of  $\hat{M}^G$  with  $M^G$  using the test set  $D^{test}$ . If the difference in accuracy is less than the tolerated error  $\epsilon$ , then the watermarking technique satisfies the fidelity requirement.

Effectiveness involves ensuring that the watermark modulation properly embeds the watermark into the model. This is verified by comparing the original watermark  $b$  with the watermark extracted ( $b^{ext}$ ) from the final model  $M^G$ . The effectiveness is quantified by the Watermark Detection Rate (WDR, see Equation 12) and compared against a predefined threshold  $\mathcal{T}$  (as defined in Table 2). The WDR must exceed

the threshold to certify the intellectual property protection of the model.

These requirements can be summarized as follows:

1.  $Verify(\hat{M}^G, \theta) = 1$
2.  $|Acc(\hat{M}^G, D^{test}) - Acc(M^G, D^{test})| < \epsilon$

Table 3 presents the test accuracy and the WDR between  $M^G$  and  $\hat{M}^G$ . The first requirement is consistently met by  $\hat{M}^G$ , as for all test runs,  $WDR(\hat{M}^G, \theta) = 1.00$ , which is above the threshold  $\mathcal{T} = 0.98$ . From this data, we observe that the watermark has a negligible impact on performance, as evidenced by  $|Acc(\hat{M}^G, D^{test}) - Acc(M^G, D^{test})| = 0$ , which is less than  $\epsilon$ .

### 4.1.2 Efficiency: Computational Overhead

As clearly defined by Tekgul *et al.* (Tekgul et al., 2021) the watermark embedding should not increase the communication and computation complexity overhead. Our method does not increase the communication overhead compared to a FL with HE as a privacy mechanism. However, we increase the computation in the  $WRE$  (i.e., Watermark Re-Embedding function) step. To perform one epoch for the embedding,  $WRE$  computes exactly one forward pass to compute  $b^{proj}$  and one backward pass over the loss (Equation 8).

$$\mathcal{L} = \sum_{n=1}^N \sigma(\mu - b_i \mathcal{E}(b_i^{proj})) + \lambda \sum_{h=0}^m \mathcal{E}(w_h)^2 \quad (16)$$

Regarding the matrix multiplications performed during the backward pass, the computation complexity is  $O(2N) + O(2MN)$  homomorphic multiplication in the worst case where  $N$  is the size of the watermark  $b$  and  $M$  the number of rows in  $K_{proj}$ .

## 4.2 Simulated FHE Training

In this subsection, we evaluate the robustness of the proposed watermarking modulation against the fine-tuning and pruning attacks. We evaluate as well the impact of the approximation of the ReLU function and verify the robustness of our watermark. Due to the use of HE, which seriously increases computation time, these experiments were conducted using a simulated framework, that is to say, a framework where computations are conducted on clear values but with exactly the same calculations in clear as if HE was used.



Table 2: Experimental settings used for each configuration.

	Simulated FHE	Real FHE
Dataset	CIFAR10 (Krizhevsky et al., 2009)	Binary classification of "3" and "8" with MNIST (LeCun and Cortes, 2010)
Watermark Size	$N = 256$	$N = 16$
Model	ResNet18	One-layer neural network
FL Setting	$C = 0.5$ clients are selected among 10 clients at each round. 5 epochs are performed by the selected clients. The distribution is I.I.D.	
Metrics	$Acc$ is used for the main task accuracy while $WDR$ is used for the watermark. The accepted threshold for the watermark recovery is defined by $\mathcal{T} = 0.98$ . The accepted difference between the un-watermarked model and the watermarked one is fixed to $\epsilon = 5e - 2$ .	

Table 3: Comparison of the test accuracy and WDR for  $M^G$  and  $\hat{M}^G$ . Values shown are mean and their corresponding standard deviation computed over 5 runs.

	Accuracy	WDR
	Test	$b$
With Watermark	$0.97 \pm 9e - 4$	$1.00 \pm 0.00$
Without Watermark	$0.97 \pm 2e - 3$	$0.46 \pm 0.10$

Table 4: Comparison of the test accuracy and WDR according to the use of an approximation for the embedding. Show values are mean and their corresponding standard deviation computed over 5 runs.

