SoK: On the Security & Privacy in Federated Learning

Gorka Abad  
Digital Security Group  
Radboud University  
Nijmegen, The Netherlands &  
Security in Digital Platforms  
Ikerlan Technology Research Centre  
Arrasate-Mondragón, Spain  
gabad@ikerlan.es

Stjepan Picek  
Digital Security Group  
Radboud University  
Nijmegen, The Netherlands  
stjepan.picek@ru.nl

Aitor Urbieta  
Security in Digital Platforms  
Ikerlan Technology Research Centre  
Arrasate-Mondragón, Spain  
aurieta@ikerlan.es

Abstract—Advances in Machine Learning (ML) and its wide range of applications boosted its popularity. Recent privacy awareness initiatives as the EU General Data Protection Regulation (GDPR) – European Parliament and Council Regulation No 2016/679, subdued ML to privacy and security assessments. Federated Learning (FL) grants a privacy-driven, decentralized training scheme that improves ML models’ security. The industry’s fast-growing adaptation and security evaluations of FL technology exposed various vulnerabilities. Depending on the FL phase, i.e., training or inference, the adversarial actor capabilities, and the attack type threaten FL’s confidentiality, integrity, or availability (CIA). Therefore, the researchers apply the knowledge from distinct domains as countermeasures, like cryptography and statistics.

This work assesses the CIA of FL by reviewing the state-of-the-art (SoTA) for creating a threat model that embraces the attack’s surface, adversarial actors, capabilities, and goals. We propose the first unifying taxonomy for attacks and defenses by applying this model. Additionally, we provide critical insights extracted by applying the suggested novel taxonomies to the SoTA, yielding promising future research directions.

Index Terms—Federated Learning, Taxonomy, Systematization, Threat model, Security, Privacy, Attacks, Defenses.

I. INTRODUCTION

Recent improvements and research in Artificial Intelligence (AI) and Machine Learning (ML) opened a broad number of applications and opportunities [1]. More concretely, ML is applied to different tasks, and autonomous driving [3] and cybersecurity [4]. What is more, ML can be used in both offensive and defensive tasks. For example, ML has a defensive role, determining if the network traffic is legitimate or a file contains malware. ML has also been applied in offensive tasks, e.g., generating malicious content to bypass such defenses [5]. In the same way, the usage of ML models in some critical scenarios, i.e., autonomous driving, requires safety measures for preventing accidents involving human lives [6]. Related to this, in the last three years, AI sector leaders, e.g., Google, Amazon, Microsoft, and Tesla, received attacks on their ML algorithms [7], which are publicly available as Machine Learning as a Service for little cost. As such, improving the security and privacy of ML becomes an urgent and crucial task.

To enhance security and privacy, in 2016, Google presented a novel privacy-driven, decentralized ML scheme that enables training many devices towards a single-shared model without exposing the training data, named Federated Learning (FL) [8]. Similar to ML, FL has grown in popularity as a result of its broad applicability [9], [10]. Preserving privacy is the primary concern of FL’s development [8]. Legal requirements as the EU General Data Protection Regulation (GDPR) – European Parliament and Council Regulation No 2016/679 boosted FL’s progress and usage [11]. The wide range in applicability and its popularity placed FL in security and privacy assessments’ scope [12]. Recent privacy assessments concluded that privacy leaks occur in different segments of the FL procedure by various means [13]. Different research areas (cryptography, ML, distributed technologies) have already analyzed FL’s security or privacy under their scope [14]. However, since FL’s introduction in 2016, none have developed a unifying taxonomy for attacks and defenses that help FL with its security and privacy assessment. Some work has been done respectively in specific areas of FL, e.g., privacy [15], security [16], edge computing [17], and open problems [18].

The rapid growth of attack and defense proposals thwart efforts to determine the current security and privacy state of FL [18]. This article systematizes the current state of security and privacy in FL by developing two unifying taxonomies. First, we analyze the state-of-the-art (SoTA), which converges into a threat model. We further develop two holistic taxonomies for attacks and defenses by applying them to the reviewed articles. By this means, the proposed taxonomies solve the lack of consensus in attacks and defenses derived from the extensive attack surface and fast evolution of the technology. Therefore, our contributions are threefold:

- We create a threat model describing all the assets that pose a risk for FL. It identifies and makes understandable the FL’s threats while considering its context.
- By applying the threat model, we develop two unifying novel taxonomies for attacks or defenses, which joins...
different lines of research regarding FL’s security and privacy. The proposed taxonomies consider a wide range of parameters that unequivocally describe an attack or a defense.

- According to the proposed holistic taxonomies, we investigate the SoTA attacks and defenses and classify them accordingly. From there, we extract the relationships between attacks and defenses, metrics used for evaluation, and common experimental setups.

II. MACHINE LEARNING

This section briefly explains the centralized ML procedure, where a dataset is collected from various sources and kept in a single place for the training process. In particular, we describe the phases involving the ML procedure.

A. Types of Learning

ML is the automated process of accurately making predictions based on the existing data. ML algorithms adapt their structure to fit the provided data to maximize the outcome via an iterative process known as training [19]. Learning can be supervised, unsupervised, semi-supervised, and reinforcement [19], [20].

We mainly focus on supervised learning for the rest of the article. In supervised learning, the goal is to perform classification (map an input to a label) or regression (map an input to a continuous output). The model uses a dataset, a collection of labeled examples \( \{(x_i, y_i)\}_{i=1}^N \), where \( N \) is the size of the collection. Each sample \( x_i \), named a feature vector, contains a label \( y_i \) of class \( c_n \): \( C = \{c_1, c_2, c_3, \ldots, c_n\} \). Each dimension of the feature vector \( x_i^{(j)} \) \( (j = 1, 2, 3, \ldots) \) describes (in a specific way) the sample. For instance, a feature \( x_i^{(1)} \) may describe the height of a person and \( x_i^{(2)} \) its weight. The goal of supervised learning is to create a model that, with an unseen feature vector \( x \) as an input, can infer its label \( y \).

B. Machine Learning Phases

The ML procedure is divided into two phases. The training phase describes the stage where the model uses the dataset for learning. A model \( f \) takes a feature vector \( x \): \( f(x) \) as input for describing its label \( y \leftarrow f(x) \). During the learning process, the model looks for an optimal vector of weights \( \theta \in \Theta : f_\theta \). \( f_0 \) is set by an iterative learning process, aiming to minimize a loss function \( L(x) : \arg\min \ L(x) \) while a disjoint test set is used for measuring the quality of the model.

The inference (test) phase relates to when the model is deployed after achieving convergence, ready for performing a given task. A trained model \( f_\theta \) performs inference over a feature vector \( y \leftarrow f_\theta(x) \) as the input.

C. Federated Learning

The FL network comprises three sections: the clients, the aggregator, and the communications among them. Each component of the network, upon consensus, trains the same ML model. The training procedure, e.g., batch size, learning rate, is also chosen by agreement. Each client owns a disjoint dataset to locally train the model, which is afterward uploaded to the aggregator. Instead of uploading the model itself, clients upload the vector of weights representing the model. Later, the aggregator, following an aggregation algorithm, i.e., Federated Averaging (see Section II-C1), joins each client’s model. After aggregation, the vector of weights is sent back to each client for another training round until convergence.

This client-aggregator distribution is known as client-server or decentralized architecture, unlike centralized ML. By removing the aggregator and performing the aggregation process among the clients in the network, a fully decentralized architecture is obtained, named peer-to-peer (P2P) FL or fully decentralized FL [21], [22].

1) Federated Averaging: Stochastic Gradient Descent (SGD) [23] is the most common optimization method for minimizing a loss function in ML. Since its release, the ML community has driven its research towards improving SGD [20]. Federated SGD (FedSGD) is considered the baseline aggregation algorithm for developing an improved version of SGD [24]. It functions as follows: for each client, \( k \in K \), where \( |K| > 0 \), it performs a local training procedure. The batch size \( \beta \) is the entire dataset size, i.e., \( \beta = \infty \). Thus, for each training round, each client \( k \) sends the trained weights \( \Theta \) to the server. Then, the server will, by averaging, aggregate those and send the newly joined model to each client \( k \).

