Chapter 13
Protecting Against Data Leakage in Federated Learning: What Approach Should You Choose?

Nathalie Baracaldo and Runhua Xu

Abstract Federated learning (FL) is an example of privacy by design where the primary benefit and inherent constraint is to ensure data is never transmitted. In this paradigm, data remains with its owner. Unfortunately, multiple attacks capable of extracting private training data by inspecting the resulting machine learning models or the information exchanged during the FL training process have been demonstrated. As a result, a plethora of defenses have surfaced. In this chapter, we overview existing inference attacks to assess their associated risks and take a close look at the significant corpus of popular defenses designed to mitigate them. Additionally, we analyze common scenarios to help provide clarity on what defenses are most suitable for different use cases. We demonstrate that one size does not fit all when selecting the right defense.

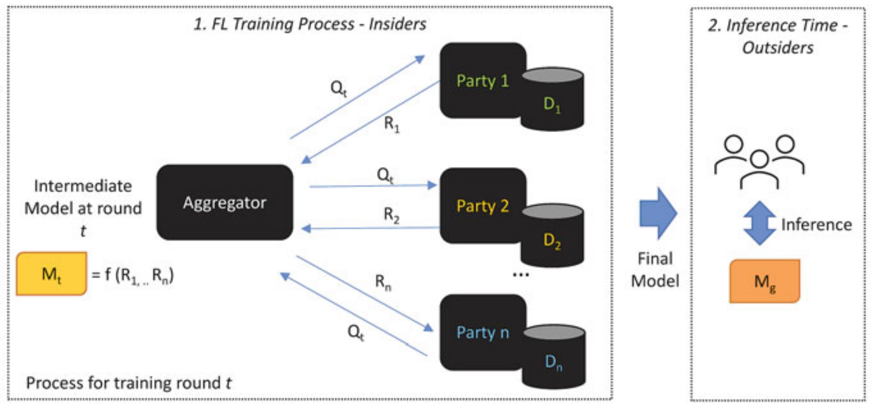
13.1 Introduction

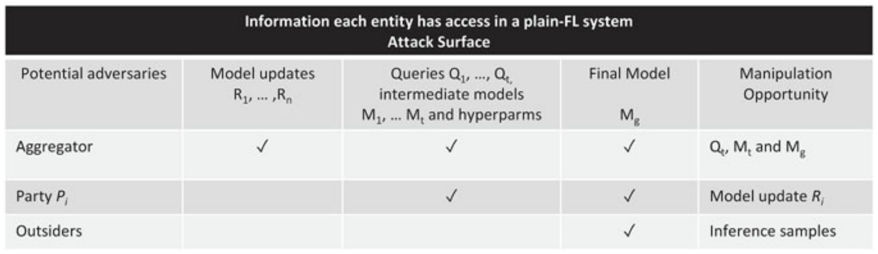
Privacy by design has been one of the main drivers of federated learning (FL), where each training party maintains their data locally while collaboratively training a machine learning (ML) model. Compared to existing ML techniques that require the collection of training data into a central place, this new paradigm represents a big improvement that balances utility of data and privacy. Given this significant change in data collection, FL is becoming a popular approach to protect data privacy.

In its basic form, Fig. 13.1, the FL training process requires an aggregator and parties to exchange model updates. Although no training data is shared during this process, this basic setup has been shown to be vulnerable to *inference* of private training data. Risks of inference of private information are prevalent in multiple stages of the FL process. Inference attacks can be classified based on the attack surface used to obtain private data into (1) those carried out on the final model

N. Baracaldo (✉) · R. Xu
IBM Research – Almaden, San Jose, CA, USA
e-mail: baracald@us.ibm.com; runhua@ibm.com

H. Ludwig, N. Baracaldo (eds.), *Federated Learning*,
https://doi.org/10.1007/978-3-030-96896-0_13





Information each entity has access in a plain-FL system
Attack Surface

Potential adversaries	Model updates $R_1, \ldots, R_n$	Queries $Q_1, \ldots, Q_t$, intermediate models $M_1, \ldots M_t$ and hyperparms	Final Model M_g	Manipulation Opportunity
Aggregator	✓	✓	✓	Q_t, M_t and M_g
Party P_i		✓	✓	Model update R_i
Outsiders			✓	Inference samples

Fig. 13.1 FL system with n parties $P_1, \ldots P_n$ without any privacy protection techniques. All entities participating in the FL training process are considered insiders while entities getting access to the final predictive model are consider outsiders. The table shows the attack surface for each of the entities; this information is available for multiple rounds

produced by the federation and (2) those executed using information exchanged during the FL training process.

Attacks executed exploiting the final model are applicable to *all* ML models and are not exclusive to FL. In other words, threats of inference based on a model itself are inherent to ML *regardless* of how the model was trained. Attacks in this category include data extraction, membership inference, model inversion, and property inference attacks, e.g., [12, 57, 70]. All of them infer information about the training data and in some cases the training data itself.

The second category of attacks are *specific* to FL and represent a new threat. These attacks take advantage of the information transmitted during the FL training process. For example, attack techniques such as [28, 39, 89, 91] use the newly exchanged information, exercise the potential for manipulating the training process or both. In some cases, this new information creates more successful attacks, while in some other cases the threats are not as worrisome as attacks are only applicable to some artificial settings. In this chapter, we present and contrast the differences and similarities of carrying out attacks based on the model and the FL process in Sect. 13.2.

The existence of inference risks have led to the proposal of a plethora of defenses that aim to provide different privacy guarantees. These include the introduction of differential privacy at the party side [30, 80, 84] and at the aggregator side, the integration of different cryptosystems to enable secure aggregation including Pailler [60], Threshold Pailler [19], different flavors of functional encryption [2–4, 8, 11], pairwise masking [10], homomorphic encryption (HE) [29], or the combination of multiple of those. The number of options available is substantial, and some of them have been shown to be vulnerable to inference attacks on themselves.

With this plethora of attacks and defenses, important questions remain to be answered: *How critical are inference attacks? How much should we worry about them? If we need to mitigate the risk, what defense should we use?*

In this chapter, we set out to help answer these questions. Selecting a defense technique for privacy preservation is not an easy task, and in the current landscape, it is challenging to compare the assumptions, disadvantages, and advantages of different solutions.

We show that *one size does not fit all* and that it is imperative to consider what initially may be seen as nuances of different defenses. In fact, as it is often the case in the security and privacy disciplines, a thorough risk assessment is needed to identify a suitable threat model for a particular application[1]. In some cases, more than one defense needs to be applied to mitigate stringent privacy requirements.

