When Federated Learning meets Watermarking: A Comprehensive Overview of Techniques for Intellectual Property Protection

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 of centralizing their data. Among other advantages, it comes with privacy-preserving properties making it attractive for application in sensitive contexts, such as health care or the military. Although the data are not explicitly exchanged, the training procedure requires sharing information about participants' models. This makes the individual models 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*—DNN Watermarking, Federated Learning, Intellectual Property, Security.**

I. Introduction

In the fast-paced digital era, machine learning (ML) has emerged as a revolutionary technology that drives innovation across various industries. Its ability to analyze vast amounts of data and make accurate predictions has opened up a world of possibilities. From enabling personalized recommendations in e-commerce [1] and improving medical diagnoses [2] to enhancing autonomous vehicles [3], the applications of machine learning are both diverse and impactful.

The effectiveness of machine learning models depends on the availability of large, high-quality datasets. Traditionally, the success of ML algorithms relied on centralized data storage and processing, where organizations and institutions accumulated vast amounts of information to train their models effectively. Unfortunately, this approach increases the risk of privacy leakage, as sensitive information about individuals, customers, or proprietary processes may inadvertently be exposed to unauthorized entities. This presents a roadblock to the advancement of machine learning, as data holders are understandably hesitant to compromise their users' privacy and organizational security. Moreover, safeguarding the privacy and security of sensitive data is not just an ethical obligation but a legal necessity as highlighted by various data protection regulations around the world, such as the General Data Protection Regulation (GDPR) [4] in EU, California Consumer Privacy Act (CCPA) [5] in US, and Data Security Law (DSL) [6] in China.

Addressing these challenges, Federated Learning (FL) [7] emerges as a promising solution. FL enables multiple data owners or distributed agents to collectively train a global model without directly sharing their private data. Incorporating privacy-preserving techniques like Homomorphic Encryption [8], Multi-Party Computation [9], Private set intersection (PSI) [10], Differential Privacy [11], [12], or hardware-based solutions such as trusted execution environment (TEE) [13], FL ensures data privacy and security while enabling collaborative model training. The significance of federated learning lies not only in its ability to alleviate privacy concerns but also in its potential to unlock the untapped value of distributed data. By enabling cooperation among organizations and institutions, federated learning allows for more comprehensive and representative models, ultimately leading to more accurate predictions and valuable insights [14].

Another critical aspect to take into account is the protection of the intellectual property of the trained model, aimed at preventing theft, plagiarism, and unauthorized usage by external parties. In response to this concern, model watermarking has been introduced since 2017 as a solution to embed a unique watermark or fingerprint into the model. This measure effectively safeguards the model's intellectual

‡Equal contributions.

property and enables tracing the source in case of any illicit model leakage. However, the majority of model watermarking methods address the problem of local or centralized training. The collaborative setting of Federated Learning raises new challenges in the context of ownership protection. How to identify that a participant contributed to the training of a model resulting from Federated Learning? How to be sure that the final model will not be misused by the aggregation party that coordinates the Federated Learning? These issues may slow down parties from joining the learning consortium. Previous work by Yang *et al.* [15] has conducted a survey on federated watermarking solutions, primarily focusing on security issues within Federated Learning. However, the paper only presents two watermarking algorithms (Waffle [16] and FEDIPR [17]) specifically designed for the context of FL.

A. *Contributions*

Our paper makes the following main contributions:

- We emphasize the foundational principle of secure federated learning and stress the importance of comprehensive security guarantees that align with the trust levels of the participants. This includes addressing various aspects such as training on non Independently and Identically Distributed (non-I.I.D) data and ensuring robustness against poisoning and backdooring attacks.
- We conduct a comprehensive examination of the challenges concerning model watermarking in the context of Federated Learning.
- We expand upon the work presented in [15] by providing an in-depth analysis of nine currently available DNN watermarking schemes in the FL context, along with their limitations.
- We bring attention to unresolved issues that necessitate further exploration.

B. *Organization*

The rest of the paper is organized as follows. Section II describes background information on Federated Learning and DNN watermarking concept. Section III presents our definition of watermarking for Federated Learning followed by an overview of the six existing works combining FL and DNN. In Section IV, we analyze then the different elements that have to be taken into consideration while designing a watermarking scheme for a collaborative training setting. We identify the unsolved challenges and thus give an insight into possible future works. Section V concludes the paper.

II. Background

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

A. *Federated Learning*

Federated Learning is a machine learning 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 [18].

The most popular framework is called Client-Server FL or scatter & gather framework. Here's a high-level overview of this setting:

1) Setup:
   - Central Server: A central server manages the overall training process and initiates model updates.
   - Clients: These are the individual devices, such as smartphones, IoT devices, or computers, that participate in the training process. Each edge device has its local dataset that cannot be shared with the central server due to privacy concerns.
2) Initialization:
   - Initially, the central server initiates a global model M_G (e.g. with random parameters) and distributes it to a subset of clients selected randomly at each round.
3) Local Training:
   - Each client trains the global model M_G on its local dataset using its computational resources. The training is typically performed using gradient descent or a similar optimization algorithm.
4) Model Update:
   - After the local training is complete, the clients generate a model update (typically gradients) based on the locally processed data.
5) Aggregation:
   - The clients send their model updates back to the central server without sharing their raw data.
   - The central server aggregates all the received model updates to create a refined global model. This is usually done by averaging the model updates (see e.g. [19]).
6) Iterative Process:
   - Steps 3 to 5 are repeated for multiple rounds or epochs, allowing the global model to improve over time by leveraging knowledge from various clients.
7) Centralized Model Deployment:
   - Once the federated training process is complete, the final global model can be deployed from the central server to all clients for local inference.

Note that, the presence of the server is not mandatory to perform FL. In a decentralized setting also known as a cyclic framework, clients can perform FL without the supervision of a server. BrainTorrent [20] 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/updates. Then it updates its own model with the previous result.

Another type of decentralized learning is Split-Learning [30] which consists of splitting the DNN model 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

FL frameworks	Developed by	Purpose	Security protocols provided
Fed-BioMed [21]	INRIA	Research	DP, HE
TensorFlowFederated [22]	Google	Research	DP
PySyft [23]	OpenMined	Research	MPC, DP, HE, PSI
Flower [24]	Flower Labs GmbH	Industrial	DP
FATE [25]	WeBank	Industrial	HE, DP, MPC
OpenFL [26]	Intel	Industrial	TEE
IBM Federated Learning [27]	IBM	Industrial	DP, MPC
NvFlare [28]	Nvidia	Industrial	HE, DP, PSI
Clara [29]	Nvidia	Industrial	DP, HE, TEE

TABLE I: Examples of Open-source Federated Learning (FL) frameworks

the DNN model 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 activation maps/gradients, respectively, to update the whole model.

The FL is also defined by the data features partitioning, which can be categorized into three types: horizontal, vertical, and hybrid FL. In the case of horizontal FL, all clients share the same set of features, but their sample data may differ. In contrast, vertical FL assumes that all clients possess the same sample data but have distinct features. Lastly, hybrid FL involves clients with varying samples and features across the board.