Federated Averaging (FedAvg) improves the training procedure and leverages the decentralized scheme of FL. It is based on mini-batch training where \( \beta \) is smaller than the dataset, i.e., \( \beta \neq \infty \) for a single epoch, improving the training speed. For each training round with \( \beta \) as batch size, the weights \( \Theta \) are sent to the server. Then the server aggregates the updates similar to FedSGD and sends those back to each client \( k \) for another training round. FedAvg improves the generalization and accuracy of the overall model. However, it incurs noticeable communication overhead [8].

D. Datasets & Data Distribution

Since datasets are owned by each client, their heterogeneity and data distribution directly influence the local model accuracy. Therefore, FL leverages the heterogeneity of datasets of clients to improve the global model quality. The data distribution where each client has different dataset lengths and data types is known as non-Independent and Identically Distributed (non-I.I.D.). These are contrary to Independent and Identically Distributed (I.I.D.) data, where each client has the same dataset length without overlapping. For achieving a non-I.I.D. distribution from I.I.D. data, the Dirichlet distribution [25] is usually used. The attacks and defenses performance is directly affected by the data distribution, as demonstrated in [26]. In FL real-world scenarios, data distribution will commonly follow a non-I.I.D. distribution [8], [27]. After SoTA analysis, we found that the main application field of FL is image-based datasets.

Additionally, data is partitioned into Vertical FL (VFL), Horizontal FL (HFL), Federated Transfer Learning (FTL), and hybrid FL. HFL is a data distribution that shares the same features related to unique identities. The original use
of Google’s GBoard used HFL since clients make similar use of the keyboards, from which it extracted the same features for distinct users. This approach takes advantage of either the extracted features without identifying clients. As an example, an application collects usage data from clients worldwide. The collected data is transverse but, the identity of clients remains unknown. Datasets are horizontally partitioned when: let \( i, j \) be two clients that share the same features \( D \) for a feature space \( X : X_i^D = X_j^D \). However, they do not share the same client IDs \( ID_i \neq ID_j \).

Contrary to HFL, VFL is the data partitioning scheme that shares the same client identity for distinct features. Usually, the application of VFL requires a trusted third party along with encryption. For example, an application creates personalized advertisements. The application collects or buys information regarding a target client from different sources that enhance the experience of the target client. Datasets are vertically partitioned when: let \( i, j \) be two clients that share unique features \( D \) for a feature space \( X : X_i^D \neq X_j^D \). However, they share the same client IDs \( ID_i = ID_j \).

FTL [8], [31], [32], adapted from classical ML scenarios, is a scheme that grants the ability to train a new requirement on an already trained model. The new requirement could solve a different problem from what the original was trained for by training on a similar dataset. Training on a pre-trained model enhances the overall model quality, which is easier than training from scratch [29]. For example, a model that detects malicious network traffic from HTTP requests now detects malicious traffic over MQTT [13]. A model \( f \) that was trained over a domain \( D \) containing a feature space \( X \) can predict a label \( y \) on the label space \( Y \) for an input \( x \) as \( y \leftarrow f(x) \). Considering the source domain \( D_S \) and the target domain \( D_T \) where \( D_S \neq D_T \) the objective of FTL is to help improve the learning capabilities of the target predictive function \( f_T \) in \( D_T \) using the knowledge of \( D_S \).

Similarly, under hybrid settings, different clients share distinct features [33]. Let \( i, j \) be two clients that share the distinct features \( D \) and distinct IDs \( ID \) for a feature space \( X : X_i^D \neq X_j^D \wedge ID_i \neq ID_j \).

During our research, we found that the most used data distribution is HFL, whereas a single article used a P2P architecture. Apart from data distribution, FL is categorized into cross-device FL and cross-silo FL, depending on the number of clients and their capabilities [13]. The network contains many low computational powered devices in cross-device settings, as in Google’s GBoard scenario [8], [27], [34]. Usually, the clients are IoT devices or smartphones.

Contrary to cross-device settings, in cross-silo the network contains a few computationally powerful devices as data centers [27]. Usually, cross-silo architectures are more connection reliable than cross-device settings, where the clients may be unreachable [11].

III. Threat Model

In this section, we analyze and structurally represent the information that affects the security and privacy of FL. We first investigate the attack surface, i.e., what sections of FL are vulnerable in what phases. Second, we study the different actors involved in the FL process and how they may threaten the FL environment. Third, we analyze the adversarial capabilities and classify them as white-box and black-box. Finally, we review the adversarial goals and map them according to confidentiality, integrity, and availability (CIA).

A. FL Attack Surface

The FL network is composed of clients, an aggregator, and the communications between them. During the training phase, clients train their datasets locally. A poisoned dataset, e.g., gathered from untrusted sources, will contaminate the client’s model, polluting the aggregated model after submission. Furthermore, an attacker may compromise model exchanges by intercepting, eavesdropping, interrupting, or modifying the communications.

Since FL is an iterative process, models are not required to be near convergence for using them. Hence, models can be threatened during the test phase by querying adversarial inputs, i.e., Adversarial Examples, or extracting private information, i.e., Model Inversion, by observing the model’s outputs. Similarly, the aggregator might be threatened by external adversaries that mangled the aggregation process. Lastly, a malicious external actor could target the communications between clients and the aggregator through eavesdropping, interrupting, or altering communications.

B. Adversarial Actors

Two actors participate in the FL process, the clients and the aggregator. Under adversarial settings, a third one, an outsider as an external actor, has to be considered. The trust level for the FL environment is directly affected by the trust levels of the actors. Depending on whether the adversarial actors are part of the network or outsiders, they are classified as an insider or outsider. An insider is an actor that actively participates in the FL process. The aggregator and the clients belong to this group. The client has access to the model, its weights, the learning procedure, the dataset, and the communications. Likewise, the aggregator has access to each client model, their weights, the communications, and the aggregation algorithm. Usually, insiders are categorized as honest-but-curious and interact passively with the capabilities mentioned above. As an outsider, a third party has a lower level of knowledge. Depending on its capabilities, an outsider may have access to the communications and query the model.

C. Adversarial Capabilities

As previously introduced, outsiders and insiders have different levels of knowledge and capabilities. This section defines and categorizes adversarial capabilities accordingly. Since FL clients have unlimited access to the model’s snapshots, they could leverage certain information, such as the local dataset, local learning rate, number of local epochs, local batch size, model architecture, communications, or its weights for empowering the attack. Similarly, the aggregator has access
to a snapshot of the model. It could leverage information for each client update, communications, or aggregation algorithm to improve attacks capabilities. On the contrary, an outsider could leverage information regarding communications and oracle access to the model to enhance its attacks.

Thus, we consider a black-box attack when an adversary can query the model \( f(x) \) with any arbitrary input \( x \) and obtain the result. However, the adversary cannot access any inner computation of the model or the training procedure. In a white-box setting, the attacker may leverage or modify any of the above information to empower the attack. Because of the FL nature, most of the attacks are white-box.

D. Adversarial Goals

Adversarial actors may have different aims and approaches for attacking certain parts of the FL surface. We classify the adversarial goals concerning their capabilities, the FL phase, and the CIA triad. The attack aim is categorized as:

- Targeted: the adversarial aim is to map chosen inputs to desired outputs or predictions.
- Untargeted: the adversarial goal is to degrade the primary task performance, so the model does not achieve near-optimal performance.

Targeted and untargeted attacks are achieved independent of the FL phase or the adversarial knowledge. However, training phase attacks target the CIA of the model. For example, Poisoning attacks target the integrity and availability of the model. Meanwhile, Inference attacks target confidentiality. Test-time attacks mainly target confidentiality. However, untargeted attacks as Adversarial Examples target the integrity of the model. Likewise, untargeted attacks, whose primary goal is to degrade the model’s performance, can lead to a denial of service, causing unavailability.