Unfortunately, there is *no free lunch* when incorporating defenses into the FL process. Incorporating them may result in higher training times, lower model performance, or more expensive deployments. For this reason, it is imperative to decide the *right* level of protection for a particular federation. This sweet spot highly depends on the scenario where FL has been applied to. For example, consider a use case where a single company has stored its data in multiple clouds and wants to run FL to obtain a ML model. Here, potential inference attacks may not be relevant. However, if multiple competitors decide to train a model together, there may exist mistrust and there may be an incentive for them to try to infer private information of some other participating parties. In other cases, regulation and compliance requirements may require additional protections, as it is the case for healthcare data and personal information [14, 63].

In this chapter, our aim is to provide clarity in the area of inference attacks to FL and to help provide some understanding of which attacks are relevant and which defenses are suitable. In Sect. 13.2, we begin by presenting in detail the system entities, assumptions and attack surfaces of the FL system as well as the potential adversaries. We then provide a thorough overview of relevant state-of-the-art inference attacks. Then, in Sect. 13.3 we characterize existing defenses in terms of functionality offer, disadvantages, and existing vulnerabilities. To this end, we carefully define various privacy goals and then compare defenses based on the

[1] A threat model defines the trust assumptions place in each of the entities in a system. For example, it determines if parties fully trust the aggregator or whether they only trust it to a certain degree.

different privacy guarantees offered and the attacks they can prevent. In some cases, we also discuss potential vulnerabilities of some defenses. In Sect. 13.4 we discuss some guidelines that demonstrate how to map threat models to some of the defenses and subsequently conclude our chapter.

13.2 System Entities, Attack Surfaces, and Inference Attacks

Figure 13.1 presents an overview of the entities involved during the training and inference process. We categorize relevant entities as *insiders* and *outsiders*, where insiders are entities involved during the training process while outsiders are entities that uniquely have access to the final model produced by the federation.

There are two main insider entities involved in the FL training process: the *parties* who own the data and the *aggregator* who orchestrates the learning process. Figure 13.1 presents a *plain FL* system; we refer to it as plain because no privacy-preserving provisions have been added to it. The federation consists of n parties P_1, P_2, $\ldots P_n$, where each party P_i owns its private dataset D_i. The training process starts with the aggregator sending *queries* Q_t requesting aggregated information needed to build a ML model to the parties, and the parties use their training data to answer that query. We refer to the reply sent back to the aggregator as a *model update* and denote the model update of party P_i as R_i. When the aggregator receives all the replies R_1, $R_2 \ldots R_n$., it proceeds to fuse or aggregate them creating a new *intermediate global model*. At the end of round t, the intermediate global model is denoted as M_t. The global model is then used as input to create the next query for the parties and the process repeats until the target model performance is reached.

Most privacy attacks are designed for gradient descent and its variants, which can be used to train neural networks, linear models, support vector machines (SVMs), among other model types. Hence, we briefly revisit how this algorithm is applied in FL. For a training round t, the aggregator sends the initial weights of the model to each party. Then, each party P_i runs gradient descent or stochastic gradient decent (SGD) using its own dataset D_i for a pre-specified number of epochs and hyperparameters. The party then sends back the resulting model weights, which are referred to as *model updates*. Alternatively, the party may send gradients as model updates.[2] The aggregator in its simplest form takes all the model updates and fuses them. The most popular way to aggregate model updates is to use FedAvg [53] where the aggregator uses the number of samples of each party to compute a weighted average of the model updates. The new model weights are sent back to the parties, and the process repeats until the target accuracy is reached or the maximum

[2] Experimentally, we have found that exchanging model weights leads to faster convergence than exchanging gradients.

rounds elapses. Notice that the fusion model updates obtained by the aggregator can be used to acquire the ML model.

13.2.1 System Setup, Assumptions, and Attack Surfaces

To fully understand the attack surface in FL and the existing risks, we begin by clearly defining the assumptions commonly made about the FL distributed system setup and the information that is typically assumed to be known by insiders before the training process begins.

From the machine learning perspective, there are multiple common assumptions. Before FL starts, it is often assumed that data at each of the parties is already annotated, that the set of labels are known, and that the model definition is available [24, 50]. Thus, prior to the federation starting, the aggregator and the parties know that there are, for example, five classes, and that each of the parties has pre-processed its dataset in the same fashion. If structural data is used, all parties and aggregator also know the number, order of the feature set, categorization, and the pre-processing technique to be used. Accordingly, in this chapter, we assume this information is known by all insiders participating in the FL process. Additionally, the aggregator has access to the *model updates* sent by all the parties and the intermediate models, while parties have access to their training data, aggregated models, and model updates.

From the system perspective, it is important to set up the distributed system in a secure way. In particular, throughout the FL process, messages exchanged between parties and the aggregator need to be properly authenticated and secure channels need to be established. In this way, external entities are prevented from snooping messages exchanged between the aggregator and parties. Man-in-the middle attacks where an adversary may try to impersonate a targeted party or the aggregator are also prevented. Hence, in the following discussion we assume external entities cannot snoop private information by inspecting exchanged messages or impersonate insiders.

13.2.2 Potential Adversaries

In FL, the training data is not explicitly shared. Hence, we focus on inference threats. The *goal* of an adversary is to infer private data based on the information they have access to (we will elaborate on the attackers' goals in more detail later as they differ for each attack).

The table in Fig. 13.1 summarizes the information that is available for each potential adversary and the information they can manipulate to their advantage. As we can see, a malicious aggregator has the opportunity to manipulate the aggregation process producing incorrect intermediate models and queries. It may also manipulate the final model. Similarly, a malicious party may manipulate the model updates sent to the aggregator. Finally, outsiders have the power to manipulate the inference queries to the final model to try to infer private information.

The adversaries can be further characterized according to their behavior as follows:

- A *Curious and passive aggregator* may try to infer private information based on all model updates exchanged during the training process. If the final or intermediate models are known, this adversary may also try to use the model to infer private data from participants.
- A *curious and active aggregator* may utilize the same information to perform inferences but may try to manipulate the queries sent to each party, the intermediate and final model updates.
- A *curious party* may try to infer private information based on the queries received and its knowledge about the learning objective.
- A *curious and active party* may try to infer private information based on the queries received and may also carefully craft the attacks to try to further determine information about other parties' private data.
- *Colluding parties* are conspiring parties who may try to infer information by orchestrating attacks together. These parties may share information among them and may actively or passively attack the system to infer private information about other parties.
- *Curious colluding aggregator and parties* it is possible for a curious aggregator and a few malicious parties to try to passively or actively launch privacy attacks to infer information about the training data and properties of a targeted party(ies).
- *Curious outsiders* are impersonated by external adversaries who do not participate in the training process but have access to the final ML model. They may try to use the final model and its predictions to infer private training data or related information.