	Accuracy	WDR
	Test	$b$
ReLU	$0.81 \pm 1e - 3$	$1.00 \pm 0.00$
$\sigma \& R$	$0.81 \pm 2e - 3$	$1.00 \pm 0.00$

#### 4.2.1 Impact of the Approximation

The error generated by the ReLU approximation in the distance function  $d$  (see Equation 4) needs to be quantified experimentally. Table 4 shows the difference between both approaches. The first line shows the loss function that uses ReLU (corresponding to Equation 4 and denominated as "ReLU" in the Table 4). The other one corresponds to the loss function that uses the approximation defined in Equation 8 and is denominated by " $\sigma + R$ " in the Table 4. From this result, we can see that using the approximation does not affect both the main task and the watermark.

#### 4.2.2 Robustness

The watermark should be robust against various types of attacks. The attacker can try to remove the watermark using techniques such as fine-tuning and pruning. The primary goal of the attacker is to derive a surrogate model that does not contain the watermark. At the same time, he seeks to maintain the performance of the attacked model on the main task. These two goals need to be reached with a significantly lower

Table 5: Fine-tuning of a watermarked model during 100 epochs with the whole dataset.

	Accuracy	WDR
Epoch	Test	$b$
0	0.81	1.00
25	0.82	1.00
50	0.83	1.00
75	0.84	1.00
100	0.84	1.00

complexity than training the model from scratch or having access to all the training data, which is difficult to achieve, especially in a federated context. In other words, the attacker aims to build a model  $M^A$  from  $\hat{M}^G$  following the two conditions

- $Verify(M^A, \theta) = 0$
- $|Acc(M^A, D^{test}) - Acc(\hat{M}^G, D^{test})| < \epsilon$

**Fine-Tuning.** The fine-tuning attack consists of continuing to perform the training algorithm with a train set to erase the watermark. In this tests, we fine-tuned  $\hat{M}^G$  in centralized training during 100 epochs assuming that the attacker has access to the whole training dataset; a very strong hypothesis. The learning rate was fixed to  $1e - 2$ . As we can see from Table 5, performing fine-tuning cannot degrade the detection of the watermark.

**Pruning** Pruning (Blalock et al., 2020) is a common method used to erase the watermark since the parameters used during the embedding are often not the same for the main task. In this experiment, we used the pruning method that consists of zeroing out the parameters with the lowest  $L^1$ -norm. As we can see from Figure 3 the WDR is equal to 1.00 until 90% of parameters are pruned. These results demonstrate that our watermark is resistant to pruning since the IP of the model can be verified for any percentage of parameters pruned.

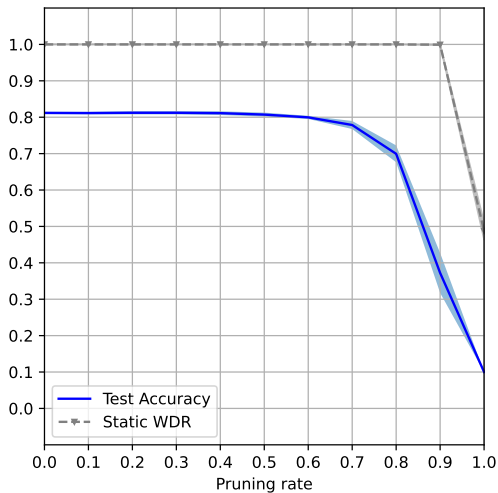


Figure 3: Test accuracy and WDR according to the pruning rate. Values shown are mean and their corresponding standard deviation computed over 5 runs.

## 5 DISCUSSION & LIMITATION OF THE STUDY

Our watermarking modulation has the same performance in terms of robustness compared to the one performed without HE in FedTracker (Shao et al., 2024) and FedIPR (Li et al., 2022). However, it is well known in the literature that this type of technique is not resistant to some attacks such as overwriting (Darvish Rouhani et al., 2019), Privacy Inference Attack (PIA) (Wang and Kerschbaum, 2019), and ambiguity attack (Kapusta et al., 2024; Chen et al., 2023b). Our method proves the feasibility of White-Box watermarking in this specific context but other White-Box watermarking techniques (e.g. (Nie and Lu, 2024; Lv et al., 2023)) can be adopted following the same proposed methodology. As mentioned before, the main issue remains in the non-polynomial operations. They can be approximated by an  $a$ -degree polynomial but the overhead can grow exponentially with  $a$ , so it is a compromise to be found according to the application’s needs.