The details of the attacks and CIA mapping are explained in Sections IV and V. In Sections IV, V, and VI, we study the threat model and the SoTA countermeasures.

IV. TRAINING IN ADVERSARIAL SETTINGS

This section categorizes and explains the training-time attacks concerning the CIA. We further propose a taxonomy for each attack type by analyzing the SoTA attacks presented in Table I and considering the adversarial goals and capabilities shown in Fig. I.

A. Targeting Integrity & Availability

1) Poisoning Attacks: Under FL, each client owns a dataset that could be modified for malicious purposes. As insiders, attackers have access to the training procedure, the model, and the dataset. An attacker leverages those capabilities to perform Poisoning attacks. Untargeted Poisoning attacks reduce the overall performance of the model. The attack ranges from degrading the main task where the model does not achieve a near-optimal performance to increasing the convergence time leading to a Denial of Service (DoS). However, targeted Poisoning attacks modify the model to achieve the desired output.

Once the joined model is poisoned, the aggregator shares it with each client in the network, thus poisoning every local model. The adversarial objective is achievable under two techniques:

- Single-shot: poison a model in a single FL round with a single adversarial client.
- Multiple-shot: poison a model in several FL rounds with 1 ≤ adversarial clients.

Furthermore, depending on the capabilities of the attackers, attacks are further categorized into Data Poisoning, Model Poisoning, and Backdoors attacks.

a) Data Poisoning: Data Poisoning attacks are most common in ML but not very effective in FL because updates from benign clients tend to reduce the negative effect of the malicious dataset during the aggregation process. As the attackers modify the dataset, Data Poisoning is thus considered a white-box attack.

Data Poisoning attacks rely on dataset modification during local training to degrade the overall model performance achieved by label-flipping. The intuition behind label-flipping is to alter the labels of the local dataset. An attacker owns a dataset made of \( \{(x_i, y_i)\}_{i=1}^{N} \), a set of vectors \( x_i \) and its corresponding class \( y_i \), where \( N \) is the size of the dataset. The attack aims to change the ground truth class \( y_i \) of \( x_i \), to an adversarial class \( a \). Then, the dataset gets poisoned by an adversarial entry \( \{(x_i, a)\} \). For example, for the MNIST dataset, an attacker will swap the ‘1’ data samples to ‘7’, namely 1 to 7 attack. For the general case \( \{(x_i, y_i)\} \rightarrow \{(x_i, a)\} \) and for this example \( \{(x_i, 1)\} \rightarrow \{(x_i, 7)\} \).

For example, the authors in [42] presented an anomaly detection model for IoT using FL, where they gradually injected malicious traffic in the dataset using label-flipping, successfully bypassing defensive clustering techniques. Following the same technique, authors in [38] noticeably reduced the model’s predictive accuracy.

Label-flipping is further categorized regarding the existence of a source of label verifier, which proves the correctness of the chosen label as a pre-processing step before training:

- Clean-label [43, 44, 45]: the intuition behind clean-label attacks is that they do not require control over the labeling function. The poisoned dataset seems to be correct to the eyes of an expert label-verifier identity. These types of attacks are difficult to detect and usually occur when data is gathered from untrusted sources to construct a dataset. An attacker may place a poisoned instance online and wait for it to be gathered, labeled, and included in the dataset. Since the piece of data seems to be accurately labeled and thus used for training, the model is trained over a poisoned dataset and, therefore, contaminated.
- Dirty-label [46]: dirty-label consists of modifying the dataset by changing the label of a data piece to the desired one (targeted) or to a random one (untargeted), e.g., changing all planes labels into birds that the final model classifies planes images as birds (targeted) or arbitrarily assigning labels to the training samples.
b) **Model Poisoning:** Data Poisoning attacks have shown poor performance in real-life scenarios [37]. Thus, a new alternative has emerged to manipulate model weights directly: Model Poisoning attacks [47]. They overcome the low success of Data Poisoning attacks by using Boosting. Boosting is a technique that enlarges clients’ contributions by multiplying the vector weights (the update) by a scaling factor. Similarly, clients could modify any additional parameter from the training procedure for malicious purposes. After the training procedure over a poisoned dataset, the attacker owns a model \( f \) represented by a vector of weights \( w \). The attacker boosts its model by multiplying it by a Boosting factor \( \lambda : \lambda w \).

Authors in [47] further optimized the Model Poisoning attack by applying SGD to the scaling factor for finding its optimal value. An adversarial client could change the local training rate, the weights, the training epochs, or submit arbitrary updates. Since this attack modified the vector of weights directly, it is a white-box attack. A Boosted model has more influence over the global model, achieving successful single-shot attacks. However, defenses have arisen towards this Boosting-based model replacement attacks because it increases the \( \lambda \_n \) norm of the vector of weights (see Section VI). Additionally, several proposed stealthy techniques evaded such defenses [48], [49], [50], [51], [52], [53]. Furthermore, [54] proposed Sybils that evaded and improved attacks quality. They are coordinated attacks among various adversarial clients. A single attacker could control more than a single client, allowing an adversary to smartly distribute its attack, improving stealthiness and its adversarial accuracy. Sybils are mainly used in Model Poisoning but are also present in other types of adversarial threats [55]. Likewise, the authors in [48] evaluated Model Poisoning attacks against Byzantine-robust FL. They tested Model Poisoning against Krum [56], Bulyan [57], trimmed mean [58], and mean defensive mechanisms. None of them could stop the attack. The authors further investigated adapting Krum and Reject on Negative Impact (RON) [59] for defending Model Poisoning attacks, but with not enough success. Other approaches [60], [61] leveraged Generative Adversarial Networks (GANs) [62] for generating poisoned data. The created poisoned dataset is then used to train an adversarial model that reduced the attack’s assumptions and increased the Model Poisoning attack’s feasibility in real-world scenarios.

c) **Backdoor Attacks:** Backdoors [63] are a particular type of Poisoning attack, also named Trojans. Authors in [37] investigated the possibility of performing a Data Poisoning attack on chosen labels while working non-adversarially on the rest. They leveraged specific data of the dataset that contained certain information, i.e., green cars with white stripes, and then applied label-flipping to those. Thus, the resulting model performed in an adversarial manner over green cars with white stripes as input. However, Backdoors need Boosting to become effective [37]. A model \( f \) is trained on backdoored data samples and represented by \( w \) vector of weights. The attacker may (or may not) want to boost its model by a scaling factor \( \lambda : \lambda w \). Furthermore, the attacker may leverage information about an anomaly detection \( L_{ano} \) for evading aggregator’s defensive techniques. By controlling the importance of evading the defense \( \alpha \), the attacker may change the loss function \( L \) from its training procedure, \( L = \alpha L + (1 - \alpha) L_{ano} [37] \).

Regarding the trigger used, they are further classified into:

- **In-Backdoor** [47]: when the trigger label exists in at least one of the datasets of any client in the network. For a set of datasets \( S \) where \( S = \{D_1, D_2, D_3, ... D_n\} \), let \( n \) be the number of clients in the network. \( D_n \) is made of data samples \( \{(x_i, y_i)\}_{i=1}^n \). At least one sample’s class \( y_i \) of at least one party in \( S \) does belong to the adversarial class \( a : \{(x_i, a)\} \in S \).

- **Out-Backdoor** [64]: when the trigger class does not exist in any of the datasets for any client in the network, i.e., \( \{(x_i, a)\} \notin S \).

Depending on the trigger type, Backdoor attacks are categorized into:

- **Edge Backdoor:** the authors in [65] suggested a type of Backdoor attack that relied on using rare data pieces, i.e., edge samples of the dataset with adversarial labels, for local training so the resulting model will only misclassify those rare inputs. One benefit of this method is that gradient-based defense techniques are unlikely to detect this attack [65]. The attacker uses an edge case dataset \( D' \) extracted from \( D \). Then it performs standard local training aiming to maximize the accuracy. However, Server Cleaning techniques (see Section VI) removed such a simple attack because the data distribution of \( D' \) is largely different from \( D \). For solving it, they further proposed a stealthier Projected Gradient Descent (PGD) attack. The attacker applied PGD to maximize the similarity of updates from benign clients. To improve the attack, the authors used Boosting for achieving a model replacement.