Any such adversaries may carry out attacks to try to infer private information. In the following we overview multiple inference attacks.

13.2.3 Inference Attacks to Federated Learning

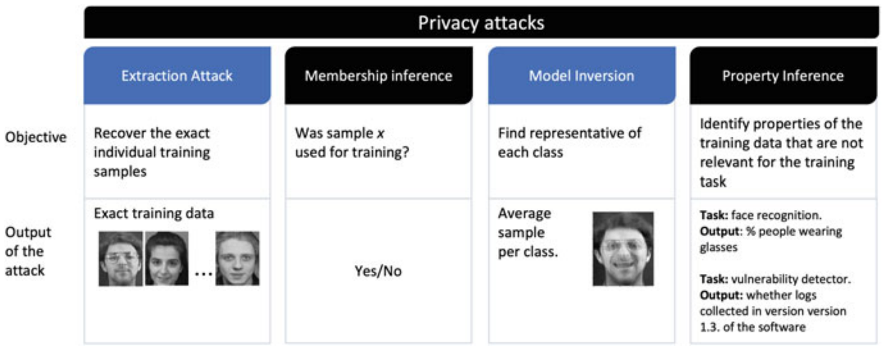
Multiple inference attacks have been demonstrated in FL systems. They can be classified according to their inference objective in the following four categories:

1. *Training data extraction:* These attacks aim to recover the *exact* individual training samples used during the training process. In other words, for an image based task, the output of the attack is a pixel-by-pixel output of the training data, while for a text-classification task, the output of the attack is the word-by-word text of the training corpus.

2. *Membership Inference:* In this type of attacks, the objective of the adversary is to determine if a particular sample was part of the training data used to create a model. This clearly constitutes a privacy violation if being part of a training set reveals, for example, a medical condition, social or political association.
3. *Model Inversion:* Here, the goal of the adversary is to construct a representative of each of the classes. This type of attack leads to great privacy violations in cases where each class contains samples that are similar among them. Consider the case of a face recognition model where a class contains information about a single individual. In this case, the results of the model inversion are visually similar to images of the person that were used during the training process. However, if the class members are *not* all similar, the results do not look like the training data [70], and hence this attack may produce innocuous results.
4. *Property Inference:* This type of attack focuses on revealing properties of the training data that are not relevant for the training task at hand. Attacks in this settings can extract global properties of the training dataset such as the ratio of the samples included or can focus on extracting properties of sub-populations of the training data.

We summarize the goals and outputs of these four types of attacks in Fig. 13.2. A variety of attacks have been demonstrated. In Table 13.1 we characterize representative attacks based on the adversaries that execute them and the information that they use to infer private training data. Again, we emphasize that attacks which can be carried out by outsiders are not specific to FL.

We now briefly describe in some detail how the attacks are carried out. This will help understand the vulnerabilities of the system and determine what defenses are more suitable to mitigate different attacks in Sect. 13.3.



	Privacy attacks			
	Extraction Attack	Membership inference	Model Inversion	Property Inference
Objective	Recover the exact individual training samples	Was sample *x* used for training?	Find representative of each class	Identify properties of the training data that are not relevant for the training task
Output of the attack	Exact training data …	Yes/No	Average sample per class.	**Task:** face recognition. **Output:** % people wearing glasses **Task:** vulnerability detector. **Output:** whether logs collected in version version 1.3. of the software

Fig. 13.2 Inference attacks to machine learning. The figure contrasts the different objectives of each attack and presents sample outputs for different attacks. We use the faces ORL dataset of the AT&T Laboratories Cambridge for illustration purposes. The output for the model inversion attack was generated using the attack presented in [26]

Table 13.1 Inference attacks to undefended FL systems

Attacks	Attack surface		Adversary
	Model	**Model update**	
Training data extraction attacks			
Henderson et al. [33], Carlini et al. [12], Zanella et al. [87], Carlini et al. [13]	Black-box		Outsider, insiders
DLG [91], iDLG [89], Geping et al. [28], Cafe [39]		Gradients	Curious aggregator
Wang et al. [81], Wei et al. [83]	White-box	Gradients	Curious aggregator
Song et al. [75]		Gradients	Aggregator active and passive modes
Membership attacks			
Shokri et al. [70], Salem et al. [68], Hayes et al. [32], Choquette et al. [16]	Black-box		Outsider, insiders
Nasr et al. [57] (multiple attacks)	Black-box		Curious aggregator or curious parties
		Gradients	Active curious party
		Gradients	Active curious aggregator
		Gradients	Active isolating aggregator
Model inversion			
Fredrikson et al. [25, 26]	Black-box		Outsider, insider
Hitaj et al. [34]		Gradients	Curious party
Property attacks			
Ateniese et al. [7], Ganju et al. [27]	White-box		Outsider, insider
Melis et al. [54]		Gradients	Curious party

13.2.3.1 Training Data Extraction Attacks

Multiple attacks have been proposed to extract data samples used during the training process by querying a published model [12, 13, 33, 73, 87]. Most of them are devoted to extracting training data from neural networks [12], text-based models [13, 87], and embeddings [73], but there are some approaches for other types of models such as [33].

Data memorization of generative text models has been studied for traditional and FL settings in [12] and [78], respectively. Generative text models help users to auto-complete their phrases to speed texting and email writing. The adversary's goal in this setting is to extract secret sequences of text to learn private information; for example, inputting to the system the sentence *my social security number is* may lead to the exposure of social security numbers used during the training process. In [12]

Carlini et al. proposed metrics to measure memorization in these types of models when the adversary has black-box[3] access to a published model that can query at will. Later, Thakkar et al. [78] compare differences between the memorization exhibited by models trained in FL versus those trained in traditional fashion. Their experiments suggest that training in FL may help reduce memorization and they hypothesize the diverse distribution of data among users is responsible the situation. However, both studies are based on black-box access. That is, they only consider the role of an outsider. Further analysis is required to identify how much information can be leaked when the adversary has more information, as it is the case for FL insiders.