As discussed before, the main issue with FHE cryptosystems is their computation overhead. Each computation, such as vector-matrix multiplication performed in Equation 2, can grow exponentially the embedding time. For the watermarking embedding, this overhead mainly comes from the sizes of  $b$  and  $w$ . Inserting a larger binary string increases the number of columns for  $K_{proj}$  while choosing a larger  $w$  increases the rows of  $K_{proj}$ . On the other hand, the main advantage of this modulation is the fact that we embed the watermark in one targeted layer (using  $K_{ext}$ )

which allows us to perform the embedding without considering the depth of the model.

In Section 4, we fixed the number of epochs performed in  $WRE$  to two because the White-Box watermarking is sufficiently robust to resist against the learning process (as shown in Section 4.2.2) and a few steps of the re-embedding process are enough to preserve the watermark during the FL. However, it is clear that even if the server is in charge of embedding the watermark, it cannot use Verify (Equation 15) since the reconstructed watermark  $\mathcal{E}(b^{proj})$  is encrypted. Moreover, he cannot perform early-stopping or increase the number of epochs if the watermark is not fully embedded since the embedding is blind.

## 6 CONCLUSION

In this work, we demonstrate the feasibility of watermarking an encrypted neural network in Homomorphic FL. We first demonstrate how we can apply White-Box watermarking to encrypted parameters and shift this method to Homomorphic FL. Our experiments demonstrate the capability of our method using a TenSeal, a concrete FHE library. However, current FL White-Box watermarking approaches have known limitations, such as their vulnerability to overwriting, and extensive research needs to be done to design a new HE-compatible White-Box watermarking technique that is more robust to such attacks. Our next work will investigate how to apply the most recent White-Box watermarking scheme with HE while minimizing the computation overhead of the homomorphic encryption.

## ACKNOWLEDGMENTS

This work is funded by the European Union under Grant Agreement 101070222. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Commission (granting authority). Neither the European Union nor the granting authority can be held responsible for them. Additionally, this work is supported by the CYBAILE industrial chair, which is leads by Inserm with the support of the Brittany Region Council, as well as by French government grants managed by the Agence Nationale de la Recherche under the France 2030 program, bearing the references ANR-22- PESN-0006 (PEPR digital health TracIA project).