- **Semantic Backdoor:** a type of attack that does not modify the input but the labels, e.g., all the cars with white stripes are labeled as birds [37]. It is compulsory to use Boosting to ensure that the aggregation survives the averaging, and the Backdoor is thus successfully transferred to the global model.

- **Pixel-pattern Backdoor:** primarily used in image recognition, and in contrast with Semantic Backdoors, pixel-pattern attacks [66] inject a pattern in a particular area of the input space. After altering some local data, the attacker performs local training with a mixture of adversarial and benign data to follow a similar non-Backdoored data distribution. The resulting model misclassified the inputs that contained the trigger pattern while correctly classifying harmless inputs, e.g., some pixel pattern has been injected into several car images and labeled as a bird. After local training and global model aggregation, the resulting model will misclassify every car input containing the trigger, predicting them as birds, while a benign user gets a correct classification for a non-Backdoored car input. For improving the stealthiness, the authors in [67] evaluated pixel-pattern Backdoors with no scaling factor where they evaded most of the clipping...
defensive mechanisms. As a particular case of pixel-pattern attacks, single-pixel attacks do not necessarily use large patterns to trigger a Backdoor. Instead, they only require modifying a single pixel in the input space [68].

Since Backdoors development, the research drifted towards testing them in different environments [69], [70] and improving Backdoor techniques as dynamically changing Backdoors [52]. In addition, [71] investigated Backdoors in real-world scenarios. They performed the research over non-I.I.D. data distribution and various datasets. They also analyzed norm-clipping and Differential Privacy defenses, with almost no model performance degradation. Authors in [72], [73] proposed evading techniques, namely distributed Backdoor attacks, where the Backdoor got split among adversarial clients. Each client performed a Backdoor in a coordinated manner, improving stealthiness, and after aggregation, the split Backdoor gets joined.

Backdoor attacks are recognized as both Data and Model Poisoning, depending on the altered factor. Solely altering the dataset, i.e., a Backdoor without Boosting is considered a Data Poisoning-based Backdoor. However, when using Boosting techniques, Backdoors are considered Model Poisoning-based Backdoors [74].

B. Targeting Availability

1) Free-riders: Free-riders [75], [76] are clients that do not contribute to the network by different means. The motivation behind the attack is that the Free-rider client does not want to collaborate in the network or does not have enough resources for training the local model, e.g., enough computational power or a sufficiently large dataset. The attacker disguises itself by sending random updates similar to the joined model to the aggregator downgrading its overall quality. Commonly the adversary does not require any extra information for sending random updates. An attacker submits a random vector of weights $\mathbf{w}_{\text{rand}}$ to the aggregator. Or the attackers send an update that resembles to the aggregated model $\mathbf{w}_{\text{agg}}$: $\mathbf{w}_{\text{rand}} \simeq \mathbf{w}_{\text{agg}}$.

When the server implements defense mechanisms [76], the attacker might leverage additional information for evading them in a white-box manner. SoTA Free-rider attacks aimed to bypass defenses implemented by the aggregator [77]. The attacker trains the model locally with a small dataset for saving resources. Since clients have control over the training procedure, the attacker claims to use a large dataset. The attack reduced the model convergence time by 50%. The authors in [75] analyzed the ease of detection for plain Free-rider attacks. Thus, they developed a stealthy Free-rider attack, which added noise to the crafted updates. Then, SGD is applied to maximize the resemblance with the updates of other clients. Likewise, some servers offered rewards for contributing clients. A Free-rider might change the weights via Boosting for achieving higher representation on the global server and own a more significant part of the reward [77].

V. INFERRING IN ADVERSARIAL SETTINGS

This section explains inference-time attacks and maps them according to the CIA.

A. Targeting Integrity

1) Evasion Attacks: Mostly used in ML, Adversarial Examples [78] leverage a precisely crafted input to misclassify the model at inference-time. Since the model is unknown to the attacker, the attack is a black-box. However, under FL, where the model is known to an adversarial inside the network, they may leverage that extra information for performing a more powerful attack. Thus it would be considered a white-box. The intuition behind the attack is: from an input $\mathbf{x}$ and a crafted noise $\epsilon$ fool the model $f$ to misclassify it in a targeted or untargeted manner. A crafted input $\mathbf{x} + \epsilon$ and its ground truth label $y$ fool the model as $y \leftarrow f(\mathbf{x} + \epsilon)$. Similarly, Evasion attacks are also developed in a physical context [79], [80], where the crafted noise is included physically in the real world rather than embedded via software, e.g., evading face detection systems [79].

B. Targeting Confidentiality

1) Inference Attacks: Inference attacks use different knowledge to extract private information. Depending on the adversarial capabilities and the phase attacked, the attacker leverages oracle access to the model, its inner computations, or the updates [81]. Depending on the attacker’s activities, Inference attacks are classified as:

- Passive [82], [83]: the attacker, honest-but-curious, could only read or eavesdrop on communications, the local model, and the dataset.
- Active [83]: more effective than passive attacks but less stealthy, leverages the extra information obtained by mangling the local model, local training procedure or its parameters, local dataset, or communications.

Furthermore, depending on the target information, Inference attacks are further classified as:

- Membership Inference [85]: the goal is to infer whether some data belongs to the training dataset. Under FL, it is even possible to suggest which user owns the dataset. The attacker aim is to infer if some data piece $\{(\mathbf{x}, y)\}$ belongs to a local dataset $D_i : \{(\mathbf{x}, y)\} \in D_i$, where $n$ is the number of clients in the network. Or in the general case whether $\{(\mathbf{x}, y)\} \in S$ belongs to the dataset $S = \{D_1, D_2, D_3, ..., D_n\}$.
- Property Inference [84], [85]: this attack leverages model snapshot updates concerning a dataset without the objective property and another one containing it. Afterward, a trained binary classifier can relate the property with an unseen update, e.g., an update got by eavesdropping, to conclude whether the update owns the objective property. For a data piece $\mathbf{x}$ containing the property $p : \mathbf{x}^{(p)}$ belonging to the dataset $D_n$ being $n$ the number of clients, the attacker infers if the property belongs to the dataset $\mathbf{x}^{(p)} \in D_n$. 

• Model Inversion [86], [87]: an ML model is trained to infer a label for a given input. As demonstrated by [88] ML models tend to remember training data. The same authors inverted the ML inference-time process by first providing the label backward to the model while maximizing the likelihood of the target label. The result is a piece of data similar to the one used during training. Even authors in [86] achieved pixel-wise accuracy on reconstructing images and token-accurate on text-based models [89]. Model Inversion attacks can be categorized depending if the attacker can solely query the model, i.e., black-box, or has access to the model parameters, i.e., white-box. From a model \( f \) trained on the dataset \( D \), the Model Inversion attack tries to reconstruct a piece of data \( \hat{x} \approx x \in D \) by querying \( f \).

Regarding the SoTA, the authors in [83] proposed a white-box Membership Inference attack. They leveraged the attacker's position inside the network to acquire the model. With the model, they obtained every layer's intermediate computation on an input. They then trained a Convolutional Neural Network (CNN) with the gained weights, distinguishing if the input is present in some client dataset. Extending this attack, the authors in [90] successfully performed user-level Membership Inference mapping the input to a specific client's dataset. Leveraging the inferred information, the authors in [91] performed an improved Model Inversion attack. Similarly, the authors in [82] used a GAN for empowering their attack. The GAN generated data samples that followed the same data distribution as the real one. Then this data was passed to the global model for setting a label and merging the generated data with actual data. Given that the generated data were not present in none of the client's datasets, the authors trained a model to infer its membership. For Model Inversion attacks, the authors in [92], [93] also used GANs. With a malicious server, they deployed a GAN trained on the updates of a victim client. Levering an auxiliary dataset, the model could create samples similar to the input. Improving this approach, the authors in [94] developed a Generative Recurrent Neural Networks that removed the need for an auxiliary dataset. Contrary, in [95], the authors considered VFL instead of HFL. They researched the effects of Membership Inference in VFL for Logistic Regression and Decision Trees, widely used models in the industry. The authors in [96] investigated privacy threats to Linear Regression, Logistic Regression, and Decision Tree while considering Model Inversion attacks.