A substantial number of data extraction attacks in FL for general models have been highlighted and the great majority make use of model updates that contain gradients [5, 28, 39, 81, 89, 91]. To understand how these attacks are possible, consider a simple text-based classifier that uses a bag-of-words as a way to encode the training data. Recall that once a party has been queried with an initial model, it uses its local training data to train for a few epochs and then returns the gradients to the aggregator. A curious aggregator may observe the gradients received from a party, which may be sparse, and can trivially know whether a word was part of the training data because a non-zero gradient is only possible if an original word was present in the party's training data [53]. Gradient-based attacks have also been demonstrated for images including number digits [5], faces [91], and other datasets [28].

An efficient attack known as *Deep leakage from gradients* (DLG) was proposed in [91], where a curious aggregator can obtain the training samples and the labels of a victim party. DLG was demonstrated to have high attack success rates in a few optimization interactions for some network topologies; however, it does not work unless the batch size utilized is small, for example, eight. A subsequent attack [81] increases the recovery accuracy for more general model initialization by using a distance metric based on a Gaussian kernel based on gradients. However, the approach also has limited attack success rates for larger batch sizes. This is clear a limitation that has raised the speculation on whether using higher batch sizes could deter the problem.

Unfortunately, increasing the batch size to prevent these attacks has been shown to be a naive defense in [39], where a new gradient-based attack called CAFE was presented. CAFE's attack success was demonstrated even under larger batch sizes of 40. The attack uses a surrogate or synthetic dataset created offline. By assuming the model is known, which is clearly the case for the aggregator, the aggregator can pass the surrogate data through the model and compute fake gradients. After that, the aggregator compares the resulting surrogate gradients with the real gradients coming from the victim party and tries to minimize the difference by updating its surrogate

[3] Black-box access in ML refers to a scenario where the adversary cannot access model parameters and can only query the model. White-box access, conversely, refers to settings where the adversary has access to the inner-works of the model.

dataset. This process can continue over the course of training to recreate the original dataset. An improvement in label guessing was presented in [89]. Additionally, studies and approaches such as the one presented by Geiping et al. [28] have shedded light into the vulnerabilities of commonly used vision network architectures beyond the topologies originally considered by DLG. Other attack optimizations inspired by the original DLG continue to emerge in the literature.

While most attacks are based on gradient-based inferences, gradient-based algorithms in FL are frequently trained exchanging model weights. As mentioned before, we have experimentally observed that exchanging model weights leads to faster convergence times. In some circumstances, a curious aggregator could still carry out the same inference procedure by computing the gradient between two iterations with the same party. This, however, would require for the victim party to be queried in subsequent training rounds that are not too far apart. Hence, further experimental analysis in this direction is needed to evaluate the attacks under those system setups. Finally, as our overview of these attacks shows, there is a trend to continuously improve limitations of existing data extraction attacks.

13.2.3.2 Membership Inference Attacks

Multiple membership inference attacks [57, 68, 70] have demonstrated that it is possible to know if a sample, called a *target sample*, was part of a training dataset, violating the privacy when the inclusion in a training set is itself sensitive. For example, consider an ML model trained based on photographs of people who suffer from a taboo disease or a political or group affiliation. In this case, getting to know if a person is part of the training set will reveal its health condition or that it is part of a particular group.

Attacks in this category take advantage of the notoriously higher prediction confidence that models exhibit when they are queried on their training data. This is mainly caused by overfitting of the model to training samples. Traditional attacks in this category assume the adversary has black-box access to the model (does not know the model parameters) and can only query the model for prediction. The adversary queries the model multiple times to understand how it behaves with respect to a set of engineered inputs to reconstruct the loss surface of the model. After multiple such queries, the adversary can determine if a sample was part of the training dataset. Most attacks in this classification make use of the confidence scores associated with a prediction query to guide the creation of informative queries.

Because a variety of attacks make use of the confidence score to achieve their objective, some attempts to hide confidence scores or reduce the number of queries a classifier can answer have been proposed as potential mitigation techniques [38, 68, 85]. However, those solutions do not effectively prevent membership inference. It has been demonstrated that even when a model does not expose the confidence scores associated with its classification, it is possible for adversaries to successfully execute a membership attack [16]. In the label-only attack presented in [16], for a target sample, the adversary generates multiple perturbed samples and queries the classifier to determine the robustness of the model to modifications. Data augmentation and adversarial samples are used to generate the perturbed samples. Based on the classifier's replies, it is possible to determine if there is a strong or weak membership signal.

Membership inference attacks can also be carried out in FL based on messages exchanged. In particular, Nasr et al. [57] present a variety of attacks that take advantage of the model updates exchanged during the training process. All the attacks proposed in [57] inspect the gradients of each of the neural network layers separately to take advantage of the fact that higher layers of the neural network are fine-tuned to better extract high level features that may reveal private information at higher rates than lower level layers of the network. To make decisions on whether the network was trained using the target sample, the attack may use an unsupervised inference model based on auto-encoders or a shallow model for scenarios where the adversary has some background knowledge on the victim's data.

Nasr et al. propose attacks that can be launched by a curious aggregator or a curious party in *active* or *passive* mode. A *curious aggregator* can carry three types of attacks (1) a passive attack where it observes each of the model updates of the parties and tries to determine membership of a target sample for that party, (2) an active attack where the aggregator manipulates the aggregated model according to the previous discussion, and (3) an active and isolating attack where the curious aggregator does not aggregate the model updates of other participants to increase the attack success rate.

The attack proposed for malicious parties is limited with respect to the one launch by an aggregator because parties can only see intermediate models. Hence, the object of a curious party is limited to determine if *any* of the other parties has the target sample. For malicious parties, the attack can be passive (no manipulation of the model update) or active.

Active membership attacks are exclusive to FL; an active attack refers to whether the adversary manipulates the model updates or model to induce observable gradient behaviors on member samples. In the case of an active curious party trying to infer the membership of a target sample x, the adversary will run gradient *ascent* on sample x by the model and update its local model in the direction of the increasing loss on the sample. This modified model weights are shared with the aggregator, who then sends the new model updates with the parties. Interestingly, when a party has sample x, its local SGD will abruptly reduce the gradient of the loss on x. This signal can be captured to infer the party has the target sample by a supervised or unsupervised inference model. Because multiple rounds are required to train a model, the adversary has several opportunities to manipulate the model updates, which leads to higher attack success rates.

In conclusion, FL does offer new opportunities for adversaries to carry out membership inferences attacks.

13.2.3.3 Model Inversion Attacks

The goal of model inversion attacks is to construct representative of each of the classes. Model inversion attacks to traditional training processes include [25, 26]. Typically these attacks use the confidence score output by the model to guide the reconstruction of data samples of a known and targeted label. For example, Fredrikson et al. showed that an adversary with access to the model and some demographic information about a patient can predict the patient's genetic markers [25].