Ideally, the data should be Independently and Identically Distributed (I.I.D) for each client. However, in real-world FL scenarios, data distribution and amount differ between clients since they collect and use their own data. This leads to a non-I.I.D data partition, which can result in significant performance loss [31]. To address this issue, several aggregation functions have been proposed, such as FedProx [32] and SCAFFOLD [33]. These methods aim to improve the performance of federated learning despite the non-I.I.D data distribution across clients.

The aim of federated learning is to ensure the security and privacy of clients' data while achieving a model's performance equivalent to centralized training. However, this objective can be compromised if the server or certain clients act maliciously. In real scenarios, the server may be considered an honest but curious entity, meaning it will abide by the FL protocol regarding client selection and global model aggregation/distribution, but it will attempt to infer information about the client data. This situation commonly arises in classical federated learning setups, where only model parameters/gradients are shared. The server can potentially employ attacks like the inversion attack, which reconstructs the data used for learning from the gradients shared by the clients. To counteract this type of attack, various countermeasures have been developed, one of which is homomorphic encryption (HE) [8]. Homomorphic encryption enables linear operations to be performed on data from their encrypted form (without having access to the secret key), preventing the server from conducting inversion attacks on encrypted gradients. However, using HE has some drawbacks, including high computational and communication complexity, and it also restricts the range of aggregation methods, as it only allows linear operations to be performed on the encrypted data.

Another approach to address privacy concerns is Differential Privacy, which combats inversion attacks by introducing noise to the updated gradients [11], [12]. Nevertheless, this technique faces a limitation due to the trade-off between utility and privacy. Alternatively, Trusted Execution Environments (TEEs) [13], [34] present another solution, where a trusted third party is utilized to provide guarantees for code and data confidentiality and integrity. Multi-party computation (MPC) has also been implemented to achieve secure model aggregation in federated learning, as demonstrated in [13], [34], [35]. However, similar to encryption-based solutions, MPC-based approaches are susceptible to efficiency issues. In addition to reinforcing the confidentiality of the aggregated data, MPC - and more precisely its private set intersection protocol [10] - can be applied to identify the intersection of feature spaces between clients in vertically partitioned data.

In scenarios where a group of clients consists of malicious users, unlike the server, these clients have access to their local data and models. Leveraging this knowledge, they can employ various attacks to undermine the model's performance. Some of these attacks include poisoning attacks [36], [37], which aim to introduce malicious data to corrupt the model's training or attacks that cause the model to misclassify data with specific patterns while maintaining its performance on the primary task, known as backdooring attacks [38], [39]. These malicious actions pose significant challenges to the security and integrity of the federated learning process.

To defend against poisoning and backdooring attacks in the context of federated learning, two classes of approaches have been proposed. The first class involves developing robust aggregation techniques, such as Krum aggregation [40] or median and trimmed mean aggregation [41], which are designed to identify and remove abnormal local models contributed by potentially malicious clients. These techniques help improve the overall model's resilience to attacks. The second class of defenses relies on the use of anomaly detection techniques implemented by the server at each round of federated learning. These anomaly detection methods enable the server to identify and filter out abnormal client updates before performing the model aggregation process [42]–[44]. By doing so, the server can mitigate the impact of potential malicious clients and enhance the security of the federated learning system. For more comprehensive insights into the threats and defenses related to federated learning, interested readers can refer to the work cited in [37].

To summarize, to establish a secure federated learning (FL) environment, the first step is to conduct a security analysis of the entities engaged in FL and assess their level of trust. This analysis helps identify potential threats and risks specific to the scenario being considered. Once the security analysis is complete, the next step involves integrating the security tools discussed earlier to design a robust FL model that performs effectively while ensuring data and model privacy. In Table I, we present a list of secure federated learning frameworks, along with the corresponding security tools that are employed to provide the necessary security measures. Furthermore, when developing an efficient and practical watermarking solution for federated learning, it is crucial to consider all the aforementioned security requirements. We elaborate on how these constraints can be accommodated in the context of federated learning watermarking in Section IV.

B. DNN Watermarking

1) Requirements: DNN watermarking is a promising solution for ownership protection of ML models [45]–[50]. Inspired by multimedia and database watermarking [51]–[55], it consists in introducing a secret change into the model parameters or behavior during its training, in order to enable its identification in the future. As multimedia content watermarking, DNN watermarking must respect some requirements to be effective for IP protection. Table II summarizes these requirements.

The watermark must be secret (Secrecy). This requirement refers to the fact that any person who analyzes the model is not able to detect if the latter is watermarked. White-Box techniques can change the distribution of the parameters and then differs from a non-watermarked model. In addition, a watermarking technique must also preserve the model's performance on the main task (Fidelity). If the watermarking embedding process returns a model that has an accuracy up to a defined ϵ compared to a non-watermarked model, then the watermarking technique is not efficient. Reliability ensures a low false negative rate for the owner during IP verification, while Integrity aims to prevent false positive claims by other parties. The watermarking algorithm should be independent of the model (Generality) while providing a large insertion Capacity, which can be either zero-bit, indicating only the presence of a watermark, or multi-bit, allowing the encoding of multiple bits of information.

Furthermore, the watermark must exhibit Robustness against attacks aimed at removing or detecting it. A removal attack is considered effective if it maintains a high test accuracy while eliminating the watermark. It is efficient if the resources required for the attack, such as runtime, are relatively small compared to retraining the model from scratch. Some common attacks include:

- Pruning Attack: Setting the less useful weights of the model to zero.
- Fine-Tuning Attack: Re-training the model and updating its weights without decreasing accuracy.
- Overwriting Attack: Embedding a new watermark to replace the original one.
- Wang and Kerschbaum Attack: For static white-box watermarking algorithms, it alters the weight distribution of the watermarked model, relying on visual inspection.
- Property Inference Attack: Training a discriminating model to distinguish watermarked from non-watermarked models, thereby detecting if a protected model is no longer watermarked.
- Another attack is the Ambiguity Attack, which forges a new watermark on a model, making it challenging for external entities, like legal authorities, to determine the legitimate watermark owner. This ambiguity prevents the legitimate owner from claiming the copyright of the intellectual property.

2) Related works: DNN Watermarking can be distinguished into two types of techniques: White-Box [56]–[62], [62]–[69] and Black-Box [63], [70]–[83] watermarking. Each technique is defined by the type of access to the model parameters during the verification process.