2) Model Extraction Attacks: Model Extraction attacks try to mimic or fully copy a target model. Leveraging oracle access to a model \( f \), Model Extraction attacks try to reconstruct it with oracle access \( x : f(x) \approx \hat{f}(x) \). The model \( \hat{f} \) acquired by approximation relies on creating a model that, by iterative adjustments, performs similar to the original model. However, it is not architecturally the same. For reconstructing a floating-point accurate model, an analytical approach is needed, named direct analysis [27], [98]. Because of the gradient descent procedure during training, which implies second derivatives computation, the model holds some critical points where the derivative equals 0. Meeting those points leads to floating-point accurate weights recovery.

Extraction attacks are common in ML, where an adversary does not know details about the model rather than its architecture, i.e., black-box settings. However, clients of the FL network already have access to the model. Nevertheless, an outsider may perform a Model Extraction attack for owning the model.

VI. TOWARDS SECURE FL

In this section, we explain the SoTA defensive mechanisms and group them according to the CIA. Additionally, we propose a taxonomy for each defense type by analyzing the SoTA defenses presented in Table IV and considering the adversarial goal and capabilities presented in Fig. 2.

A. Defending Confidentiality

1) Cryptographic Methods: Cryptographic Methods (CMs) prevent Inference attacks in FL. For a general use case, the clients or the aggregator perform local or in-server encryption to the updates. Thus, securing the communications between them. This defensive mechanism prevents an attacker from privacy leaks, but it is computationally expensive.

Secure Multi-party Computation (SMC) enabled secret sharing without a trusted third party [99], [100]. Furthermore, SMC deals with cross-silo settings, where clients may be unreachable, dropouts are common, and communications expensive. Other authors [101] proposed leveraging a proxy-like Trusted Execution Environment (TEE) that mixes layers from the updates of different clients before sending these to the aggregator. Similarly, Homomorphic Encryption (HE) relies on encrypting data, where mathematical operators over encrypted and non-encrypted data result in the same. Furthermore, HE theoretically does not carry a performance loss on the model convergence [102], which enabled its use in decentralized settings [11], [27], [103], [104].
Likewise, the authors in [117] proposed using a Self-defensive mechanism that reduces the communications over differential Privacy in the server and the clients. Compressing noise to the updates, similar to [113], which applied Differentially during the training process or in the server by the decision, will contain noise, degrading attack efficiency. However, model. Therefore, an Inference attack, such as Model Inversion, requires the server for limiting a wide range of attacks [107], [108], preserving mechanism that adds Gaussian or Laplace noise to methods are not viable in the FL context.

Another approach followed by [110], [111], [112] added noise to the updates, similar to [113], which applied Differential Privacy in the server and the clients. Compressing the model before sharing it with the aggregator is another defensive mechanism that reduces the communications overloads apart from improving confidentiality [64], [114], [115], [116]. Likewise, the authors in [117] proposed using a Self-Normalizing Neural Network, which combined normalizing and Differential Privacy techniques, adding noise per NN layer. For preventing GAN-based Inference attacks, the authors in [118] developed an anti-GAN mechanism. They leveraged a Wasserstein GAN for distorting the datasets of clients. Thus, an attacker using a GAN-based attack would learn over a wrong data distribution, reducing the attack performance.

**Challenge 1:** CMs can prevent Inference attacks only. A recent proposal has successfully performed Inference over encrypted data. Thus exploring enhancements for CMs is necessary. Alternatively, we must understand if such defense methods are not viable in the FL context.

2) **Differential Privacy:** Differential Privacy is a privacy-preserving mechanism that adds Gaussian or Laplace noise to the model for limiting a wide range of attacks [107], [108], [109]. The core idea is that noise alters the weights of the model. Therefore, an Inference attack, such as Model Inversion, will contain noise, degrading attack efficiency. However, the model accuracy will also drop. Each client adds the noise locally during the training process or in the server by the aggregator.

Another approach followed by [110], [111], [112] added noise to the updates, similar to [113], which applied Differential Privacy in the server and the clients. Compressing the model before sharing it with the aggregator is another defensive mechanism that reduces the communications overloads apart from improving confidentiality [64], [114], [115], [116]. Likewise, the authors in [117] proposed using a Self-Normalizing Neural Network, which combined normalizing and Differential Privacy techniques, adding noise per NN layer. For preventing GAN-based Inference attacks, the authors in [118] developed an anti-GAN mechanism. They leveraged a Wasserstein GAN for distorting the datasets of clients. Thus, an attacker using a GAN-based attack would learn over a wrong data distribution, reducing the attack performance.

**Challenge 2:** Differential Privacy provides defense against a variety of attacks without significant degradation of the model performance. Still, it remains to be studied if it is the best defensive mechanism for defending various attacks.

### B. Defending Integrity & Availability

1) **Pruning:** Data-filtering is a common technique for preventing Poisoning attacks in ML [119] by removing the poisoned data from the dataset. However, under FL settings, the datasets of clients are not accessible to the aggregator. After the attack, some neurons of the poisoned model contained the adverse effect. The intuition of Pruning [120] is to remove those for sanitizing the model. The authors in [121] applied Pruning to an FL environment, successfully preventing Poisoning attacks.

**Challenge 3:** Pruning techniques were solely applied for defending against Model Poisoning attacks. Further work is required to establish the viability of Pruning for defending against other types of attacks.

2) **Server Cleaning:** Server Cleaning, also named Sanitization [27], focuses on filtering malicious updates, similarly to Robust FL Aggregators, where the server discards them, i.e., does not modify the base aggregation algorithm. However, Server Cleaning requires a separated aggregator algorithm. For
such purposes, the server might leverage different techniques based on the intrinsic properties of the updates [74].

Typical approaches used statistical deviation or patterns that differentiate a malicious update. The authors in [59] presented a defense based on RONI, where the server discarded updates that negatively affect the accuracy. Towards calculating such, they proposed training a classification model based on a clean-holdout dataset. Then they compared the classification accuracy between the clean model and the uploaded. If the uploaded model’s accuracy was lower than the clean’s, it got discarded. This approach required a heterogeneous dataset to generalize properly and not discard benign updates.

Similarly, the authors in [21] dynamically discarded adversarial clients. They adapted the Induced Ordered Weighted Averaging (IOWA) operator for the FL environment. IOWA is a function for weighting the updates of the clients on a distributed system. Similarly, the authors in [122] proposed a self adversarial-detecting system where the clients were clustered and shared relevant features. Other approaches, as in [123], performed clustering on the server for detecting malicious updates. A similar approach was followed by [124] applying Support Vector Machines, dimensionality reduction using Principal Component Analysis [38], or Digestive Neural Networks [125]. Similarly, the authors in [70] applied Matching Networks for discarding updates based on similarity. Other authors [31] applied Differential Privacy locally and tested the models’ quality on edge, improving communications efficiency while providing privacy guarantees. The authors in [126] leveraged a Blockchained structure for ensuring data integrity and traceability, where a committee validates updates via Proof-of-Work. For further read in Blockchain-based FL, refer to [127].

The authors in [76] defended Free-riders attacks by applying a Deep Autoencoding Gaussian Mixture Model (DAGMM). DAGMM is composed of a compression and estimation net, where the updates are passed through the networks, and their distances are measured between dimensions and their standard deviation. On the same basis, the authors in [128] applied Autoencoders for the defense. In particular, they proposed a DAGMM, where the encoder performed a dimensionality reduction. Afterward, the remaining dimensions were passed to an estimation network that predicted the likelihood density. The larger the density more likely it was to be a malicious update. Similarly, in [77], the authors proposed a density-based anomaly detection. They assigned an anomaly score to each model, where the aggregator weighted them according to the score. Therefore, reducing the adversarial relevance on the global model. Other confidence or score-based anomaly detection have been proposed [77], [129], [130].