Model inversion attacks have been tailored to FL in [34], where Hitaj et al. proposed a Generative Adversarial Network (GAN)-based procedure that operates on model updates and forces victims to reveal more information by carefully crafting gradients. The attack can be carried out by any participating party, where such curious party aims to gain information about a target label. The adversary trains a GAN based on observed model updates. The GAN subsequently generates prototypical samples of the private training set. The curious party can create inputs to the training process that force the victim to release more accurate private information. Concretely, the adversary will generate errors for recovered samples, ensuring the victim party tunes the model and reveals further information about the target sample. Here, the adversary actively manipulates the training process.

Finally, model inversion attacks are particularly dangerous if there is a meaningful *average* representation of a class. Otherwise, the output may not provide useful information to the attacker.

13.2.3.4 Property Inference Attacks

An adversary can infer information about the properties of the inputs, such as the environment where the data was produced, even when the model task is completely independent of that extracted information. Some approaches have focus on extracting *global properties* of the training set [7, 27] while more recent approaches have focused on extracting *sub-population properties* [54]. Global properties include the distribution of different classes in the training dataset, for example, a neural network trained to classify smiling faces may leak relative attractiveness of the individuals in the training set. Another less innocuous example of inference of global properties was presented by Ganju et al. [27] who demonstrated models may help an adversary determine whether a machine where the training logs were collected contained two important vulnerabilities that could lead the adversary to illicitly gain bitcoin by exploiting such vulnerabilities. Sub-population properties refer to properties of a particular sample in the training data, for example, in a model classifying medical reviews, an adversary may be capable of inferring the medical speciality from which they came [54].

Global property inference was first proposed by Ateniese et al. in [7], who demonstrated that SVMs and hidden Markov models (HMM) easily leak global properties. Their approach uses multiple meta-classifiers trained in surrogate datasets that exhibit the tested properties and requires white-box access to the model. After that, the approach was extended to work into fully connected neural networks in [27].

With the introduction of FL, attacks that can further isolate *sub-population properties* have been proposed. Melis et al. recently proposed a novel and more fine-grained attack that combines membership and property inference in FL settings in [54]. Here, a curious party can first identify the presence of a particular record in the training dataset of a victim party by running a membership inference attack. Similarly to global property attacks, this attack also trains meta-classifiers using auxiliary labeled data and uses them to determine if a property exist or not in one sub-populations. Experimental results show that the attack is successful for image and text-classifiers, and that are possible with two to 30 parties. In this case, FL opens attack surfaces for malicious insiders to leak more private information.

13.3 Mitigating Inference Threats in Federated Learning

While the amount of attacks to FL system keeps increasing, so do the number of solutions that is proposed to address these threats. Defenses in this area are diverse in nature and protect different aspects of the FL training process or deployment of the model. Broadly speaking, defenses in this area include:

1. *Modification of the training procedure and restrictive interfaces to query a model:* Examples in this category include pruning [37] or compressing the model updates sent to the aggregator to deter gradient-based attacks, adding regularizers to reduce overfitting [70] and prevent extraction attacks, minimize the number of queries or avoid reporting confidence of models to prevent membership attacks [85]. All these defenses have been demonstrated to fail under adaptive attacks [16, 83, 83]. Because these solutions have been demonstrated to lead to a false sense of privacy, we omit them in the following analysis.
2. *Syntactic and perturbation techniques:* Techniques in this category include adaptations of k-anonymity [76] and differential privacy (DP) [22], which have been incorporated to FL training process by multiple defenses including [80] and [17], respectively. We will see these approaches may help deter *some* of the attacks previously presented.
3. *Secure aggregation and secure hardware techniques:* This category includes approaches that use different cryptosystems to ensure the aggregator cannot access individual model updates, as well as approaches that require specialized hardware execution environments to ensure the execution flow is followed and the computation is maintained private.

How Do All These Techniques Compare? In the following, we characterize these techniques based on the information that is maintained private according to the following definitions which are illustrated in Fig. 13.3.

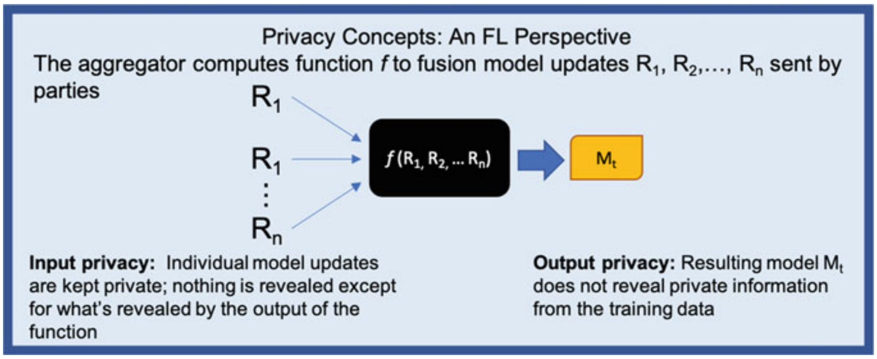

Privacy Concepts: An FL Perspective
The aggregator computes function f to fusion model updates R1, R2,..., Rn sent by parties
R1
R1
Rn
f (R1, R2, ... Rn)
Mt
Input privacy: Individual model updates are kept private; nothing is revealed except for what's revealed by the output of the function
Output privacy: Resulting model Mt does not reveal private information from the training data

Fig. 13.3 Illustration of the *input privacy* and *output privacy* concepts in the FL context. Function f computed by the aggregator fusions model updates $R_1, R_2, \ldots, R_n$ sent by parties

1. *Input Privacy*: Solutions that fall in this category preserve the privacy of the model updates shared by each of the parties. In other words, a solution provides privacy of the input if the FL training process does not reveal anything other than what can be inferred by the resulting final ML model.

   Techniques in this category can be used to protect against threats coming from malicious aggregators trying to isolate or infer private information from parties. Multi-party computation techniques are often used to maintain this type of privacy. However, we will see that these techniques come in many flavors and provide different guarantees and *some* are vulnerable to inference attacks.
2. *Output Privacy*: Techniques in this category ensure that the final model or intermediate models do not leak private information of the training data. Defenses that fall in this category are designed to prevent attacks coming from outsiders and insiders that make inferences base on the models. Differential privacy is a defense in this category, but as we will see, it has its own limitations.
3. *Privacy of the Input and Output:* Approaches in this category preserve both the privacy of the input and the output.