## REFERENCES

- Bas, P., Furon, T., Cayre, F., Doërr, G., Mathon, B., Bas, P., Furon, T., Cayre, F., Doërr, G., and Mathon, B. (2016). A quick tour of watermarking techniques. *Watermarking Security*, pages 13–31.
- Baumstark, P., Monschein, D., and Waldhorst, O. P. (2023). Secure plaintext acquisition of homomorphically encrypted results for remote processing. In *2023 IEEE 48th Conference on Local Computer Networks (LCN)*, pages 1–4. IEEE.
- Bellafqira, R. and Coatrieux, G. (2022). Diction: Dynamic robust white box watermarking scheme. *arXiv preprint arXiv:2210.15745*.
- Bellafqira, R., Coatrieux, G., Genin, E., and Cozic, M. (2019). Secure multilayer perceptron based on homomorphic encryption. In *Digital Forensics and Watermarking: 17th International Workshop, IWDW 2018, Jeju Island, Korea, October 22-24, 2018, Proceedings 17*, pages 322–336. Springer.
- Benaissa, A., Retiat, B., Ceber, B., and Belfedhal, A. E. (2021). Tenseal: A library for encrypted tensor operations using homomorphic encryption. *arXiv preprint arXiv:2104.03152*.
- Blalock, D., Gonzalez Ortiz, J. J., Frankle, J., and Gutttag, J. (2020). What is the state of neural network pruning? *Proceedings of machine learning and systems*, 2:129–146.
- Boenisch, F. (2021). A systematic review on model watermarking for neural networks. *Frontiers in big Data*, 4:729663.
- Bouslimi, D., Bellafqira, R., and Coatrieux, G. (2016). Data hiding in homomorphically encrypted medical images for verifying their reliability in both encrypted and spatial domains. In *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 2496–2499. IEEE.
- Brakerski, Z., Gentry, C., and Vaikuntanathan, V. (2014). (leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)*, 6(3):1–36.
- Chen, H., Gilad-Bachrach, R., Han, K., Huang, Z., Jalali, A., Laine, K., and Lauter, K. (2018). Logistic regression over encrypted data from fully homomorphic encryption. *BMC medical genomics*, 11:3–12.
- Chen, J., Li, M., and Zheng, H. (2023a). Fedright: An effective model copyright protection for federated learning. *arXiv preprint arXiv:2303.10399*.
- Chen, Y., Tian, J., Chen, X., and Zhou, J. (2023b). Effective ambiguity attack against passport-based dnn intellectual property protection schemes through fully connected layer substitution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8123–8132.
- Cheon, J. H., Kim, A., Kim, M., and Song, Y. (2017). Homomorphic encryption for arithmetic of approximate numbers. In *Advances in Cryptology—ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I 23*, pages 409–437. Springer.
- Chillotti, I., Gama, N., Georgieva, M., and Izabachène, M. (2020). Tfhe: fast fully homomorphic encryption over the torus. *Journal of Cryptology*, 33(1):34–91.
- Chillotti, I., Joye, M., and Paillier, P. (2021). Programmable bootstrapping enables efficient homomorphic inference of deep neural networks. In *Cyber Security Cryptography and Machine Learning: 5th International Symposium, CSCML 2021, Be'er Sheva, Israel, July 8–9, 2021, Proceedings 5*, pages 1–19. Springer.
- Clet, P.-E., Stan, O., and Zuber, M. (2021). Bfv, ckks, tfhe: Which one is the best for a secure neural network evaluation in the cloud? In *Applied Cryptography and Network Security Workshops: ACNS 2021 Satellite Workshops, AIBlock, AIHWS, AIoT, CIMSS, Cloud S&P, SCI, SecMT, and SiMLA, Kamakura, Japan, June 21–24, 2021, Proceedings*, pages 279–300. Springer.
- Darvish Rouhani, 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*, pages 485–497.
- Du, S., Lee, J., Li, H., Wang, L., and Zhai, X. (2019). Gradient descent finds global minima of deep neural networks. In *International conference on machine learning*, pages 1675–1685. PMLR.
- Fan, J. and Vercauteren, F. (2012). Somewhat practical fully homomorphic encryption. *Cryptology ePrint Archive*.
- Fan, L., Ng, K. W., and Chan, C. S. (2019). Rethinking deep neural network ownership verification: Embedding passports to defeat ambiguity attacks. *Advances in neural information processing systems*, 32.
- Gottemukkula, V. (2019). Polynomial activation functions.
- He, Z., Zhang, T., and Lee, R. B. (2019). Model inversion attacks against collaborative inference. In *Proceedings of the 35th Annual Computer Security Applications Conference*, pages 148–162.
- Hu, H., Salcic, Z., Sun, L., Dobbie, G., Yu, P. S., and Zhang, X. (2022). Membership inference attacks on machine learning: A survey. *ACM Computing Surveys (CSUR)*, 54(11s):1–37.
- Kallas, K. and Furon, T. (2023). Mixer: Dnn watermarking using image mixup. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5.
- Kapusta, K., Mattioli, L., Addad, B., and Lansari, M. (2024). Protecting ownership rights of ml models using watermarking in the light of adversarial attacks. *AI and Ethics*, pages 1–9.
- Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images.
- Lansari, M., Bellafqira, R., Kapusta, K., Thouvenot, V., Bettan, O., and Coatrieux, G. (2023). When federated learning meets watermarking: A comprehensive overview of techniques for intellectual property protection. *Machine Learning and Knowledge Extraction*, 5(4):1382–1406.