Furthermore, the authors in [131] suggested training a Linear Regression model for identifying data pieces that are close to legitimate data pieces from a poisoned dataset. Differently, the authors in [42] proposed the use of the K-means clustering algorithm, leveraging a clean-holdout dataset for removing outliers based on trimmed means.

Also, during the Boosting process in Model Poisoning attacks, the weights were adapted for increasing relevance in the aggregated model [71], which carried an increase in the $l_1$ norm. Compared with benign updates, the server might define a threshold and discard the updates that do not satisfy it [132]. Likewise, the authors in [51] proposed using the Kolmogorov-Smirnov (K-S) test for cleaning the model. The K-S test is used in non-parametric statistics for measuring the distance between two data distributions. They compared the distance between data distributions, removing those distributions that were different from the majority. However, they only demonstrated its effectiveness under I.I.D. settings.

TEE enables a secure area that provides confidentiality and integrity guarantees. The authors in [133] applied TEE to centralized ML for training a Deep Neural Network (DNN) and analyzing its security. Similarly, the authors in [134] applied TEE locally for the clients for training and on the server for securing the aggregation. This scheme, as proved, successfully defended against inference-time attacks while also guaranteeing data integrity. TEE ensures that the aggregation procedure or the model’s parameters are not halteded.

**Challenge 4: Server Cleaning methods can defend against the majority of attacks except for Adversarial Examples. However, this method has to be specifically implemented for defending against a particular threat. Therefore, a further study with more focus on generalizing Server Cleaning techniques is needed.**

3) Adversarial Training: As previously introduced, Data Poisoning attacks alter the local dataset. Because of the decentralized setting of FL, Data Cleaning methods were not applicable. Adversarial Training inserts poisoned data on the dataset during the training phase [135], [136]. Once trained, the model can differentiate poisoned data. However, this defense requires a large dataset and is computationally challenging. As a solution, the authors in [137] proposed sharing resources, where the most prepared or powerful devices could share the secure model with other clients. Furthermore, for preventing Evasion attacks, the authors in [138] smoothed the training data by applying Gaussian noise and including adversarial data in the training dataset.

**Challenge 5: Adversarial Training exclusively defends against Adversarial Examples in FL environments. It is important to understand how Adversarial Training could be used for protecting against other types of attacks.**

4) Robust FL Aggregation: During the aggregation process, the server could discard malicious updates [139], [140], [141], [142], [143], [144], [145] or remove the poisoning effect [146]. As previously mentioned, FedAvg is the baseline aggregation algorithm. Still, the server could use a robust aggregator instead. For instance, the authors in [56] described Krum. For each client’s update, the server calculated the sum of the Euclidean distance towards the other models. Then, the server selected the model with a lower distance, i.e., the most similar. Therefore, for networks where over 50% is not an adversary, the chosen model was benign. Krum notably reduced the adversarial accuracy but was not as effective in non-I.I.D. settings. The same authors presented a variation of Krum,
named multi-Krum [56], which smartly altered between Krum and averaging for improving performance while providing resilience.

In [147], the robust algorithm adjusted the learning rate per round and dimension to maximize the loss on adversarial updates instead of minimizing it. The authors in [49] designed a divide-and-conquer dimensionally reduction defense. After reducing the dimensions, the search spectrum was smaller and thus more efficient, then the authors applied single value decomposition for detecting outliers. Another novel approach named Baffle was presented by [148]. They designed an algorithm where the clients send the updates to some chosen clients evaluating the updates against their local dataset and sharing the results back to the aggregator. After benchmarking, the aggregator could detect inconsistencies in clients.

Analogously, FoolsGold [149] and Sniper [150] are defense mechanisms against Sybil attacks. The authors suggested that since Sybils work together towards a common adversarial goal, their updates were cosine-similar. Therefore, the benign and malicious updates were not cosine-similar. Thus, the server could remove those. Even though this defense performed correctly against Sybil’s targeted poisoning attacks over non-I.I.D. and I.I.D., it did not function with a single attacker [149], [150]. Similarly, following the same intuition, the authors in [151] presented Romoa, which, based on historical updates, discarded malicious ones.

Moreover, coordinate-wise trimmed median [58] aimed to remove Sybil updates of Poisoning attacks. The intuition was to eliminate updates that are dissimilar from harmless updates. Then the remaining updates were averaged. In addition, the authors in [152] presented the Robust Federated Aggregation algorithm, which aggregated the updates by calculating the weighted geometric median using the smoothed Weiszfeld’s algorithm. Similarly, authors in [58] proposed the coordinate-wise median defense, which performed an arithmetic median for each update. Bulyan [57] used an aggregator algorithm such as Krum for interactively selecting some updates, aggregated by trimmed-mean.

Other approaches focused on setting confidence values or scores to each update for getting corresponding participation on the joint model [26], [153], [154], [155]. Similarly, the authors in [156] relied on the Shapley value for eliminating Backdoor attacks. The Shapley value yields how each client collaborates to the common objective, thus removing low collaborative updates.

**Challenge 6:** Robust FL Aggregation mechanisms properly defend against training-time attacks. However, they do not apply to inference-time attacks. We propose performing additional research on evaluating Robust FL Aggregators for defending against threats at inference-time.

5) GAN-based Defenses: GANs [62] have demonstrated outstanding performance in the last years [157]. Designed for images, GANs can create pictures and video from noise. They are composed of two networks, a generator \( \mathcal{G} \) and a discriminator \( \mathcal{D} \). They both compete against each other simultaneously where \( \mathcal{G} \) tries to create new, non-seen images similar to the real ones. \( \mathcal{D} \) tries to differentiate actual inputs from generated ones. After achieving the Nash equilibrium, \( \mathcal{G} \) can create new inputs that \( \mathcal{D} \) cannot differentiate from real ones. Therefore, GANs may be split and use \( \mathcal{G} \) autonomously. The authors in [60], [61] used GANs for datasets augmentation, enhancing the performance of Poisoning attacks. Similarly, an adversary might perform Inference attacks by reconstructing data using \( \mathcal{G} \) [82].

As shown, Model Poisoning attacks are more powerful and more popular than Data Poisoning. Their growth in popularity influenced, along with GANs, the GAN-based defensive mechanisms development. The authors in [158], [159], [160] used GANs for creating an auditing dataset, which provided to a classifier, could differentiate adversarial updates. Another approach followed by [161] consisted in adding noise to the updates. They also used dimensionality reduction for improving training time in the cost of model accuracy.

GAN’s versatility allowed their use in both attacks and defenses successfully. However, as FL rounds go by, the data distribution drifts, requiring continuous training of GANs. Furthermore, in non-I.I.D. environments, GANs did not achieve exceptional quality as in I.I.D.

**Challenge 7:** GANs were only applied for defending against Model Poisoning attacks. It is necessary to determine GANs’ viability for other types of attacks and non-I.I.D. environments.

