Securing Intellectual Property in Federated Learning

Mohammed Lansari, Reda Bellafqira, Katarzyna Kapusta, Vincent Thouvenot, Olivier Bettan, Gouenou Coatrieux

Securing Intellectual Property in Federated Learning

Mohammed Lansari*†, Reda Bellafqira*, Katarzyna Kapusta†,
Vincent Thouvenot†, Olivier Bettan†, Gouenou Coatrieux*

* Inserm UMR 1101, IMT Atlantique
Brest, France
name.surname@imt-atlantique.fr

† ThereSIS, Thales SIX GTS
Palaiseau, France
name.surname@thalesgroup.com

***Abstract*—Federated Learning (FL) is a technique that allows multiple participants to collaboratively train a Deep Neural Network (DNN) without the need to centralize their data and therefore comes with privacy-preserving properties making it attractive for application in sensitive contexts. However, it requires sharing participant models during the training process which makes them vulnerable to theft or unauthorized distribution by malicious actors. To address the issue of ownership rights protection in the context of Machine Learning (ML), DNN Watermarking methods have been developed during the last five years. Most existing works have focused on watermarking in a centralized manner, but only a few methods have been designed for FL and its unique constraints. In this paper, we provide an overview of recent advancements in Federated Learning watermarking, shedding light on the new challenges and opportunities that arise in this field.**

***Index Terms*—ML Watermarking, Federated Learning, Intellectual Property, Security.**

I. Introduction

Sophisticated ML models require good and non-synthetic data that are often hard and costly to acquire. Government regulations, such as the European General Data Protection Regulation, have made the training process even harder by requiring to preserve privacy of the data subjects. Thus, companies and institutions have to face the issue of lack of data on the one hand and the complexity of usage of sensitive data on the other. Indeed, very often the data exists but is distributed over multiple locations and/or belongs to different owners. Centralizing it would be very costly in terms of bandwidth. Moreover, it is almost impossible if the decentralized data is private or classified.

Federated Learning [1] is a promising solution that allows multiple data owners (or distributed agents belonging to one data owner) to train a global Deep Neural Network (DNN) without directly sharing their data. It opens new possibilities in domains such as health or military, as it allows to train a model over multiple distributed datasets without centralizing the local data. Combined with privacy-preserving techniques, such as Homomorphic Encryption, Multi-Party Computation, or Differential Privacy, it can ensure a high level of protection of sensitive data. In the civil domain, it may be applied by a consortium of hospitals, where each of the hospitals possesses its own data that is impossible to share due to regulation constraints. In the military domain, it could find application for instance in the case of predictive maintenance, where multiple cooperating parties would like to benefit from the information collected by others but fear about revealing classified information.

From the point of view of ownership rights protection, the complex context combining both AI and collaboration of owners and users raises new challenges. ML watermarking techniques have been proposed since 2018. They enable to identify a ML model and thus can be applied for model theft detection. However, the majority of them address the problem of local training of a single model. The collaborative setting of FL raises new challenges in the context of ownership protection. How to identify that a participant contributed to the training of a model resulting from FL? How to be sure that the final model will not be misused by the aggregation party that coordinates the FL? These issues may slow down parties from joining the learning consortium.

II. Outline

We start with a brief presentation of the FL and DNN watermarking concepts in Section III. Then, we proceed with a definition of watermarking for FL in Section IV followed by an overview of the six existing works combining FL and DNN. In Section V, we analyze the different elements that have to be taken into consideration while designing a watermarking scheme for collaborative training settings. We identify the unsolved challenges and thus give an insight into possible future works.

III. Background

In this section, we remind the definition of FL and give a brief overview of its different existing settings. Then we introduce DNN watermarking.

A. Federated Learning

FL is a ML setting where $K \in \mathbb{N}^*$ entities called clients collaborate to train a global model M_G while keeping the training data $D_C = \{D_{C_i}\}_{i=1}^{K}$ decentralized [2].