Based on these three definitions, we categorize some representative defenses in Table 13.2. In the table, we also added a column to highlight that existing techniques also differ on whether the intermediate or final models are exposed in plaintext to the aggregator. This reflects a different trust model that is important to select the right solution for different use cases. Note that even if the model is not revealed in plaintext to the aggregator, a solution may not protect the privacy of the output because the final model, when decrypted by the parties, is still vulnerable to output inference. In the following, we briefly present existing defenses and provide information about the attacks that they address and the ones they cannot thwart.

Table 13.2 Privacy-preserving defenses for Federated Learning. The second column refers to the intended privacy goal; we will show that some of the approaches may lead to privacy leakage if not deployed correctly. In particular, plaintext results may suffer from disaggregation inferences if not deployed in conjunction with additional provisions

	Privacy of		Aggregator's access to
Existing proposals	Input	Output	fusion model updates
Secure aggregation:			
– Partial HE [58, 88]	✓	✗	Encrypted
– Fully HE [67]	✓	✗	Encrypted
– Threshold Paillier (TP) [80]	✓	✗	Plaintext
– Garbled Circuit (GC)	✓	✗	Plaintext
– Pairwise Mask (PM) [10, 40, 72]	✓	✗	Plaintext
– TPA-based Functional encryption (FE) [84]	✓	✗	Plaintext
Differential privacy:			
– Global DP (Aggregator)	✗	✓	Plaintext
– Local DP (Party side) [6, 30, 80, 84]	✓	✓	Plaintext with DP Noise
Hybrid approaches			
– DP with Threshold Paillier [80]	✓	✓	Plaintext with DP Noise
– DP with Functional Encryption [84]	✓	✓	Plaintext with DP Noise
– DP with pairwise masking [42]	✓	✓	Plaintext with DP Noise
Special hardware support			
– Truda (three aggregators)[15]	✓	✗	Partial model in plaintext
Approaches specific for XGBoost			
– Pairwise mask [48]	✓	✗	Plaintext
– Oblivious TEE-based XGBoost [46]	✓	✗	Plaintext

13.3.1 Secure Aggregation Approaches

To achieve privacy of the input, secure multi-party computation has been proposed to achieve *secure aggregation (SA)*. SA allows an entity to compute a function f that takes as input $R_1, R_2, \ldots, R_n$ without getting to know any of the inputs. There are multiple ways to preform SA including pairwise masking and modern cryptographic schemes such as fully homomorphic encryption, functional encryption, partial homomorphic encryption, threshold Pailler, among others. These techniques differ in multiple aspects as shown in Tables 13.2 and 13.3. The key differences are:

- *Supported threat model:* The threat model they cover differs on the trust each of the SA techniques puts on the aggregator. Some approaches expose the intermediate models to the aggregator while others only expose encrypted data (see Table 13.2). Additionally, some techniques have verification provisions to

Table 13.3 Behavior of the solutions under dynamic party participation

Approach	Verified aggregation	New parties	Dropout	Special hardware
Secure aggregation				
– Partial HE [58, 88]	✓	No rekeying	✗	✗
– Fully HE [67]	✓	No rekeying	✓	✗
– Threshold Paillier [80]	✗	Rekeying	✓	✗
– Pairwise Mask [10, 40, 72]	✗	Rekeying	✓	✗
– HybridAlpha (TPA-based FE) [84]	✓	No rekeying	✓	✗
Special hardware support				
– Truda (three aggregators)[15]	✓	No rekeying	✓	✓
Federated boosted model				
– Secure Federated GBM [48]	✗	Rekeying	✓	✗
– Oblivious TEE-based XGBoost [46]	✓	No rekeying	✓	✓

ensure that the aggregator can only fusion a minimum number of model updates (see first column of Table 13.3).

- *Dynamic participation:* Another aspect where SA techniques differ is their adaptability to parties joining the federation in the middle of the training process and their tolerance to some parties leaving the federation intentionally or accidentally. Some techniques require fully halting the training process to re-key the system when a party drops or joins, while others are more resilient against these changes and can continue training without modifications (see third column Table 13.3). Clearly, techniques that do not require the entire system to be rekeyed are preferred.
- *Infrastructure:* Each approach may require a different infrastructure setup. Some of them require the use of additional fully trusted authorities or special hardware, while others need multiple non-colluding servers that may increase the cost of deployment a solution or special hardware.

In the following, we present in more detail each of SA techniques and discuss their advantages and shortcomings.

13.3.1.1 Homomorphic Encryption-Based Secure Aggregation

Homomorphic encryption (HE) can be classified into partial HE and fully HE depending on the types of operations that can be computed over encrypted data.

Partially Homomorphic Encryption schemes enable the computation of additive operations over encrypted data by an untrusted entity. Paillier cryptosystem [60] and its variants [56, 59] are some of the most used cryptosystems in this category. Partial

HE cryptosystems satisfy the following property:

$$\texttt{Enc}(m_1) \circ \texttt{Enc}(m_2) = \texttt{Enc}(m_1 + m_2).$$

Where $\texttt{Enc}(m_1)$ and $\texttt{Enc}(m_2)$ are encrypted and $\circ$ represents a predefined function. In other words, an untrusted entity receiving $\texttt{Enc}(m_1)$ and $\texttt{Enc}(m_2)$ can compute their addition without decrypting the m_1 and $m2$. Notice that the untrusted entity only obtains the result $\texttt{Enc}(m_1 + m_2)$ in encrypted form.

Because most aggregation functions in FL require uniquely additive operations, these cryptosystems are a popular option for FL [58, 88]. We now examine how this cryptosystem is applied to FL in more detail. During setup all parties agree upon a public/private key pair(pk, sk), and the aggregator receives a public encryption key pk. The parties encrypt their model updates using pk before sending them to the aggregator. Once the aggregator receives all encrypted model updates, it computes the addition of model updates using its public key pk obtaining the encrypted result. The encrypted result is then forwarded to the parties, who in turn decrypt them using sk and continue the training in plaintext.

Compared to other SA approaches, the final result of the aggregation is never revealed to the aggregator, who always obtains the fusion result encrypted. Additionally, no rekeying is needed in case of new parties join or drop the federation. However, according to [84], partial HE may result in high computation and communication costs compared to competing options. To overcome this downside, *BatchCrypt*, an approach where each party *quantizes* its gradient values into low-bit integer representations and then encodes a batch of quantized values to a long integer for encryption, was recently proposed in [88]. These pre-processing operations allow faster encryption/decryption times.