**VII. Discussion**

A. Recognizing the FL Threats

After extensive analysis of the SoTA, we mapped each attack type regarding the defensive mechanisms used in the experiments and vice versa (see Fig. 3). Each column represents the attacks, while defenses are shown horizontally. The color scheme plots the percentage of used defensive mechanisms for an attack. By analyzing the figure, we extract that for training-time attacks, the most common countermeasures are applied during the training phase, i.e., Server Cleaning, Robust FL
TABLE II: Comparison of defenses regarding different features. † denotes a backdoor attack.

| Defenses                                                                 | Data distribution | Model | Performance metrics | Attack                                                                 | Source code | FL algorithm | Framework   |
|-------------------------------------------------------------------------|-------------------|-------|---------------------|-------------------------------------------------------------------------|-------------|--------------|-------------|
| Pruning                                                                  |                   |       |                     |                                                                          |             |              |             |
| Server Cleaning                                                          |                   |       |                     |                                                                          |             |              |             |
| Cryptographic Methods                                                   |                   |       |                     |                                                                          |             |              |             |
| Adversarial Training                                                    |                   |       |                     |                                                                          |             |              |             |
| Differential Privacy                                                    |                   |       |                     |                                                                          |             |              |             |
| Robust FL Aggregation                                                   |                   |       |                     |                                                                          |             |              |             |

Shah et al. [162] | Sun et al. [156] | Chen et al. [155] | Azar et al. [144] | Lin et al. [155] | Zhao et al. [133] | Wang et al. [148] | Song et al. [157] | Liu et al. [109] | Sun et al. [151] | Cao et al. [101] | Mallick et al. [143] | Liu et al. [144] | Sun et al. [145] | Xie et al. [144] | Li et al. [146] | Xie et al. [141] | Li et al. [143] | Cao et al. [103] | Mao et al. [141] | Li et al. [143] | Liu et al. [153] | Um et al. [145] | Liu et al. [153] | Zhao et al. [145] | Lin et al. [146] | Yue et al. [134] | Yao et al. [135] | Busan [146] | Sharma et al. [152] | Sun et al. [153] |

**TABLE II**

| Feature | Analysis | Dataset Type | Data Poisoning | Model Poisoning | Free-rider | Inference | Model Inversion | Free-rider |
|---------|----------|--------------|----------------|-----------------|------------|-----------|-----------------|-----------|
| I.I.D.  | Yes      | Yes          | Yes            | Yes             | Yes        | Yes       | Yes             | Yes       |
| Non-I.I.D. | Yes      | No           | No             | No              | No         | No        | No              | No        |
| Not defined | No      | No           | No             | No              | No         | No        | No              | No        |

Regarding Pruning defenses, as seen in Table II and the heat map in Fig. 3, none of the evaluated attacks tested its performance against Pruning defenses.

**Challenge 8:** Since Pruning has not been bypassed yet in FL, we propose to develop attacks that embed the backdoor effect inside non-Poisoned neurons so that the adversarial effect remains present.

In other than FL fields, i.e., adversarial machine learning, the authors [162] developed ensemble pre-trained adversarial models for transferring the adverse effect to a target model. This approach has not been applied to FL yet.

**Challenge 9:** Since ensemble attacks are applied in other settings like centralized ML, we suggest adapting them for FL and exploring their effectiveness.

---

**Fig. 3:** Attacks and defenses relation.

**B. Attacks & Defenses Performance**

The quality of an attack or a defense is influenced by different means: the dataset type, data distribution, model type,
training procedure, number of clients, number of adversarial clients, and selected clients per round. We investigate this effect, represented in Fig. 4 for attacks and Fig. 5 for defenses.

In Model Poisoning attacks, the Attack Success Rate (ASR), how many rounds the poisoning survives the aggregation, is relevant for adjusting the boosting factor. However, augmenting the boosting factor entails an increase in the $l_n$-norm of the weights vector. Defense mechanisms also use this information to prevent Model Poisoning attacks. Furthermore, an attacker could leverage this information for increasing its stealthiness by minimizing the $l_n$-norm while maximizing the ASR. For Inference and Model Inversion attacks, since it is complicated to quantify the similarity between ground truth data and a reconstructed one, mostly in images, instead of using the ASR, it is measured by the Mean Squared Error (MSE). Likewise, an attacker may minimize the MSE to achieve better attack performance. For the rest of the cases, i.e., Free-Riders, Adversarial Samples, and Data Poisoning attacks, the attacker may leverage the information provided by the confusion matrix, e.g., accuracy, True Positive Rate, Area Under the ROC Curve, and Structural Similarity (SSIM).

Evaluating the defenses for preventing privacy degradation in Inference attacks, the countermeasures based on CM measure its effectiveness with Fake Attack Rate (FAR). FAR measures the number of times the attacks failed. Similarly, cryptography-based defenses also use Known Data Ratio as the ratio between the amount of known data by the attacker used for Model Inversion and the amount of data of the victim. For Differential Privacy, Robust FL Aggregator, or Server Cleaning, the evaluation metrics vary, caused by the wide threat surface. In contrast, the confusion matrix is the most common metric for evaluating GAN-based defenses.

We suggest that each attack or defense type requires similar evaluation metrics plus a similar setup to be comparable. In research, we discover that this is not the case. Confusion matrix metrics are the only metric considered in most situations. However, it may not be an accurate metric for evaluating attacks or defenses success.

**Challenge 10:** Different metrics are used for evaluating attacks and defenses performance. However, it is unclear what metrics measure the attacks and defenses the best.

Furthermore, we encounter an inconsistency when evaluating the effect of the number of clients on the attacks and defenses. Few articles provide the number of clients, the number of adversarial clients, and the number of clients selected per round, complicating experiments’ evaluation and repeatability.

**Challenge 11:** We propose to perform a further study to evaluate the effect of the number of clients on attacks and defenses performance.

As previously demonstrated, any defensive mechanism will somewhat reduce the model’s quality. Further research should be performed to investigate the trade-off between security, privacy, and accuracy.

**Challenge 12:** There is no research on finding the appropriate level of security against an attack. Defining security and privacy boundaries for guaranteeing a safe model is therefore relevant.

**C. Fields of Application**

In our research, we find that image-based tasks are the most common. Thus, attacks and defenses are primarily developed in this field. However, FL architectures, such as HFL, are well suited for many low-power devices, such as IoT [163]. FL lacks benchmark testing in this real-world case scenario. Therefore, extending the application field of attacks and defenses to a broader range, specifically IoT environments, is relevant.

**Challenge 13:** Since attacks and defenses are primarily tested against image-based datasets, extending the application field to IoT and testing their validity is necessary.

According to the gathered data, attacks and defenses proposals are mainly focused on I.I.D. environments, which is not a real case scenario in decentralized settings.

**Challenge 14:** Many of the proposed attacks and defenses are proposed under I.I.D. environments. Extending them...
and studying their applicability for non-I.I.D. scenarios are necessary.

As mentioned above, image-based tasks enable the use of specially designed attacks and defenses. GANs-based attacks and defenses are mainly applied in 2D environments and lack applicability in 1D environments, such as IoT. Therefore, adapting GANs or Variational Autoencoders for 1D would extend the attacks and defenses application field.

**Challenge 15:** We recommend establishing the viability of GAN-powered attacks and defenses in 1D environments like IoT.

Lastly, in our research, we discover that attacks and defenses are only proposed for supervised learning.

**Challenge 16:** Since FL threats and countermeasures are developed for supervised learning, we suggest evaluating their usage in unsupervised learning and FTL.

### VIII. RELATED WORK

The security and privacy of ML have been extensively evaluated [168], [169], [170]. However, such aspects are still under early-phase evaluation due to FL’s novelty. The diversity of scenarios makes performing an FL’s security and privacy evaluation as a whole difficult. For that reason, existing works solely focused on specific threats or use-cases for auditing FL. This effect is directly visible in the latest overviews for FL’s security and privacy, presented in Table III.

The authors in [13], [27] evaluated the security and privacy issues in FL by reviewing some of the SoTA attacks and defenses. While both works considered different types of FL architectures, solely [13] reviewed data partitioning types. In contrast, [27] slightly studied data distribution and its implications. Other works [14], [164], [165], [166] mainly focused on Poisoning and Inference attacks. Regarding privacy, existing works [11], [12], [15], [17], [167] evaluated privacy implications and issues on Inference attacks. Other authors [13] further studied privacy implications in FL by proposing a taxonomy for privacy-preserving defenses. Regarding security, authors in [74] analyzed the security implications of datasets. Similarly, recent work [17] examined some SoTA attacks under a security lens and proposed a taxonomy for that.

In our work, we thoroughly evaluate the SoTA attacks and defenses under the scope of security and privacy. We further study different FL architectures, data distributions, data partitioning, evaluation metrics, and frameworks. Leveraging the gathered data, we design the first holistic taxonomy for attacks and defenses while considering security and privacy.