The most popular framework is called Client-Server FL. In this setting, the server train M_G by receiving and aggregating the client models weights $W_C = \{W_{C_i}\}_{i=1}^{K}$ (see e.g. [3]). However, the presence of the server is not mandatory to perform FL. In a decentralized setting, clients can perform FL without the supervision of a server. BrainTorrent [4] proposes a server-less and peer-to-peer Federated framework. In this solution, a random client is selected to be the aggregator. Then, it checks if other clients have an updated version of the model. If yes, they send it to the aggregator who performs an averaging of the model weights. Then it updates its own model with the previous result.

The FL setting is also characterized by the features partition. We distinguish three types of partition: horizontal, vertical, and hybrid FL. In horizontal FL, all clients have the same features but not the same samples. On the other hand, vertical FL assumes that all clients have the same samples but not the same features. Finally, hybrid FL supposes that samples and features are different from one client to another.

In contrast with previous settings, Split-Learning [5] consists of splitting the DNN between the server and the clients. Many configurations are possible but the most common for client privacy is the U-shaped configuration. The aim is that each client has the first and last layers of the DNN and the server has the rest of the layers. In such a way, clients perform the forward/backward pass keeping their data (inputs and labels) private, and send/receive only the activations/gradients to update the whole model.

B. DNN Watermarking

DNN watermarking is a promising solution for the ownership protection of ML models. Inspired by image watermarking, it consists of introducing a secret change into the model parameters or behavior during its training, in order to enable its identification in the future. As image watermarking, DNN watermarking must respect some requirements to be effective for IP protection. Table I summarizes these requirements. In general, a watermarking technique needs to preserve the performances on the main task (**Fidelity**), while providing a large insertion **Capacity** (enabling multiple verifications) and a strong **Robustness** against watermarking removal techniques [6].

DNN Watermarking can be distinguished into two types of techniques: White-Box [7] [8] [9] [10] and Black-Box [8] [11] [12] [13] [14] watermarking. Each technique is defined by the type of access to the model during the verification process.

In the White-Box setting, we assume that the owner will have full access to the model (architecture, weights, activations, etc.). In this way, to insert a watermark into a DNN, the owner will hide a vector of bits b into the model's parameters or activations. One of the first proposed method in [7], uses a regularization term to embed b with a secret key S into the model's parameters. Several techniques have been published to meet the associated challenges. DeepSigns [8] proposes to use the probability density function of the output for selected layers to embed the information. Tartaglione *et al.* [9] defines a strategy that consists in embedding the watermark before the training and then training the model while adding a constraint that penalizes the model for small perturbations on the watermarked parameters.

On the other hand, Black-Box setting assumes that the owner can perform the verification process only through an API: he can interact with the model only by giving inputs and receiving associated predictions. Knowing that the owner watermarks the model by changing its behavior. The common technique consists training of the model using a trigger set $T = (X_i, Y_i)_{i=1}$, which is composed of crafted inputs X_i with their associated outputs Y_i [11]. Zhang *et al.* [12] propose to use the same technique but with different types of inputs. In addition to unrelated images, they use training examples with two types of trigger: random noise and a textual pattern. The trigger set can also be built using adversarial examples [13] or a filter as a mapping function [14].

To evaluate the performances of the watermarking embedding, White-Box setting relies on the Binary-Error-Rate between the reconstructed message $\tilde{b}$ and the original one b. In Black-Box watermarking, we evaluate the accuracy of the model on the trigger set T.

IV. Watermarking for Federated Learning

In this section, we introduce and define what is watermarking for FL including the different possible scenarios. Then, we formulate requirements for watermarking deployed in a FL context. Finally, we analyze the six state-of-the-art methods.