Approaches in this category are limited to additive operations. For this reason, fusion algorithms that require more than a simple average of the model updates cannot be implemented with partially HE schemes.

Fully Homomorphic Encryption schemes support all operations over encrypted data. Thus, they can be used to implement many more fusion algorithms. However, they are more expensive computationally speaking. One such approach was presented in [67].

All HE approaches by themselves are capable of preventing a curious aggregator from inferring data from the intermediate and final models. This is the case because the aggregator can only see this information encrypted. However, parties can decrypt the models and, hence, are still able to infer information from the model itself. Therefore, in Table 13.2, we have marked HE approaches as not providing privacy of the output.

13.3.1.2 Threshold Paillier-Based Secure Aggregation

The Threshold Paillier cryptosystem [19] is a variant of the Paillier encryption system [60] and, thus, supports the additive HE property. The main difference is that in this variant, a predefined number of trusted parties t are required to collaboratively decrypt the ciphertext.

The use of this cryptosystem in FL settings was proposed in [80] to prevent collusion attacks by making sure the final fusion result cannot be obtained unless t trusted parties collaboratively decrypt it. During setup, each participant obtains a public key pk for encryption and a private *share secret key* sk_i for partial decryption, and the aggregator receives a *combing decryption key* dk. Each participant uses pk to encrypt its model update and sends it to the aggregator. The aggregator fuses the received encrypted model updates and sends the result back to t parties. Each party then uses its sk_i to perform its partial decryption and sends it back to the aggregator. Finally, after the aggregator receives at least t partially decrypted results, it makes use of dk to acquire the aggregated model updates in plaintext.

Threshold Pailler approaches for secure aggregation are suitable for federations where the aggregator is trusted to obtain the aggregated model updates in plaintext. Additionally, they provide an interesting property that ensures that at least t trusted parties need to agree to decrypt the model update. However, this functionality comes at the cost of more communication rounds between parties and aggregator resulting in longer training times.

13.3.1.3 Pairwise Mask-Based Secure Aggregation

Each party conceals their model update using pairwise random masks between users to hide the individual input. Following that, the aggregator simply adds up those masked model updates to obtain the global model. The protocol is built in such a way that the pairwise random masks are cancelled out after all model updates are aggregated. The initial pairwise mask design [10] relies on *t-of-n secret sharing* [69] and requires four rounds of communication among parties and the aggregator. The overhead of [10] grows quadratically with the number of parties. To further improve the efficiency and scalability of [10], Turbo-Aggregate [72] employs additive secret sharing and a multi-group circular strategy for model aggregation, while FastSecAgg [40] proposes a novel multi-secret sharing scheme based on a finite-field version of the Fast Fourier Transform.

Although pairwise mask-based approaches support parties dropping out, applying these techniques increases the number of messages exchanged between parties and the aggregator increasing the training time. Additionally, the approach requires rekeying when new parties arrive to the federation.

13.3.1.4 Functional Encryption-Based Secure Aggregation

Functional encryption (FE) [11] is an emerging family of cryptosystems that allow the computation of a function f over a set of encrypted inputs, where the decrypter obtains the final result of f in plaintext. To compute the function, the decrypter entity needs to use a *functional key* that depends on the function evaluated f and the encrypted data. This functional key is provided by a trusted third party (TPA). However, recently, new cryptosystems are removing the need for this entity.

The use of functional encryption for FL was first proposed by Xu et al. in [84], where the concrete cryptosystem of choice is the multi-input inner-point functional encryption [31]. This cryptosystem enables the aggregator to obtain in plaintext the result of the function $f(x, y) = \sum_i x_i y_i$, where x_i corresponds to the private encrypted model update coming from party i and y_i corresponds to the weight used to fuse model update x_i. That is, when y is a vector with ones, all model updates have the same weight. When $y_i = 0$, it means that x_i will be ignored during the aggregation.

During setup, given a maximum number of supported parties, the TPA initializes the functional cryptosystem. Each party then receives its party-specific secret key sk_i from the TPA. During training, each party encrypts its model update using sk_i and sends the resulting ciphertext to the aggregator. After a predefined amount of time elapses, the aggregator fusions all encrypted model updates received and prepares vector y. Vector y is used to request a functional decryption key dk_y from the TPA to fusion the received model updates and obtain the result in plaintext. If some parties did not reply, the aggregator does not include a position for them in vector y. This enables for a clean manage of dropouts without rekeying or further communication rounds. Vector y is then sent to the TPA, who inspects y to ensure providing the functional key for the received vector would not allow for inference attacks where a malicious aggregator may try to isolate the model updates of one or very few parties. If y complies with a pre-specified number of replies, the functional encryption key is sent to the aggregator. Otherwise, the aggregator cannot compute the aggregated result.

A distinctive advantage of the approach proposed in [84] is that it provides support for verification of the number of aggregated model updates preventing attacks from a curious aggregator trying to isolate a few model replies. It also supports dynamic party dropout and new parties joining, without requiring more than one message exchanged between the parties and the aggregator compared to other SA approaches such as Pairwise Masking or threshold Pailler. On the downside, the approach proposed in [84] requires the use of a TPA. This limitation could be solved with new advances in the functional encryption field.

13.3.1.5 Summary Secure Aggregation

We overviewed representative SA approaches and highlighted their differences and shortcomings. Some of the approaches are limited to additive fusion algorithms and do not react fast to dynamic settings when parties join and drop requiring expensive rekeying operations. Another salient difference is the number of messages that need to be exchanged between the aggregator and parties to obtain the SA results, with functional encryption solutions requiring a single message, while pairwise mask solutions require four messages, and the threshold Pailler requiring three messages.

Another main difference between SA solutions is their trust assumption in the aggregator. Most approaches trust the aggregator with the SA results in plaintext (Table 13.2), with the exception of Partial HE schemes where resulting SA model updates are encrypted and remains unknown to the aggregator. In scenarios where the aggregator deploys the final model and has test data to score the model performance, having the model in plaintext is useful. For use cases in which the aggregator is fully untrusted, Partial HE-based techniques may be more suitable.

SA techniques where the resulting SA fusion results are in *plaintext* have been recently shown to be vulnerable to disaggregation attacks [45, 71]. In disaggregation attacks, a curious aggregator may use SA results from multiple rounds to infer the model of a single party. When the party's model is isolated, the curious aggregator can carry any attack presented in Table 13.1 that takes as input the model itself making SA pointless. To prevent this type of attack, careful sub-sampling of parties has been proposed in [71]. This approach adds an additional layer of protection and requires the integration of TEE to bootstrap trust (Sect. 13.3.3). We note, however, that the solution only works for federations with large number of parties. Therefore, given the current state of the art, to prevent disaggregation attacks in small federations, it is best to use SA techniques where the result is encrypted.