In the White-Box setting, we assume that the owner will have full access to the model (architecture, parameters, activation maps...). In this way, to insert a watermark into a DNN, the owner will hide a piece of information b in the form of a binary string or an image (e.g. Quick Response code) into the model's parameters [57], [69], [84], activation maps [63], [65], [66] or by adding a passport layer [64], [85]. As formulated in [66], a white box watermarking scheme is defined as follows:

1) Initially, a target model M is considered, and a features extraction function $Ext(M, K_{ext})$ is applied with a secret key K_{ext}. The features obtained can be a subset of the model weights, where K_{ext} indicates the indices of the selected weights. Alternatively, the features can be model activation maps for specific input data secretly chosen from a trigger set. These features are then utilized for watermark insertion and extraction.
2) The embedding of a watermark message b involves regularizing M using a specific regularization term E_{wat}. This regularization term ensures that the projection function $Proj(., K_{Proj})$ applied to the selected features encodes the watermark b in a predetermined watermark space, which depends on the secret key K_{Proj}. The goal is to achieve the following after training:

$$Proj(Ext(M^{wat}, K_{ext}), K_{Proj}) = b \qquad (1)$$

where M^{wat} is the watermarked version of the target model M. To achieve this, the watermarking regularization term E_{wat} relies on a distance measure d defined in the watermark space. For example, in the case of a binary watermark with a binary string of length l, i.e., $b \in \{0,1\}^l$, the distance measure could be the Hamming distance, Hinge distance, or Cross-Entropy.

Fidelity	The watermarked model needs to have performances as close as possible compared to the model without watermark
Capacity	The capacity of a technique to embed multiple watermarks
Reliability	Demonstrate a low false negative rate, enabling legitimate users to accurately identify their intellectual property with a high level of certainty.
Integrity	Demonstrate a low false positive rate, preventing unjustly accusing honest parties with similar models of intellectual property theft.
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 II: DNN Watermarking requirements and properties

The regularization term is formulated as:

$$E_{wat} = d(Proj(Ext(M^{wat}, K_{ext}), K_{Proj}), b) \quad (2)$$

To preserve the accuracy of the target model, the watermarked model M^{wat} is usually derived from M through a fine-tuning operation parameterized with the following loss function:

$$E = E_0(X_{Train}, Y_{Train}) + \lambda E_{wat} \quad (3)$$

where $E_0(X_{Train}, Y_{Train})$ represents the original loss function of the network, which is essential to ensure good performance in the classification task. E_{wat} is the regularization term added to facilitate proper watermark extraction, and λ is a parameter that adjusts the trade-off between the original loss term and the regularization term.

3) The watermark retrieval process is relatively straightforward. It involves using both the features extraction function $Ext(., K_{ext})$ and the projection function $Proj(., K_{Proj})$ as follows:

$$b^{ext} = Proj(Ext(M^{wat}, K_{ext}), K_{Proj}) \quad (4)$$

where b^{ext} is the extracted message from the watermarked model M^{wat}.

For example, in the watermarking scheme introduced by Uchida *et al.* [57], the feature extraction function $Ext(., K_{ext})$ involves computing the mean value of secretly selected filter weights $\mathbf{w}$, where K_{ext} represents the index of the chosen convolutional layer. The projection function $Proj(., K_{Proj})$ in [57] is designed to insert a watermark $b \in \{0,1\}^l$, where l is the length of the watermark, and it is defined as follows:

$$Proj(\mathbf{w}, K_{Proj}) = \sigma(\mathbf{w}K_{Proj}) \in \{0,1\}^l \quad (5)$$

Here, K_{Proj} represents a secret random matrix of size $(|w|, l)$, and $\sigma(.)$ is the Sigmoid function:

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

Uchida *et al.* [57] use binary cross entropy as the distance measure d for the watermarking regularization E_{wat}, given by:

$$d(b, y) = -\sum_{j=1}^{l} (b_j \log(y_j) + (1 - b_j)\log(1 - y_j)) \quad (6)$$

With the information on d, $Proj(., K_{Proj})$, and $Ext(., K_{ext})$, the loss E_{wat} for watermarking a target model M can be computed using (2).

Another example of a watermarking scheme proposed in [86], where the feature extraction function $Ext(., K_{ext})$ consist of the scaling parameters of the Batch Normalization (BN) weights $W_\gamma = (\gamma^1, \gamma^2, ..., \gamma^l)$ (defined in Eq. (7)) with l channels is chosen according to the secret position parameter K_{ext}.

$$y^i = \gamma^i * x^i + \beta^i \quad (7)$$

where γ^i and β^i are the scaling and bias parameters in channel i for BN layer respectively, x^i is the input of the BN layer. The projection function $Proj(., K_{Proj})$ in [86] is designed to insert a watermark $b \in \{0,1\}^l$, where l is the length of the watermark, and it is defined as follows:

$$Proj(W_\gamma, K_{Proj}) = Sgn(W_\gamma K_{Proj}) \in \{0,1\}^l \quad (8)$$

where K_{Proj} is smiliar to Uchida *et al.* scheme [57], which represents a secret random matrix of size $(|w|, l)$, and $Sgn(.)$ is the sign function:

$$Sgn(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{Otherwise.} \end{cases} \quad (9)$$

The Hinge Loss is used as the distance measure d for the watermarking regularization E_{wat}, given by:

$$d(b, y) = -\sum_{j=1}^{l} \max(\mu - b_i y_i, 0) \quad (10)$$

Here, μ represents the parameter of the Hinge Loss as defined in [87]. Similar to the scheme proposed by Uchida *et al.* in [57], and utilizing the information about d, $Proj(., K_{Proj})$, and $Ext(., K_{ext})$, the watermarking loss E_{wat} for the purpose of watermarking a target model M can be computed using the equation (2).

On the other hand, the 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 [73]. During each epoch, instead of giving a batch only from the train set to the model, we give a concatenation between a batch from the train set and the trigger set.

For example, Zhang *et al.* [72] propose to use the same technique but with different types of inputs. They try three

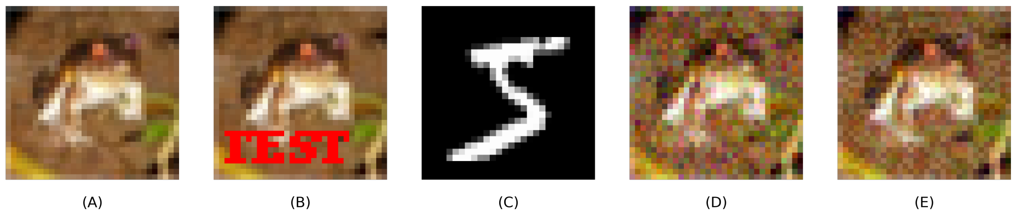

Fig. 1: Sample from possible trigger sets : (A) Original (B) Content (C) Unrelated (D) Noise (E) Adversarial

different types of images for the trigger set that are illustrated in Figure 1 :

1) Content Watermarking: Adding meaningful content to images from the train set. The model should be triggered by the content and returns the associated fixed label. In our example, the text ”TEST” is used to trigger the model.
2) Unrelated Watermarking: Images that are irrelevant from the main task of the model. Each image has an associated label (like in [73]) or each sample can have its specific output. In our example, some images from the MNIST dataset are used to trigger the model.
3) Noise Watermarking: Adding a specific noise to images from the train set. Then the model classifies any images with this specific noise as a predefined label. In our example, we add a small Gaussian noise to trigger the model.

The trigger set can also be built using adversarial examples. Authors of [75] proposed to use a trigger set composed of two adversarial examples :

1) True adversaries: Samples that are miss-classified by the model while being close to being well classified.
2) False adversaries: Well-classified samples from which we add an adversarial perturbation without changing their classification.