A. Definition

In centralized DNN watermarking, the goal is to simply prove the model's ownership after the training process. In FL, ownership rights protection becomes a more complex problem due to the presence of multiple participants and multiple exchanges between them that have to be taken into account during the threat model formulation. To illustrate this issue, [15] shows that existing methods can naively be applied to FL in two manners. The first one consists of watermarking the model after the training. For example, by using fine-tuning to embed the watermark into the model. Without taking the fidelity requirement into account, any participant who receives the model (client or server) can steal the DNN before the last round. The second way is to embed the watermark before the training. Even if the watermark will resist during the first rounds, it will be removed after several aggregation rounds. Thus, it is important to design a specific watermarking technique for FL that will be persistent from the first round to the model deployment.

We define Watermarking for FL as the process for a participant or multiple participants to insert a watermark into the shared model.

Following the client-server FL framework, the first question is to determine which part of the federation can watermark the model. Is the server more to be trusted since it manages the federation? Or the clients since their data are used? During this study, we distinguish four watermarking scenarios for

Fidelity	The watermarked model needs to have the same performances compared to the model without watermark
Capacity	The capacity of a technique to embed multiple watermarks
Generality	The capacity of a watermarking technique to be applied independently of the architecture of the model
Efficiency	The performance cost generated by the embedding and verification process of the watermarking
Robustness	The capacity to resist against attacks aiming at removing the watermark
Secrecy	The watermark should be secret and undetectable

TABLE I: DNN Watermarking requirements

centralized FL, which we illustrate in Figure 1 according to who is watermarking the model :

- (**S$_1$**) **Server** : The server is in charge of watermarking the global model.
- (**S$_2$**) **Clients** : One or multiple clients among the federation watermark their updates to spread that are embedded into the global model.
- (**S$_3$**) **Server and clients** : The server and the clients collaborate to watermark the global model together.
- (**S$_4$**) **Clients in a decentralized context** : Only clients collaborate to watermark the global model together (decentralized FL).

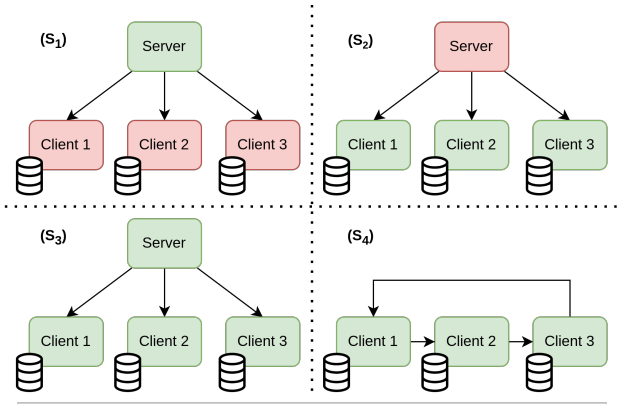

Fig. 1: An example of each possible scenario of watermarking in FL with one server and three clients. Green rectangles are the participants who follow the same watermarking procedure together. Red rectangles are those who are not enrolled in the watermarking procedure.

All watermarking requirements defined in Table I are also true in the federated context. However, due to the new constraints and the extension to several participants, we can add precision to three of them:

1) **Capacity :** When multiple clients want to insert their own message b_{C_i}, the watermarking technique needs to avoid possible conflict between the different inserted b_{C_i}. The number of bits needs to be enough.
2) **Generality :** In a real FL scenario, many additional mechanisms are added for security and privacy such as robust aggregation functions (Section V-B) or Differential Privacy [16] (Section V-E). The watermarking technique must be applied independently to these mechanisms.
3) **Efficiency :** The cost generated by the embedding process is more crucial in FL. For example, in a cross-device architecture, clients have low computation power and they cannot perform many operations. The watermarking techniques must take this parameter into account.

B. Related works

1) WAFFLE: WAFFLE [15] is the first DNN Watermarking technique for FL. In this solution, the server embeds the watermark (**S$_1$**) using a black-box watermarking technique using a trigger set. Any trigger set that does not need any knowledge concerning the clients' data can be used but the authors present a specific set that is more suitable for FL: the WAFFLEPattern. WAFFLEPattern is defined as a set of images containing random patterns with a noisy background. Basically, the server will embed the watermark into the global model using the two following functions :

- **PreTrain** : Train an initialized model with the trigger set until
- **ReTrain** : Fine-tune (using FTAL) the model with the trigger set until

PreTrain is used to embed the watermark in the model before the first round. During each round, after the aggregation process, the server uses **ReTrain** to re-embed the watermark into the model.