Other works [14], [164], [165], [166] apart from Poisoning attacks also investigated Inference attacks and their countermeasures. Similarly, the authors in [11], [12], [15], [167] focused their research solely on Inference threats and their privacy considerations. They also evaluated defense mechanisms, but only [12], [17] assessed privacy-driven frameworks for FL. Contrary, the authors in [10] concentrated their research on the evaluation of different defenses. They focused on Server Cleaning, CM, Differential Privacy, and Robust FL Algorithms.

### IX. CONCLUSIONS

The active area of FL’s security and privacy research is heterogeneous in proposals. This work proposes a threat model that contains the attack surface, adversarial actors, capabilities, and goals. Under its scope, we analyze the SoTA attacks and defenses and classify them accordingly, and develop two novel taxonomies for attack and defenses, respectively. We discuss properties of attacks and defenses relations, data distribution, evaluation metrics, and dataset effects. Overall, FL merged different research areas towards a common objective of securing it. Moreover, we recognize promising challenges that should be addressed in future work. The taxonomy provides context for existing and emerging proposals, slowly contributing further knowledge on the FL’s security and privacy.

### REFERENCES

[1] L. Zhou, S. Pan, J. Wang, and A. V. Vasiliakos, “Machine learning on big data: Opportunities and challenges,” *Neurocomputing*, vol. 237, pp. 350–361, 2017.
H. Wang, K. Sreenivasan, S. Rajput, H. Vishwakarma, S. Agarwal, X. Chen, C. Liu, B. Li, K. Lu, and D. Song, “Targeted backdoor attacks on deep learning systems using data poisoning,” arXiv preprint arXiv:2011.07429, 2020.

A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, “Analyzing federated learning through an adversarial lens,” in International Conference on Machine Learning, pp. 634–643, PMLR, 2019.

K. Wei, J. Li, M. Ding, C. Ma, Y.-S. Jeon, and H. V. Poor, “Covert model poisoning against federated learning: Algorithm design and optimization,” arXiv preprint arXiv:2101.11799, 2021.

V. Shejwalkar and A. Houmansadr, “Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning,” Internet Society, p. 18, 2021.

R. Tomsett, K. Chan, and S. Chakraborty, “Model poisoning attacks against distributed machine learning systems,” in Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications, vol. 11006, p. 110061D, International Society for Optics and Photonics, 2019.

A. Huang, “Dynamic backdoor attacks against federated learning,” arXiv preprint arXiv:2011.07429, 2020.

R. Kerkouche, G. ´Acs, and C. Castelluccia, “Federated learning in byzantium,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 3310–3322, IEEE, 2019.

F. Nuding and R. Mayer, “Poisoning attacks in federated learning: A new attack approach and its defense,” in 2021 IEEE Wireless Communications and Networking Conference (WCNC), pp. 1–7, IEEE, 2021.

Z. Wang, Z. Zheng, M. Song, Q. Wang, A. Rahimpour, and H. Qi, “advpattern: Physical-world attacks on deep person re-identification via adversarially transformable patterns,” in Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 8341–8350, 2019.

L. Zhu and S. Han, “Deep leakage from gradients,” in Federated learning, pp. 17–31, Springer, 2020.

J. Zhang, J. Zhang, J. Chen, and S. Yu, “GuARD: A passive local attack in federated learning,” in 2020 IEEE International Conference on Communications (ICC), pp. 1–6, IEEE, 2020.

M. Nasr, R. Shokri, and A. Houmansadr, “Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning,” in 2019 IEEE Symposium on Security and Privacy (SP), pp. 3310–3322, IEEE, 2019.

L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, “Exploiting unintended feature leakage in collaborative learning,” Proceedings - IEEE Symposium on Security and Privacy, vol. 2019-May, pp. 691–706, 2019.

L. Lyu and C. Chen, “A novel attribute reconstruction attack in federated learning,” arXiv preprint arXiv:2108.09910, 2021.

M. Fredriksson, S. Jha, and T. Ristenpart, “Model inversion attacks that exploit confidence information and basic countermeasures,” in Proceedings of the 22nd ACM SIGSAC conference on computer and communications security, pp. 1322–1333, 2015.

F. Wu, “Pflg: A privacy attack method based on gradients for federated learning,” in International Conference on Security and Privacy in Digital Economy, pp. 191–204, Springer, 2020.

M. Fredriksson, E. Lantz, S. Jha, S. Lin, D. Page, and T. Ristenpart, “Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing,” in 23rd \{USENIX\} Security Symposium \{\{USENIX\} Security 14\}, pp. 17–32, 2014.

N. Carlini, F. Tramer, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. Brown, D. Song, U. Erlingsson, et al., “Extracting training data from large language models,” arXiv preprint arXiv:2012.07805, 2020.

H. Hu, Z. Sulicic, L. Sun, G. Dobbie, and X. Zhang, “Source inference attacks in federated learning,” arXiv preprint arXiv:2109.05659, 2021.
C. Chen, B. Kailkhura, R. Goldhahn, and Y. Zhou, “Certifiably-robust federated learning,” arXiv preprint arXiv:2103.01319, 2021.

J. Hong, H. Wang, Z. Wang, and J. Zhou, “Federated robustness propagation: Sharing adversarial robustness in federated learning,” arXiv preprint arXiv:2106.10196, 2021.

D. Shah, P. Dube, S. Chakraborty, and A. Verma, “Adversarial training in communication constrained federated learning,” arXiv preprint arXiv:2103.01319, 2021.

2021 IEEE/ACM Workshop on Information Forensics and Security, pp. 94–108, IEEE, 2021.

B. Xi, S. Li, J. Li, H. Liu, H. Liu, and H. Zhu, “Batt: Backdoor detection on federated learning in e-health,” in 2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQoS), pp. 1–10, IEEE, 2021.

A. Sharma, W. Chen, J. Zhao, Q. Qiu, S. Chaterji, and S. Bagchi, “Tesseract: Gradient flip score to secure federated learning against model poisoning attacks,” arXiv preprint arXiv:2110.10108, 2021.

A. Manna, H. Kasyap, and S. Tripathy, “Moa: Model agnostic defense against targeted poisoning attacks in federated learning,” in International Conference on Information and Communications Security, pp. 38–55, Springer, 2021.

C. P. Wan and Q. Chen, “Robust federated learning with attack-adaptive aggregation,” arXiv preprint arXiv:2106.08283, 2021.

S. Savazzi, M. Nicoli, and V. Rampa, “Federated learning with cooperating devices: A consensus approach for massive iot networks,” IEEE Internet of Things Journal, vol. 7, no. 5, pp. 4641–4654, 2020.

Q. Xia, W. Ye, Z. Tao, J. Wu, and Q. Li, “A survey of federated learning for edge computing: Research problems and solutions,” High-Confidence Computing, p. 100008, 2021.

D. Zhang, X. Chen, D. Wang, and J. Shi, “A survey on collaborative deep learning and privacy-preserving,” in 2018 IEEE Third International Conference on Data Science in Cyberspace (DSC), pp. 652–658, IEEE, 2018.

Z. Li, V. Sharma, and S. P. Mohanty, “Preserving data privacy via federated learning: Challenges and solutions,” IEEE Consumer Electronics Magazine, vol. 9, no. 3, pp. 8–16, 2020.

G. A. Kaisnis, M. R. Makowski, D. Ruckert, and R. F. Baren, “Secure, privacy-preserving and federated machine learning in medical imaging,” Nature Machine Intelligence, vol. 2, no. 6, pp. 305–311, 2020.

L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. D. Tygar, “Adversarial machine learning,” in Proceedings of the 4th ACM workshop on Security and artificial intelligence, pp. 43–58, 2011.

B. Biggio and F. Roli, “Wild patterns: Ten years after the rise of adversarial machine learning,” Pattern Recognition, vol. 84, pp. 317–331, 2018.

A. Kurakin, I. Goodfellow, and S. Bengio, “Adversarial machine learning at scale,” arXiv preprint arXiv:1611.01236, 2016.