Then the model is trained to well classify the true adversaries with their true associated labels and the false adversaries with their labels. In Figure 1 we use Fast Gradient Sign Method [88] to generate a possible False adversary.

To evaluate the performance of BlackBox watermarking embedding, we assess the accuracy of the model's output on the trigger set T and their labels as follows:

$$acc = \frac{1}{|T|} \sum_{(x,y)\in T} \mathbb{1}_{M^{wat}(x)=y} \tag{11}$$

III. Watermarking for Federated Learning

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

A. Definition

In the context of centralized DNN watermarking, the goal is to simply prove the model's ownership subsequent to its training and release. 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, the authors in [16] show that the existing methods can naively be applied to FL in two manners: The first one consists in watermarking the model after the training is completed. For example, by using fine-tuning to embed the watermark into the trained model. Without taking the fidelity requirement into account, any participant that receives the model (client or server) can steal the DNN before the last round. The second way is to embed the watermark before starting the training process. 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 until the model deployment.

We define Watermarking for Federated Learning 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 three watermarking scenarios FL context, which we illustrate in Figure 2 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 watermark their updates in order to watermark the global model.
(S_3) **Server and Clients**: The server and the clients collaborate to watermark the global model together.

All watermarking requirements defined in Table II are also true in the federated context. However, due to the new constraints and the extension to several participants, we can add precision to five 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 then enough.

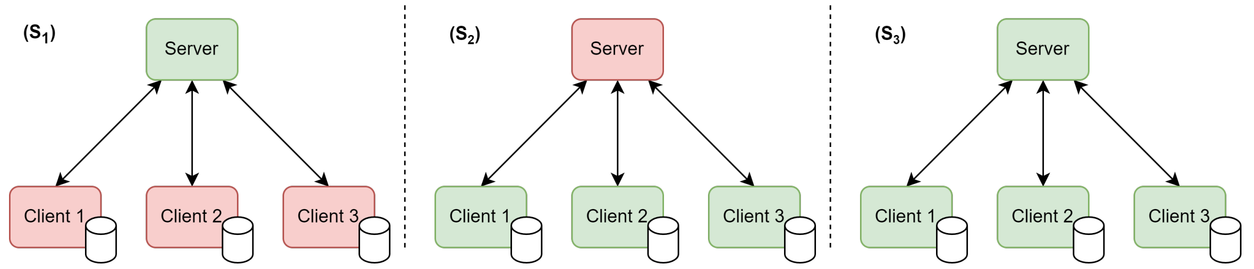

Fig. 2: An example of each possible scenario of watermarking in FL with one server and three clients. Green rectangles are the participants who follow the same watermarking procedure together. Red rectangles are those who are not enrolled in the watermarking procedure.

2) **Generality:** In a real FL scenario, many additional mechanisms are added for security and privacy such as robust aggregation functions (Section IV-B) or Differential Privacy [89] (Section IV-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.
4) **Secrecy:** If all parties are not enrolled in the watermarking process, the watermark should not be detected. In particular, if one or some clients are trying to watermark the global model, their updates need to be similar to benign updates. Otherwise, the server can use a defensive aggregation function to cancel the FL watermarking process (as described in II-A).
5) **Robustness:** Since the model can be redistributed for any clients or the server, the watermark must track who is the traitor. Traitor tracing is the fact that each actor of the federation has a unique watermark that can be used to uniquely identify the owner in addition to a global watermark.

B. Related works

In this section, we describe all state-of-the-art solutions of FL watermarking for IP protection. Note that all following papers are watermarking algorithms except the two last which are focusing on the ownership verification protocol.

1) WAFFLE: WAFFLE [16] is the first DNN Watermarking technique for FL. In this solution. The security hypothesis presented in the paper assumes that the server is a trusted party that embeds the watermark (**S$_1$**) using a black-box watermarking technique using a trigger set. Clients cannot backdoor, poison, or embed their own watermarks since they are incentivized to maximize global accuracy. The adversary can only save the model and apply the post-processing technique as described in II. Any trigger set that does not need any knowledge of the client's 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 the model has good accuracy on this set
- **ReTrain**: Fine-tune the model with the trigger set until the model has learned the watermark

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 the Projected Gradient Descent technique in a small Convolutional Neural Network (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 [96] between the original message and the reconstructed message.

3) FedTracker: FedTracker [90] 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. The server is assumed to be a trusted party that has no access to natural data related to the primitive task. Some clients can be malicious and others are assumed to be honest. For malicious clients, they can copy and distribute the model but they follow the training process to maximize the global accuracy. Each watermark can be described as follow :

- **Global Black-Box Watermark** : A trigger set is generated using the WAFFLEPattern method [16].
- **White-Box Watermark**: The server generates a random secret matrix and a fingerprint for each client.

Existing Works	**Verification**	**Watermarks embedding**	**Security Tools Compatibility**				**Clients Selection**	**Poisoning Defense**	
			DP	HE	MPC	TEE		Aggregation	Anomaly Detector
WAFFLE [16]	Black-Box	Server	◐	-	◐	◐	◐	◐	◐
FedIPR [17]	White-Box & Black-Box	Client(s)	●	◐	◐	◐	●	●	◐
FedTracker [90]	White-Box & Black-Box	Server	◐	-	◐	◐	◐	◐	◐
Liu et al. [91]	Black-Box	Client	◐	●	◐	◐	◐	-	-
FedCIP [92]	White-Box	Client(s)	◐	◐	◐	◐	●	◐	◐
FedRight [93]	White-Box	Server	◐	-	◐	◐	◐	◐	◐
Yang et al. [94]	Black-Box	Client	◐	●	◐	◐	◐	-	-
FedZKP [92]	White-Box	Client(s)	◐	◐	◐	◐	◐	◐	◐
Merkle-Sign [95]	White-Box & Black-Box	Server	◐	-	◐	◐	◐	◐	◐

TABLE III: FL watermarking taxonomy, categorized according to the access required to the verifier (white or black box), the entity that embeds the watermark (server, one or multiple clients), and their compatibility with the cryptographic security tools ●: Compatible and tested, ◐: Compatible but not tested, -: Not compatible

After the aggregation, the server embeds the **Global Black-Box Watermark** using the intuition of Continual Learning [97] 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) Liu et al. scheme: [91] The authors propose a client-side Black-box watermarking scheme (**S$_2$**). This technique is designed to embed a watermark only from one client (which represents the initiator of the FL). This framework is designed to be used under HE since the server is not trusted. The authors assume also that other clients can be malicious. During the verification process, a fully-trustworthy third party is recruited to be the arbitrator. The initiator creates a trigger set composed of Gaussian noise images with a given label as a trigger set. Then, the client's model will overfit with this set like in [73]. 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) FedRight: FedRight [93] 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 [98]. The security assumptions presented by authors assume that the server is trusted and honest while clients may not be trustworthy. 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.