2) FedIPR: FedIPR [17] is both a Black-Box and White-Box technique. This technique allows all clients to embed their own watermark in the global model (**S$_2$**) without sharing secret information. Each watermark can be described as follow :

- **Black-Box Watermark** : Each client generates a trigger set using Projected Gradient Descent technique in a small CNN trained with the local data.
- **White-Box Watermark** : Each client generates a random secret matrix and a location in the Batch-Normalisation layers to embed its message.

Both White-Box and Black-box watermarks are inserted using an additional loss during the local training. For **Black-Box Watermark**, the loss is exactly the same as the loss used for the main task but with a batch of the trigger set as input. For the **White-Box Watermark**, the loss used is a Hinge-like loss between the original message and the reconstructed message.

3) FedTracker: FedTracker [18] is a watermarking technique that allows the server to embed a global Black-box watermark (**S$_1$**) but also a White-box watermark specific to each client. Each watermark can be described as follow :

- **Global Black-Box Watermark** : A trigger set is generated using the WAFFLEPattern method [15].
- **White-Box Watermark** : Server generates a random secret matrix and a fingerprint for each client.

After the aggregation, the server embeds the **Global Black-Box Watermark** using the intuition of Continual Learning [19] to avoid forgetting the main task. Then, for the **White-Box Watermark**, the loss used is a Hinge-like loss between the original message b and the reconstructed message $\tilde{b}$.

4) Client-side Black-box watermarking: Liu *et al.* [20] propose a client-side Black-box watermarking scheme (**S_2**). This technique is designed to embed a watermark only from one client. The latter creates a trigger set composed of Gaussian noise images with a given label as a trigger set. Then, the client's model will over-fit with this set like in [11]. To tackle the fact that this particular client will probably not be selected at each round, the authors introduce a scaling factor

$$\lambda = \frac{N}{n},$$

where N is the number of clients and n is the number of clients selected at each round. The client will then send its model weights multiplied by λ. According to the authors, this will be approximately equivalent to the case that this client is selected every iteration and the watermark will be embedded more easily.

5) Merkle-Sign: Merkle-Sign [21] is a framework focusing on ownership verification in a collaborative Clients-Server setting (**S_3**). The authors propose a public verification protocol that uses the Merkle-tree [22]. In this framework, the server use at each round an embedding function to insert two identity information (i.e keys) into M_G : one that identify the server and the other one the client that will receives the model. In parallel, the server uploads the tuple of keys and the tuple of verification function (which are generated by the watermarking embedding function) into a Merkle-tree with the recording time. At the final round, the server embeds also all keys generated by the clients into M_G and updates the Merkle-tree. This framework is also compatible in a Peer-to-Peer context. The associated Black-Box watermarking schema relies on the training of an Auto-Encoder [23] (AE) for each client. Then the server averages the received AE to obtain a final AE from which it can generate a trigger set using the keys as input for the decoder part.

6) FedRight: FedRight [24] is a solution for the server to fingerprint the model in the FL framework (**S_3**). DNN fingerprinting is a process in which instead of embedding a watermark in the model, we extract a fingerprint to identify this model [25]. To do so, the server generates adversarial examples from a set of inputs (key samples). Then the server extracts the probability distribution of each prediction and feeds it to a detector with the key samples target. Then, during the verification process, this detector is used to predict whether the corresponding model is the good one or not.

V. Discussion

In this section, we identify and discuss specific challenges related to watermarking in FL. In particular, we evaluate how existing methods deal or not with these new challenges.