- LeCun, Y. and Cortes, C. (2010). MNIST handwritten digit database.
- Li, B., Fan, L., Gu, H., Li, J., and Yang, Q. (2022). Fedipr: Ownership verification for federated deep neural network models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.
- Li, F.-Q., Wang, S.-L., and Liew, A. W.-C. (2021a). Towards practical watermark for deep neural networks in federated learning. *arXiv preprint arXiv:2105.03167*.
- Li, Y., Wang, H., and Barni, M. (2021b). A survey of deep neural network watermarking techniques. *Neurocomputing*, 461:171–193.
- Li, Y. and Yuan, Y. (2017). Convergence analysis of two-layer neural networks with relu activation. *Advances in neural information processing systems*, 30.
- Liang, J. and Wang, R. (2023). Fedcip: Federated client intellectual property protection with traitor tracking. *arXiv preprint arXiv:2306.01356*.
- Liu, X., Shao, S., Yang, Y., Wu, K., Yang, W., and Fang, H. (2021). Secure federated learning model verification: A client-side backdoor triggered watermarking scheme. In *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 2414–2419. IEEE.
- Loshchilov, I. and Hutter, F. (2017). Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.
- Lukas, N., Jiang, E., Li, X., and Kerschbaum, F. (2022). Sok: How robust is image classification deep neural network watermarking? In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 787–804. IEEE.
- Lv, P., Li, P., Zhang, S., Chen, K., Liang, R., Ma, H., Zhao, Y., and Li, Y. (2023). A robustness-assured white-box watermark in neural networks. *IEEE Transactions on Dependable and Secure Computing*.
- McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. (2017). Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR.
- Miao, Y., Liu, Z., Li, H., Choo, K.-K. R., and Deng, R. H. (2022). Privacy-preserving byzantine-robust federated learning via blockchain systems. *IEEE Transactions on Information Forensics and Security*, 17:2848–2861.
- Nie, H. and Lu, S. (2024). Fedcrmw: Federated model ownership verification with compression-resistant model watermarking. *Expert Systems with Applications*, 249:123776.
- NVIDIA (2023). NVFLARE: NVIDIA Federated Learning Application Runtime Environment. Accessed: 2023-04-21.
- Paillier, P. (1999). Public-key cryptosystems based on composite degree residuosity classes. In *International conference on the theory and applications of cryptographic techniques*, pages 223–238. Springer.
- Pierre, F., Guillaume, C., Teddy, F., and Matthijs, D. (2023). Functional invariants to watermark large transformers. *arXiv preprint arXiv:2310.11446*.
- Shao, S., Yang, W., Gu, H., Qin, Z., Fan, L., and Yang, Q. (2024). Fedtracker: Furnishing ownership verification and traceability for federated learning model. *IEEE Transactions on Dependable and Secure Computing*.
- Shokri, R., Stronati, M., Song, C., and Shmatikov, V. (2017). Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE.
- Sun, Y., Liu, T., Hu, P., Liao, Q., Ji, S., Yu, N., Guo, D., and Liu, L. (2023). Deep intellectual property: A survey. *arXiv preprint arXiv:2304.14613*.
- Tekgul, B. A., Xia, Y., Marchal, S., and Asokan, N. (2021). Waffle: Watermarking in federated learning. In *2021 40th International Symposium on Reliable Distributed Systems (SRDS)*, pages 310–320, Los Alamitos, CA, USA. IEEE Computer Society.
- 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*, pages 269–277.
- Wang, B., Chen, Y., Li, F., Song, J., Lu, R., Duan, P., and Tian, Z. (2024). Privacy-preserving convolutional neural network classification scheme with multiple keys. *IEEE Transactions on Services Computing*.
- Wang, T. and Kerschbaum, F. (2019). Attacks on digital watermarks for deep neural networks. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2622–2626. IEEE.
- Xiong, R., Ren, W., Zhao, S., He, J., Ren, Y., Choo, K.-K. R., and Min, G. (2024). Copifl: A collusion-resistant and privacy-preserving federated learning crowdsourcing scheme using blockchain and homomorphic encryption. *Future Generation Computer Systems*, 156:95–104.
- Xue, M., Wang, J., and Liu, W. (2021). Dnn intellectual property protection: Taxonomy, attacks and evaluations. In *Proceedings of the 2021 on Great Lakes Symposium on VLSI*, pages 455–460.
- Yang, W., Shao, S., Yang, Y., Liu, X., Xia, Z., Schaefer, G., and Fang, H. (2022). Watermarking in secure federated learning: A verification framework based on client-side backdooring. *arXiv preprint arXiv:2211.07138*.
- Yang, W., Yin, Y., Zhu, G., Gu, H., Fan, L., Cao, X., and Yang, Q. (2023). Fedzkip: Federated model ownership verification with zero-knowledge proof. *arXiv preprint arXiv:2305.04507*.
- Yu, S., Hong, J., Zeng, Y., Wang, F., Jia, R., and Zhou, J. (2023). Who leaked the model? tracking ip infringers in accountable federated learning. *arXiv preprint arXiv:2312.03205*.
- Zhang, C., Li, S., Xia, J., Wang, W., Yan, F., and Liu, Y. (2020). {BatchCrypt}: Efficient homomorphic encryption for {Cross-Silo} federated learning. In *2020 USENIX annual technical conference (USENIX ATC 20)*, pages 493–506.