6) FedCIP: FedCIP [92] is a White-Box watermarking framework that is compatible with FL security aggregation and allows tracking traitors. This solution allows the clients to watermark the FL model (**S$_2$**). They assume that the server is honest but curious while model theft can be done by any malicious client. The server cannot directly modify the model's parameters. And the last security hypothesis relies on the fact that the trigger set should not rely on the original data. This technique relies on the concept of a unique watermark per client for a given number of rounds called a cycle. During a cycle, a portion of cK of clients is designated to contribute during the following rounds. For each client, the watermark is unique during the cycle. If the round is the first iteration of the cycle, the algorithm will quickly replace the previous watermark with the new one. Otherwise, a small update is made to reinforce the watermark. The server divides the model parameters for all clients to avoid watermark conflicts and it records the participants at each round which is used for the verification process.

The verification is then performed using the "Federated Watermark" which is a concatenation of the watermark of all the contributors for a given cycle. If a client leaks the model multiple times during the FL, the authors point out that they can find the traitor by computing the intersection of the participants that have a high Watermark-Detection-Rate with the leak models. In particular, they can identify the traitor if $c^n K \leq 1$ where n is the number of traitor leaks.

7) Yang et al. scheme [94]: Yang *et al.* [94] proposed a Black-Box watermarking framework based on Liu *et al.* [91] solution (described in Section III-B4) (**S$_2$**). The main contribution of this paper is the proposed trigger set schema. Instead of using a classical Gaussian noise-based trigger set, they build images using permutation-based secret keys and noise-based patterns. The authors claim and demonstrate that this trigger set is resistant to ambiguity attacks. Whether the adversary tries to brute force the secret key or generate a new trigger set with its own key, both ways are hard to accomplish.

8) Merkle-Sign: Merkle-Sign [95] 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 [99]. In this framework, the server use at each round an embedding function to insert two pieces of identity information (i.e. keys) into M_G: one that identifies the server and the other one the client that will receive 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. In the final round, the server embeds also all keys generated by the clients into M_G and update the Merkle-tree. This framework is also compatible in a Peer-to-Peer context. The associated Black-Box watermarking scheme relies on the training of an Auto-Encoder [100] (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.

In the Merkle sign scheme, the server is trusted, and clients are considered honest but curious. The authors explore various attacks to assess the verification protocol's robustness, including ambiguity, spoiling, and traitor-tracing attacks. Ambiguity attacks can be countered by using an authorized time server or decentralized consensus protocol for timestamp authorization [65]. Security against spoil attacks involves applying multiple watermarks to the model, allowing clients to prove ownership even if an adversary spoils one watermark. For traitor tracing, the server embeds unique watermarks for each client, enabling successful ownership declaration over pirated models. However, the paper lacks a protocol for identifying traitor clients after training has terminated, as the model is watermarked using information from all clients simultaneously.

9) FedZKP: The authors in [86] propose a verification protocol based on the zero-knowledge proof, specifically on xLPN ZKP [101] called FedZKP. FedZKP doesn't require a trusted verifier to protect the confidentiality of the credentials, which reduces the risk of credential leakage. The security hypothesis presented in the paper assumes that clients are trusted entities, while the server and verifying authority are considered honest but curious (**S₂**). Watermarking is conducted by the clients, each equipped with a public and private key. The hash of all clients' public keys is then embedded into the model using a white-box mode. The paper examines four attacks: fine-tuning, pruning, and targeted destruction attacks. The experimental results demonstrate the watermark's robustness against these attacks. The last attack discussed in the paper involves a scenario where a user is required to prove ownership of a DNN model by sharing their public key with the trusted authority. However, this opens up the possibility for an adversary who intercepts the key to falsely claim ownership of the model. To address this vulnerability, the verification authority implements a zero-knowledge proof protocol, ensuring that only the individual possessing the private key associated with the public key in the watermarking can successfully demonstrate ownership.

It is worth noting that while the majority of the proposed solutions focus on IP protection, some papers explore alternative approaches using FL watermarking. For instance, WMDefense [102] utilizes FL watermarking to detect malicious client updates by assessing the degree of watermark recession. In this section, we have covered solutions encompassing both Black-Box/White-Box and Client(s)/Server watermarking with various ownership verification schemes. The summary of these solutions is presented in Table III. Each solution is categorized based on the verification process (White-Box/Black-Box), the entity performing the watermarking (Client(s) or Server), compatibility with standard security tools, compatibility with communication efficiency systems such as client selection, and compatibility with poisoning defense mechanisms. Upon review of the table, it becomes apparent that many solutions have not been tested with security tools like Differential Privacy (DP) or Homomorphic Encryption (HE). Additionally, none of the solutions have undergone testing with Multi-Party Computation (MPC) and Trusted Execution Environments (TEE). The same observation applies to poisoning defense mechanisms, where comprehensive testing is yet to be explored across various solutions

IV. 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. Trigger-set based-watermarking on 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 **S₂** and **S₃**. However, it is not so easy in **S₁**. When the client C_k wants to watermark its model M_{C_k}, he can rely on two things :

1) Have 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 II-B). Using such techniques is motivated by the fact that training the model from these datasets is 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 [103].

In **S₁**, since the server has not its own dataset, it cannot perform such type of watermarking. The choice is limited by using unrelated or noise-based inputs such as WafflePattern [16] or generating them from auto-encoders provided by clients such as Merkle-Sign [95]. As we said before, learning two tasks separately can be negative for Fidelity. FedTracker [90] is the only server-side framework in which they use Continual Learning [97] and a global gradient memory to not interfere with previously learned tasks.

In addition, this limitation leads to exposure to evasion attacks. During the Black-Box verification process, the owner will ask about the suspicious Application Programming Interface (API) that possibly contains his DNN. However, the attacker may evade this verification using a query detector [104]. 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 [105]. WAFFLE [16] authors confirm the intuition that the performances of such detectors depend a lot on the data quantity and capacities of the attacker.

B. Aggregation functions

The most common aggregation function is `FedAvg` [7] which consists of averaging clients' parameters 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` [40] `Trim-mean` or `Bulyan` [106]. 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 al.* [91] and its extension Yang *et al.* [94] 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 [43].

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 using `FedAvg` [32]. Existing watermarking techniques for FL have not evaluated methods that tackle this problem such as `FedProx` [32], `FedNova` [107] or `SCAFFOLD` [33]. If we want to use the proposed solution in a real Secure FL framework, these methods need to be tested in a such context which is actually not the case.

C. Clients selection

For communication efficiency and when the number of clients is too high, a fixed number of clients is selected at each round. 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, the majority of papers 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 own watermark, the watermark of not selected clients risks being degraded or removed in the global model. Authors of FedIPR [17] show that for $c > 0.2$ the detection rate for both White-Box and Black-Box watermark is near 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%. FedCIP [92] framework is specially designed with the clients' selection routine and demonstrates good performances for several watermarking requirements.

D. Cross-device setting

All proposed papers are treating the case in which we have a small number 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 [18]. 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 III-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. And it leads to difficulty for each client to embed its vector b without conflicting in the face of other clients' watermarks.