A. Black-Box watermarking in the Server side

Black-Box watermarking consists in changing the behavior of the model. To do so, most methods let the model overfit on the trigger set by adding a regularization term in the loss function. Usual DNN watermarking techniques can easily be applied in **S_2**, **S_3** and **S_4**. However, it is not so easy in **S_1**. When the client C_k wants to watermark its model M_{C_k}, he can rely on two things :

1) Have an access to its private dataset D_{C_k}
2) Train the model on both the main task dataset D_{C_k} and its trigger set T_{C_k} at the same time

A large number of Black-Box watermarking techniques need to build T_{C_k} using $D_{C_k} = (X_i, Y_i)_{i=1}$ (as discussed in III-B). Using such techniques is motivated by the fact that training the model from these datasets is a multi-task learning. Building T_{C_k} from D_{C_k} helps to reduce the negative impact on learning from two domains. Moreover, learning these two tasks together avoids "catastrophic forgetting" [26].

In **S_1**, since the server does not have its own dataset, it cannot perform such type of watermarking. The choice is limited by using unrelated or noise-based inputs as WafflePattern [15].

The problem is that this limitation leads to an exposure to evasion attacks. During the Black-Box verification process, the owner will ask the suspicious Application Programming Interface (API) that possibly contains his DNN. However, the attacker may evade this verification using a query detector [27]. Since the trigger-set is built using images that are qualified to be Out-Of-Distribution (OOD), this implies an easier detection for the attacker [28]. WAFFLE [15] authors confirm the intuition that the performances of such detector depends a lot on the data quantity and capacities of the attacker.

B. Aggregation functions

The most common aggregation function is `FedAvg` [1] which consists on averaging clients' weights after they perform multiple epochs on mini-batches. Each client weight matrix is multiplied by a scaling factor defined as $\frac{n_{C_k}}{n}$ where n_{C_k} is the number of samples in D_{C_k} and $n = \sum_k^K n_{C_k}$. Many aggregation functions emerged to meet various challenges in FL. Since the clients do not necessarily know which aggregation function the server is using, the proposed methods must be independent of this parameter.

For the Byzantine-attacks problem in which one or multiple clients try to disturb the FL process. These attacks can be simple noise weights or complex label-flipping backdoors. To

leverage this problem, multiple aggregation functions appear to select only benign updates such as `Krum` [29] `Trim-mean` or `Bulyan` [30]. Since clients' watermarking techniques are sensitive to the embed message b and the trigger set T, they keep their updates far from each other. A part of updates can be rejected for this reason if we use defensive aggregation techniques. As an example, FedIPR shows that the **White-Box Watermark** results are similar to `FedAvg` with a detection rate of 97.5% using `Trim-mean`. However, the **Black-Box Watermark** reaches only 63.25% of the watermark detection rate at the end of the FL process. Even if this score is enough to detect plagiarism, using a defensive aggregation function has a huge impact on the watermark. Liu et at. [20] have not tested yet their solution with a defensive aggregation function but we can guess that multiplying weights by a so big scaling factor λ can be easy to detect for `Krum` as a Byzantine attack as shown in similar example [31].

Another problem is that `FedAvg` performs well when the data are statistically homogeneously distributed among the clients. However, in real use cases, data are heterogeneous which may lead to difficulty for the model to converge to the global minimum or diverge using `FedAvg`. Existing watermarking techniques for FL have not evaluated methods that tackle this problem such as `FedProx` [32], `FedNova` [33] or `SCAFFOLD` [34]. If we want to use the proposed solution in a real Secure FL framework, these methods need to be tested in such a context.

C. Client selection

For communication efficiency and when the number of clients is too high, we introduce a client selection. To do so, we simply randomly select cK clients with $c \in]0, 1[$. This simple mechanism can have a big effect on the watermarking. In ($\mathbf{S_1}$), this effect should be insignificant, since the server embeds the watermark at each round regardless of cK. Unfortunately, WAFFLE and FedTracker have not tested this effect. On the other hand, ($\mathbf{S_2}$) is more able to be sensitive to the client selection process. Since each client wants to insert its watermark, the watermark of not selected clients risks to be degraded or removed in the global model. Authors of FedIPR show that for $c > 0.2$ the detection rate for both White-Box and Black-Box watermark is near to 100%. When $c \leq 0.2$ the detection rate associated with a feature-based watermark is still near 100%. However, the detection rate for the backdoor falls to 62%.