E. Differential Privacy and Homomorphic Encryption

Since Federated Learning (FL) ensures the privacy of clients' data by sharing the model or gradient updates between the server and clients (or directly among clients), there are concerns about potential attacks, such as membership inference, which can reveal the presence of specific data points [108]. To address this issue, Differential Privacy (DP) is often employed, providing robust privacy guarantees for algorithms working on aggregate databases [89], [109]. In FL, a common DP technique involves introducing Gaussian random noise to the gradients sent to the aggregator, adding an extra layer of privacy protection.

Another cryptographic measure to enhance security in Client-Server FL is Homomorphic Encryption (HE) [110]–[112]. In this approach, clients can protect their model updates using their public keys, encrypting the data before sending it to the server. The server performs the model aggregation (usually using `FedAvg`) in the encrypted space, ensuring that the server cannot misuse the privilege of having access to individual client updates to learn about private datasets. Clients can then use their private keys to decrypt the aggregated global model once it is received.

The combination of Differential Privacy and Homomorphic Encryption addresses privacy and security concerns in different aspects of FL. While DP protects against attacks attempting to extract sensitive information from gradient updates, HE safeguards the clients' data during aggregation, preventing unauthorized access by the server. In the context of watermarking, the use of HE is particularly suitable for scenario

($\mathbf{S_2}$), as clients can access their model parameters to embed watermarks securely. However, a challenge remains for scenario ($\mathbf{S_1}$), as the server cannot decrypt the model parameters to perform watermarking embedding.

F. 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 a unique solution that extends to the decentralized setting. 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 [57] can be more difficult for the server or for clients. In both cases, they have access to a part of the model parameters that can be arbitrarily small.

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

G. 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 model parameters. It has also sometimes a clear view of clients' parameters 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 update that they want. Nevertheless, they have no control over what happens with their updates at 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 [37] 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 [43]. The proposed method by Yang *et al.* [91], without HE, is then easily removable and the global model will not be watermarked. A solution to catch backdoored models was presented in [113] [114]. 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 [93] and WAFFLE [16] consists in the fact that multiple clients will use their models and private dataset. As mentioned in Section IV-A, an evasion attack works better when using multiple datasets are used to train the detector. But it is also possible to fine-tune the model with the combined dataset.

In summary, none of the client-side solutions have undergone testing with poisoning detection mechanisms such as anomaly detection or defensive aggregation functions like `Krum` or `GeoMed`. As for server-side solutions, no watermarking algorithm has been found compatible with cryptographic tools like Homomorphic Encryption (HE). Additionally, certain aggregation functions are not compatible with FL watermarking in both cases. Concerning data repartition, only a small portion of the proposed solutions have been tested in a non-I.I.D scenario, with no solutions tested in vertical data repartition or split learning. Despite the multitude of proposed solutions for FL watermarking, no FL framework (as presented in Table I) has integrated a tool to embed a watermark.

V. Conclusion

In the context of Federated Learning (FL), watermarking has gained significant interest due to the limitations of applying classical Deep Neural Network (DNN) watermarking in a collaborative setting. Addressing data distribution, new distributed threat models, and incorporating additional security mechanisms becomes crucial when designing an efficient solution for collaborative ML watermarking. This survey paper provides a comprehensive overview of FL watermarking, including its definition, requirements, and a taxonomy of existing methods, along with their security assumptions. We have also discussed the constraints that watermarking in the FL context can encounter concerning privacy-preserving issues. It is our hope that this paper will contribute to the advancement of research in the area of Intellectual Property (IP) protection, particularly in the context of vertical learning and split learning. Furthermore, we anticipate that these solutions will be seamlessly integrated into secure FL frameworks, enabling their deployment across a wider range of use cases.

References

[1] K. Singh, P. Booma, and U. Eaganathan, "E-commerce system for sale prediction using machine learning technique," in *Journal of Physics: Conference Series*, vol. 1712, no. 1. IOP Publishing, 2020, p. 012042.
[2] P.-h. Conze, M. E. H. DAHO, Y. LI, I. Brahim, H. Le Boité, P. Massin, R. Tadayoni, B. Cochener, G. Quellec, M. Lamard *et al.*, "Time-aware deep models for predicting diabetic retinopathy progression," *Investigative Ophthalmology & Visual Science*, vol. 64, no. 8, pp. 246–246, 2023.
[3] P. Mallozzi, P. Pelliccione, A. Knauss, C. Berger, and N. Mohammadiha, "Autonomous vehicles: state of the art, future trends, and challenges," *Automotive systems and software engineering: State of the art and future trends*, pp. 347–367, 2019.
[4] P. Regulation, "General data protection regulation," *Intouch*, vol. 25, pp. 1–5, 2018. [Online]. Available: https://gdpr-info.eu/
[5] D. Piper, "Data protection laws of the world," *DLA Piper*, 2019. [Online]. Available: https://www.dlapiperdataprotection.com/
[6] J. Chen and J. Sun, "Understanding the chinese data security law," *International Cybersecurity Law Review*, vol. 2, no. 2, pp. 209–221, 2021.
[7] 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.
[8] A. Benaissa, B. Retiat, B. Cebere, and A. E. Belfedhal, "Tenseal: A library for encrypted tensor operations using homomorphic encryption," *arXiv preprint arXiv:2104.03152*, 2021.

[9] T. Gehlhar, F. Marx, T. Schneider, A. Suresh, T. Wehrle, and H. Yalame, "Safefl: Mpc-friendly framework for private and robust federated learning," *Cryptology ePrint Archive*, 2023.

[10] H. Chen, K. Laine, and P. Rindal, "Fast private set intersection from homomorphic encryption," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1243–1255.

[11] C. Dwork, "Differential privacy: A survey of results," in *International conference on theory and applications of models of computation*. Springer, 2008, pp. 1–19.

[12] K. Wei, J. Li, C. Ma, M. Ding, W. Chen, J. Wu, M. Tao, and H. V. Poor, "Personalized federated learning with differential privacy and convergence guarantee," *IEEE Transactions on Information Forensics and Security*, 2023.

[13] W. Zheng, Y. Cao, and H. Tan, "Secure sharing of industrial iot data based on distributed trust management and trusted execution environments: a federated learning approach," *Neural Computing and Applications*, pp. 1–11, 2023.

[14] Z. Xu, Y. Zhang, G. Andrew, C. A. Choquette-Choo, P. Kairouz, H. B. McMahan, J. Rosenstock, and Y. Zhang, "Federated learning of gboard language models with differential privacy," *arXiv preprint arXiv:2305.18465*, 2023.

[15] Q. Yang, A. Huang, L. Fan, C. S. Chan, J. H. Lim, K. W. Ng, D. S. Ong, and B. Li, "Federated learning with privacy-preserving and model ip-right-protection," *Machine Intelligence Research*, vol. 20, no. 1, pp. 19–37, 2023.

[16] 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.

[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] 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.

[19] 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.

[20] 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.

[21] F. Cremonesi, M. Vesin, S. Cansiz, Y. Bouillard, I. Balelli, L. Innocenti, S. Silva, S.-S. Ayed, R. Taiello, L. Kameni *et al.*, "Fed-biomed: Open, transparent and trusted federated learning for real-world healthcare applications," *arXiv preprint arXiv:2304.12012*, 2023.

[22] "Tensorflow federated: Machine learning on decentralized data." [Online]. Available: https://www.tensorflow.org/federated?hl=en

[23] A. Ziller, A. Trask, A. Lopardo, B. Szymkow, B. Wagner, E. Bluemke, J.-M. Nounahon, J. Passerat-Palmbach, K. Prakash, N. Rose *et al.*, "Pysyft: A library for easy federated learning," *Federated Learning Systems: Towards Next-Generation AI*, pp. 111–139, 2021.

[24] D. J. Beutel, T. Topal, A. Mathur, X. Qiu, J. Fernandez-Marques, Y. Gao, L. Sani, K. H. Li, T. Parcollet, P. P. B. de Gusmão *et al.*, "Flower: A friendly federated learning research framework," *arXiv preprint arXiv:2007.14390*, 2020.

[25] "An industrial grade federated learning framework." [Online]. Available: https://fate.fedai.org/

[26] G. A. Reina, A. Gruzdev, P. Foley, O. Perepelkina, M. Sharma, I. Davidyuk, I. Trushkin, M. Radionov, A. Mokrov, D. Agapov *et al.*, "Openfl: An open-source framework for federated learning," *arXiv preprint arXiv:2105.06413*, 2021.

[27] H. Ludwig, N. Baracaldo, G. Thomas, Y. Zhou, A. Anwar, S. Rajamoni, Y. Ong, J. Radhakrishnan, A. Verma, M. Sinn *et al.*, "Ibm federated learning: an enterprise framework white paper v0. 1," *arXiv preprint arXiv:2007.10987*, 2020.

[28] H. R. Roth, Y. Cheng, Y. Wen, I. Yang, Z. Xu, Y.-T. Hsieh, K. Kersten, A. Harouni, C. Zhao, K. Lu *et al.*, "Nvidia flare: Federated learning from simulation to real-world," *arXiv preprint arXiv:2210.13291*, 2022.

[29] "Federated learning powered by nvidia clara." [Online]. Available: https://developer.nvidia.com/blog/federated-learning-clara/

[30] 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.

[31] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.

[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] 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.

[34] F. Mo, A. S. Shamsabadi, K. Katevas, S. Demetriou, I. Leontiadis, A. Cavallaro, and H. Haddadi, "Darknetz: towards model privacy at the edge using trusted execution environments," in *Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services*, 2020, pp. 161–174.

[35] R. Kanagavelu, Z. Li, J. Samsudin, Y. Yang, F. Yang, R. S. M. Goh, M. Cheah, P. Wiwatphonthana, K. Akkarajitsakul, and S. Wang, "Two-phase multi-party computation enabled privacy-preserving federated learning," in *2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*. IEEE, 2020, pp. 410–419.

[36] M. Fang, X. Cao, J. Jia, and N. Gong, "Local model poisoning attacks to {Byzantine-Robust} federated learning," in *29th USENIX security symposium (USENIX Security 20)*, 2020, pp. 1605–1622.

[37] 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.

[38] A. Huang, "Dynamic backdoor attacks against federated learning," *arXiv preprint arXiv:2011.07429*, 2020.

[39] C. Xie, K. Huang, P.-Y. Chen, and B. Li, "Dba: Distributed backdoor attacks against federated learning," in *International conference on learning representations*, 2019.

[40] 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.

[41] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *International Conference on Machine Learning*. PMLR, 2018, pp. 5650–5659.

[42] E. M. Anass, C. Gouenou, and B. Reda, "Poisoning-attack detection using an auto-encoder for deep learning models," in *International Conference on Digital Forensics and Cyber Crime*. Springer, 2022, pp. 368–384.

[43] 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.

[44] S. Li, Y. Cheng, W. Wang, Y. Liu, and T. Chen, "Learning to detect malicious clients for robust federated learning," *arXiv preprint arXiv:2002.00211*, 2020.

[45] M. Xue, J. Wang, and W. Liu, "Dnn intellectual property protection: Taxonomy, attacks and evaluations," in *Proceedings of the 2021 on Great Lakes Symposium on VLSI*, 2021, pp. 455–460.

[46] N. Lukas, E. Jiang, X. Li, and F. Kerschbaum, "Sok: How robust is image classification deep neural network watermarking?" in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 787–804.

[47] Y. Li, H. Wang, and M. Barni, "A survey of deep neural network watermarking techniques," *Neurocomputing*, vol. 461, pp. 171–193, 2021.

[48] A. Fkirin, G. Attiya, A. El-Sayed, and M. A. Shouman, "Copyright protection of deep neural network models using digital watermarking: a comparative study," *Multimedia Tools and Applications*, vol. 81, no. 11, pp. 15 961–15 975, 2022.

[49] F. Boenisch, "A systematic review on model watermarking for neural networks," *Frontiers in big Data*, vol. 4, p. 729663, 2021.

[50] Y. Sun, T. Liu, P. Hu, Q. Liao, S. Ji, N. Yu, D. Guo, and L. Liu, "Deep intellectual property: A survey," *arXiv preprint arXiv:2304.14613*, 2023.

[51] D. Bouslimi, R. Bellafqira, and G. Coatrieux, "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)*. IEEE, 2016, pp. 2496–2499.

[52] F. Ernawan and D. Ariatmanto, "A recent survey on image watermarking using scaling factor techniques for copyright protection," *Multimedia Tools and Applications*, pp. 1–41, 2023.

[53] A. Pavan and M. Somashekara, "An overview on research trends, challenges, applications and future direction in digital image watermarking," *International Research Journal on Advanced Science Hub*, vol. 5, no. 01, 2023.

[54] D. Niyitegeka, G. Coatrieux, R. Bellafqira, E. Genin, and J. Franco-Contreras, "Dynamic watermarking-based integrity protection of homomorphically encrypted databases–application to outsourced genetic data," in *International Workshop on Digital Watermarking*. Springer, 2018, pp. 151–166.

[55] D. Hu, Q. Wang, S. Yan, X. Liu, M. Li, and S. Zheng, "Reversible database watermarking based on order-preserving encryption for data sharing," *ACM Transactions on Database Systems*, vol. 48, no. 2, pp. 1–25, 2023.

[56] C. Song, T. Ristenpart, and V. Shmatikov, "Machine learning models that remember too much," in *Proceedings of the 2017 ACM SIGSAC Conference on computer and communications security*, 2017, pp. 587–601.

[57] 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.

[58] L. Feng and X. Zhang, "Watermarking neural network with compensation mechanism," in *Knowledge Science, Engineering and Management: 13th International Conference, KSEM 2020, Hangzhou, China, August 28–30, 2020, Proceedings, Part II 13*. Springer, 2020, pp. 363–375.

[59] Y. Li, B. Tondi, and M. Barni, "Spread-transform dither modulation watermarking of deep neural network," *arXiv preprint arXiv:2012.14171*, 2020.