D. Cross-device setting

All proposed papers are treating the case in which we have a small amount of clients. The worse scenario is tested in Merkle-Sign in which 200 clients are enrolled in the federation. However, there is no solution that takes into account the cross-device setting. In this setting, a large number of clients (up to 10^{10} devices), are enrolled in the FL procedure [2]. These clients are not always reachable and they have a low dedicated computational power which is defined as a performance condition by the authors of WAFFLE.

In the Black-Box setting, the problem can come from the low computation power that does not allow the client to perform more computations to increase the batch size using trigger-set methods. For the White-Box setting, the bottleneck would be the **Capacity** as mentioned in Section IV-A. In particular in cases where the proposed methods are tested using Normalization layers such as FedIPR or FedTracker which limits the overall embedding capacity. It leads to a difficulty for each client to embed its vector b without conflicting in face of other clients' watermarks.

E. Differential Privacy

Since FL keeps the client's data private, DNNs are shared between the server and the clients or directly between clients. The model himself can give to an attacker private information in such a way that he can identify the presence of an exact data point (such an attack is called membership inference) [35]. To tackle this problem, we can use Differential Privacy (DP) which is a strong standard for providing privacy guarantees for algorithms operating on aggregate databases [16]. In FL, a usual DP technique consists of adding a Gaussian random noise to the gradients sent to the aggregator. Among related works, FedIPR is the only article for which the proposed method is tested with a DP mechanism. The presented results show that adding a small noise to the weights does not affect a lot the watermarking accuracy.

F. Homomorphic Encryption

In Client-Server FL, the server has access to all client updates. Then, it can use this privilege to try to learn information about the private datasets. A cryptographic solution to prevent such misuse from the server is, for the clients, to protect the model using Homomorphic Encryption (HE). By using this technique, clients can easily cipher their updates using public keys. Then the server will perform the aggregation, in general using `FedAvg`, in the cipher space. Then, when clients receive the updated global model, they use their private key to decipher the model weights.

In the context of watermarking, the use of HE fits perfectly for ($\mathbf{S_2}$) since the clients can access their model weights to watermark the model. However, so far no solution has been provided for ($\mathbf{S_1}$) given the fact that the server cannot decipher the model weights to perform watermarking embedding.

G. Watermarking for Non Client-Server framework

Decentralized FL ($\mathbf{S_2}$) is an interesting framework in which clients do not need a server to perform the model aggregation. The proposed methods seem to be applicable to this setting since watermarking the model from the client side does not require the server. However, Merkle-Sign is the unique solution that extends to the decentralized setting ($\mathbf{S_4}$). We can also cite Split-Learning in which the server has a part of the network and clients have another part. Performing White-Box watermarking as in [7] can be more difficult for the server or clients. In both cases, they have access to a part of the model weights that can be arbitrarily small. In the U-shape

Split Learning architecture, the server has only the middle part of the model and the client has the first and last layers. In this setting, performing a Black-Box watermarking on the server side is hard since it cannot use its inputs and labels on the model.

H. Attacks from clients and server

When we analyze ($\mathbf{S_1}$) and ($\mathbf{S_2}$) scenarios, we can see that each one has different parameters to play with whether for watermarking or disrupt it. The server can control the selected participants or how to aggregate the weights. It has also sometimes a clear view of clients' weights at each round. However, it does not have data and it cannot fully control if clients are strictly following the training process. On the other hand, clients have their private dataset and they can send the weights that they want. Nevertheless, they have no control over what happens with their updates in the server level.

If the server wants to avoid a subset of clients to watermark the model, it can use methods proposed for Byzantine attacks [36] detection. In particular, attacks that consist of multiplying the weights by a scaling factor to replace or have a bigger impact in the global model are easy to detect [31]. The proposed method by Liu *et al.* [20] is then easily removable and the global model will not be watermarked. A solution to catch backdoor-ed models was presented in [37] [38]. Then all proposed solutions that rely on a backdoor-based watermarking can be rejected.

Clients can also try to disturb the watermarking process. FedIPR ($\mathbf{S_2}$) [17] authors present the free-riders problem in which some clients do not contribute to the training of M_G and the watermarking process. Even if with their solution, they have no important impact on the watermarking, no testing has yet been done on ($\mathbf{S_1}$). Another attack that is specific to FL as described in FedRight [24] and WAFFLE [15] consists of the fact that multiple clients will use their models and private datasets. As mentioned in Section V-A, evasion attack works better when using multiple datasets to train the detector. But it is also possible to fine-tune the model with the combined dataset.

VI. Conclusion

Watermarking for FL is taking great interest since classical DNN watermarking cannot be naively applied in a collaborative context. In particular, constraints such as data distribution, new distributed threat models, and additional security mechanisms have to be taken into account while designing an efficient solution for collaborative ML watermarking. This paper provides a comprehensive overview of existing methods and exposes the different problems they face off. Several of the analyzed methods have tried their model against each Secure FL mechanisms separately. Unfortunately, none of them has tried to test their solutions in a complete and realistic Secure FL framework, including defensive aggregation functions, non I.I.D data, DP, and HE in the same experiment.

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[2] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[3] H. Brendan McMahan, E. Moore, D. Ramage, S. Hampson, and B. Agüera y Arcas, "Communication-efficient learning of deep networks from decentralized data," *arXiv e-prints*, pp. arXiv–1602, 2016.

[4] A. G. Roy, S. Siddiqui, S. Pölsterl, N. Navab, and C. Wachinger, "Braintorrent: A peer-to-peer environment for decentralized federated learning," *arXiv preprint arXiv:1905.06731*, 2019.

[5] P. Vepakomma, O. Gupta, T. Swedish, and R. Raskar, "Split learning for health: Distributed deep learning without sharing raw patient data," *arXiv preprint arXiv:1812.00564*, 2018.

[6] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao, "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 707–723.

[7] Y. Uchida, Y. Nagai, S. Sakazawa, and S. Satoh, "Embedding watermarks into deep neural networks," in *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval*, 2017, pp. 269–277.

[8] B. Darvish Rouhani, H. Chen, and F. Koushanfar, "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*, 2019, pp. 485–497.

[9] E. Tartaglione, M. Grangetto, D. Cavagnino, and M. Botta, "Delving in the loss landscape to embed robust watermarks into neural networks," in *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 2021, pp. 1243–1250.

[10] Y. Li, B. Tondi, and M. Barni, "Spread-transform dither modulation watermarking of deep neural network," *Journal of Information Security and Applications*, vol. 63, p. 103004, 2021.

[11] Y. Adi, C. Baum, M. Cisse, B. Pinkas, and J. Keshet, "Turning your weakness into a strength: Watermarking deep neural networks by backdooring," in *27th USENIX Security Symposium*, 2018, pp. 1615–1631.

[12] J. Zhang, Z. Gu, J. Jang, H. Wu, M. P. Stoecklin, H. Huang, and I. Molloy, "Protecting intellectual property of deep neural networks with watermarking," in *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, 2018, pp. 159–172.

[13] E. Le Merrer, P. Perez, and G. Trédan, "Adversarial frontier stitching for remote neural network watermarking," *Neural Computing and Applications*, vol. 32, pp. 9233–9244, 2020.

[14] J. Guo and M. Potkonjak, "Watermarking deep neural networks for embedded systems," in *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2018, pp. 1–8.

[15] B. G. Tekgul, Y. Xia, S. Marchal, and N. Asokan, "Waffle: Watermarking in federated learning," in *2021 40th International Symposium on Reliable Distributed Systems (SRDS)*. IEEE, 2021, pp. 310–320.