[60] 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.

[61] H. Chen, B. D. Rouhani, C. Fu, J. Zhao, and F. Koushanfar, "Deepmarks: A secure fingerprinting framework for digital rights management of deep learning models," in *Proceedings of the 2019 on International Conference on Multimedia Retrieval*, 2019, pp. 105–113.

[62] J. Wang, H. Wu, X. Zhang, and Y. Yao, "Watermarking in deep neural networks via error back-propagation," *Electronic Imaging*, vol. 2020, no. 4, pp. 22–1, 2020.

[63] B. D. Rouhani, H. Chen, and F. Koushanfar, "Deepsigns: an end-to-end watermarking framework for protecting the ownership of deep neural networks," in *The 24th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS). ACM*, 2019.

[64] L. Fan, K. W. Ng, and C. S. Chan, "Rethinking deep neural network ownership verification: Embedding passports to defeat ambiguity attacks," *Advances in neural information processing systems*, vol. 32, 2019.

[65] F. Li and S. Wang, "Secure watermark for deep neural networks with multi-task learning," *arXiv preprint arXiv:2103.10021*, 2021.

[66] R. Bellafqira and G. Coatrieux, "Diction: Dynamic robust white box watermarking scheme," *arXiv preprint arXiv:2210.15745*, 2022.

[67] M. Kuribayashi, T. Yasui, and A. Malik, "White box watermarking for convolution layers in fine-tuning model using the constant weight code," *Journal of Imaging*, vol. 9, no. 6, p. 117, 2023.

[68] P. Lv, P. Li, S. Zhang, K. Chen, R. Liang, H. Ma, Y. Zhao, and Y. Li, "A robustness-assured white-box watermark in neural networks," *IEEE Transactions on Dependable and Secure Computing*, 2023.

[69] E. Rodriguez-Lois and F. Perez-Gonzalez, "Towards traitor tracing in black-and-white-box dnn watermarking with tardos-based codes," *arXiv preprint arXiv:2307.06695*, 2023.

[70] H. Chen, B. D. Rouhani, and F. Koushanfar, "Blackmarks: Blackbox multibit watermarking for deep neural networks," *arXiv preprint arXiv:1904.00344*, 2019.

[71] Y. Vybornova, "Method for copyright protection of deep neural networks using digital watermarking," in *Fourteenth International Conference on Machine Vision (ICMV 2021)*, vol. 12084. SPIE, 2022, pp. 297–304.

[72] 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.

[73] 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.

[74] 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.

[75] 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.

[76] R. Namba and J. Sakuma, "Robust watermarking of neural network with exponential weighting," in *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*, 2019, pp. 228–240.

[77] Z. Li, C. Hu, Y. Zhang, and S. Guo, "How to prove your model belongs to you: a blind-watermark based framework to protect intellectual property of dnn," in *Proceedings of the 35th Annual Computer Security Applications Conference*, 2019, pp. 126–137.

[78] K. Kapusta, V. Thouvenot, and O. Bettan, "Watermarking at the service of intellectual property rights of ml models," in *Actes de la conférence CAID 2020*, 2020, p. 75.

[79] S. Lounici, M. Önen, O. Ermis, and S. Trabelsi, "Blindspot: Watermarking through fairness," in *Proceedings of the 2022 ACM Workshop on Information Hiding and Multimedia Security*, 2022, pp. 39–50.

[80] K. Kallas and T. Furon, "Rose: A robust and secure black-box dnn watermarking," in *IEEE Workshop on Information Forensics and Security*, 2022.

[81] T. Qiao, Y. Ma, N. Zheng, H. Wu, Y. Chen, M. Xu, and X. Luo, "A novel model watermarking for protecting generative adversarial network," *Computers & Security*, vol. 127, p. 103102, 2023.

[82] K. Kallas and T. Furon, "Mixer: Dnn watermarking using image mixup," in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023, pp. 1–5.

[83] G. Hua, A. B. J. Teoh, Y. Xiang, and H. Jiang, "Unambiguous and high-fidelity backdoor watermarking for deep neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[84] T. Wang and F. Kerschbaum, "Riga: Covert and robust white-box watermarking of deep neural networks," *arXiv preprint arXiv:1910.14268*, 2019.

[85] J. Zhang, D. Chen, J. Liao, W. Zhang, G. Hua, and N. Yu, "Passport-aware normalization for deep model protection," *Advances in Neural Information Processing Systems*, vol. 33, pp. 22 619–22 628, 2020.

[86] W. Yang, Y. Yin, G. Zhu, H. Gu, L. Fan, X. Cao, and Q. Yang, "Fedzkp: Federated model ownership verification with zero-knowledge proof," *arXiv preprint arXiv:2305.04507*, 2023.

[87] L. Rosasco, E. De Vito, A. Caponnetto, M. Piana, and A. Verri, "Are loss functions all the same?" *Neural computation*, vol. 16, no. 5, pp. 1063–1076, 2004.

[88] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[89] 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.

[90] 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.

[91] 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.

[92] J. Liang and R. Wang, "Fedcip: Federated client intellectual property protection with traitor tracking," *arXiv preprint arXiv:2306.01356*, 2023.

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

[94] W. Yang, S. Shao, Y. Yang, X. Liu, Z. Xia, G. Schaefer, and H. Fang, "Watermarking in secure federated learning: A verification framework based on client-side backdooring," *arXiv preprint arXiv:2211.07138*, 2022.

[95] 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.

[96] G. Wahba *et al.*, "Support vector machines, reproducing kernel hilbert spaces and the randomized gacv," *Advances in Kernel Methods-Support Vector Learning*, vol. 6, pp. 69–87, 1999.

[97] 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.

[98] 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.

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

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

[101] A. Jain, S. Krenn, K. Pietrzak, and A. Tentes, "Commitments and efficient zero-knowledge proofs from learning parity with noise," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2012, pp. 663–680.

[102] X. Zheng, Q. Dong, and A. Fu, "Wmdefense: Using watermark to defense byzantine attacks in federated learning," in *IEEE INFOCOM 2022-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2022, pp. 1–6.

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

[104] 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.

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

[106] 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.

[107] 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.

[108] 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.

[109] B. Wang, Y. Chen, H. Jiang, and Z. Zhao, "Ppefl: Privacy-preserving edge federated learning with local differential privacy," *IEEE Internet of Things Journal*, 2023.

[110] R. Bellafqira, G. Coatrieux, E. Genin, and M. Cozic, "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*. Springer, 2019, pp. 322–336.

[111] J. Ma, S.-A. Naas, S. Sigg, and X. Lyu, "Privacy-preserving federated learning based on multi-key homomorphic encryption," *International Journal of Intelligent Systems*, vol. 37, no. 9, pp. 5880–5901, 2022.

[112] W. Jin, Y. Yao, S. Han, C. Joe-Wong, S. Ravi, S. Avestimehr, and C. He, "Fedml-he: An efficient homomorphic-encryption-based privacy-preserving federated learning system," *arXiv preprint arXiv:2303.10837*, 2023.

[113] 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.

[114] 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.