[16] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016, pp. 308–318.

[17] B. Li, L. Fan, H. Gu, J. Li, and Q. Yang, "Fedipr: Ownership verification for federated deep neural network models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[18] S. Shao, W. Yang, H. Gu, J. Lou, Z. Qin, L. Fan, Q. Yang, and K. Ren, "Fedtracker: Furnishing ownership verification and traceability for federated learning model," *arXiv preprint arXiv:2211.07160*, 2022.

[19] M. De Lange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars, "A continual learning survey: Defying forgetting in classification tasks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 7, pp. 3366–3385, 2021.

[20] X. Liu, S. Shao, Y. Yang, K. Wu, W. Yang, and H. Fang, "Secure federated learning model verification: A client-side backdoor triggered watermarking scheme," in *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2021, pp. 2414–2419.

[21] F.-Q. Li, S.-L. Wang, and A. W.-C. Liew, "Towards practical watermark for deep neural networks in federated learning," *arXiv preprint arXiv:2105.03167*, 2021.

[22] G. Becker, "Merkle signature schemes, merkle trees and their cryptanalysis," *Ruhr-University Bochum, Tech. Rep*, vol. 12, p. 19, 2008.

[23] D. Bank, N. Koenigstein, and R. Giryes, "Autoencoders," *arXiv preprint arXiv:2003.05991*, 2020.

[24] J. Chen, M. Li, and H. Zheng, "Fedright: An effective model copyright protection for federated learning," *arXiv preprint arXiv:2303.10399*, 2023.

[25] X. Cao, J. Jia, and N. Z. Gong, "Ipguard: Protecting intellectual property of deep neural networks via fingerprinting the classification boundary," in *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*, 2021, pp. 14–25.

[26] R. M. French, "Catastrophic forgetting in connectionist networks," *Trends in cognitive sciences*, vol. 3, no. 4, pp. 128–135, 1999.

[27] D. Hitaj and L. V. Mancini, "Have you stolen my model? evasion attacks against deep neural network watermarking techniques," *arXiv preprint arXiv:1809.00615*, 2018.

[28] J. Yang, K. Zhou, Y. Li, and Z. Liu, "Generalized out-of-distribution detection: A survey," *arXiv preprint arXiv:2110.11334*, 2021.

[29] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," *Advances in neural information processing systems*, vol. 30, 2017.

[30] R. Guerraoui, S. Rouault *et al.*, "The hidden vulnerability of distributed learning in byzantium," in *International Conference on Machine Learning*. PMLR, 2018, pp. 3521–3530.

[31] Z. Gu and Y. Yang, "Detecting malicious model updates from federated learning on conditional variational autoencoder," in *2021 IEEE international parallel and distributed processing symposium (IPDPS)*. IEEE, 2021, pp. 671–680.

[32] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *Proceedings of Machine learning and systems*, vol. 2, pp. 429–450, 2020.

[33] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor, "Tackling the objective inconsistency problem in heterogeneous federated optimization," *Advances in neural information processing systems*, vol. 33, pp. 7611–7623, 2020.

[34] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "Scaffold: Stochastic controlled averaging for federated learning," in *International Conference on Machine Learning*. PMLR, 2020, pp. 5132–5143.

[35] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *2019 IEEE symposium on security and privacy (SP)*. IEEE, 2019, pp. 691–706.

[36] J. Shi, W. Wan, S. Hu, J. Lu, and L. Y. Zhang, "Challenges and approaches for mitigating byzantine attacks in federated learning," in *2022 IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. IEEE, 2022, pp. 139–146.

[37] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu, "Data poisoning attacks against federated learning systems," in *Computer Security–ESORICS 2020: 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14–18, 2020, Proceedings, Part I 25*. Springer, 2020, pp. 480–501.

[38] B. Xi, S. Li, J. Li, H. Liu, H. Liu, and H. Zhu, "Batfl: Backdoor detection on federated learning in e-health," in *2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS)*. IEEE, 2021, pp. 1–10.