Finally, adequately employing SA defenses[4] can only prevent inference based on individual model updates. SA on itself does not prevent any of the attacks based on the final or intermediate models. For that purpose, syntactic and perturbation approaches have been proposed and we overview them in the following.

13.3.2 Syntactic and Perturbation Approaches

Defenses in this category include solutions that aim to prevent inference attacks that use the final or model updates. K-anonymity and differential privacy are among those inference prevention techniques.

13.3.2.1 K-Anonymity-Based Approaches

K-anonymity is a technique to anonymize data that relies in hiding records in groups of k [76], these groups are defined by quasi-identifiers which are information that may serve to identify a potential individual records. For example, zip code, age, gender and race are quasi-identifiers that may serve to identify an individual. Defining the quasi-identifiers requires identifying in advance what background information an adversary may use to identify records in the training data.

[4] By *adequate* we mean applying additional sub-sampling techniques required for SA approaches vulnerable to disaggregation attacks as previously explained.

An adaptation of k-anonymity for FL training was presented in [17], where Choudhury et al. claim the approach is legally compliant with privacy legislations in the US, Canada and Spain (2020). The approach works for tabular data and applies k-anonymity in each of the parties independently prior to training resulting in multiple anonymization schemas. When the model is going to be used at inference time, the input is pre-processed according to the closest k-anonymity schema of each party.

One positive aspect of this approach is the interpretability of the parameters used: hiding a sample in a group of k is a very intuitive concept. One of the mayor drawbacks of k-anonymity-based approaches is the fact that the construct depends on the defender's ability to effectively anticipate the background information an adversary will have. Hence, if the adversary has more background information than anticipated, privacy can be compromised as demonstrated in [51]; albeit these attacks have not been demonstrated in the FL setting. *Differential privacy* is an alternative approach that addresses this shortcoming.

13.3.2.2 Differential Privacy-Based Approaches

Differential privacy (DP) [21] is a mathematical framework design to provide a rigorous measure of information disclosure about individual records used in the computation of a function. A training algorithm is described as deferentially private if and only if the inclusion of a single sample in the training data causes only statistically insignificant changes to the algorithm's output.

The formal definition of DP is the following [21]: A randomized mechanism[5] $\mathcal{M}$ provides (ϵ, δ)*-differential privacy* if for any two neighboring database D_1, D_2 that differ only in a single entry, $\forall S \subseteq Range(\mathcal{M})$,

$$\Pr[\mathcal{M}(D_1) \in S] \leq e^{\epsilon} \cdot \Pr[\mathcal{M}(D_2) \in S] + \delta.$$

When $\delta = 0$, $\mathcal{M}$ is said to satisfy ϵ*-differential privacy*.

The smaller the ϵ, the higher the protection. The additive term δ allows for a relaxation of the definition and enables mechanisms to provide higher utility, in the case of FL, this means increasing the model performance. To create a differential private mechanism, noise proportional to the sensitivity of the output is added to the algorithm's output. The sensitivity measures the maximum change of the output caused by the inclusion of a single data sample. Laplacian and Gaussian mechanisms are popular functions to achieve DP and to reduce the sensitivity, clipping values is a common practice.

An extensive set of mechanisms to optimally add DP noise while training ML models when all data is in a central place are available in the literature, e.g., [1, 36, 74, 79]. In most cases, these mechanisms can be adapted to FL.

[5] A mechanism can be understood as an algorithm designed to inject DP noise.

There are three ways in which DP can be applied in FL settings: *local*, *centralized* and using a *Hybrid* approach that encompasses DP with SA. Applying one or the other addresses a different threat models.

- **Local DP** [41, 65] is applicable in settings where parties do not trust the aggregator. For this reason, each party independently adds noise to the model updates before sending them to the untrusted aggregator. The downside of this approach is that the amount of noise typically causes the model to perform poorly.
- **Global DP** is applicable when the aggregator can be trusted to add DP noise to the model. Global DP ensures that less noise is added resulting in higher model performance compared to local DP.
- **SA-and-Local DP Hybrid** [80] This approach was created to overcome the limitations of local DP to ensure faster convergence times and better model performance. It is applicable to settings where parties do not trust the aggregator. The approach consists of using a SA technique in combination to local DP. This combination ensures that the model updates are not visible in isolation to the aggregator. Hence, each party can add less noise to its model update while maintaining the same privacy guarantee that would result by simply applying local differential privacy. In other words, this hybrid approach ensures the same DP guarantee can be obtained while the amount of total noise injected is reduced by a factor of n, where n is the number of parties. A mathematical formulation of why this is the case can be found in [80]. The SA approach proposed in [80] was Threshold Pailler, but the technique also works for other types of SA as demonstrated in [84].

A very appealing feature of DP is the mathematical guarantee it provides without making assumptions on the background information an adversary may have. At the same time, challenges of applying DP include defining the right ϵ value required for a use case, and the fact that adding noise may result in lower performing models. While ideally ϵ values should not be higher than 0.1 and the recommendation is not higher than one [23], in reality simple queries require epsilon up to one and for classical ML models epsilons up to 10. For neural networks, ϵ values greater than 100 are common [16], raising concerns of the actual protection provided by adding DP. The difficulty in interpretability of ϵ and the large values have motivated the approach based on k-anonymity previously described in this section. However, to ease this difficulty, Holohan et al. [35] proposed a methodology to map the *Randomize response* survey technique [82] to epsilon values, which provides some interpretability to ϵ.

Contrary to popular believe, DP is not a silver bullet against *all* privacy attacks presented in Sect. 13.2. Membership inference, inversion attacks and extraction attacks can be prevented by DP [12]. Let us revisit the DP definition to understand why other attacks may not be prevented by adding noise. According to its privacy definition, DP provides a quantitative privacy guarantee for indistinguishably of individual samples. However, the general information on the population is still available. In fact, one may argue that this is the main reason one would apply DP in the first place: obtaining a good generalizable model without compromising the privacy of individual records. Another assumption of DP is that records in the same dataset are independent [18, 44], which may not be the case in FL or general ML settings. Based on its definition and assumptions, it has been shown that using DP does not deter property based attacks [7, 27, 54]. The reason for which DP does not work against property attacks is the fact that they rely on aggregated properties of the population, while DP focuses on protecting individual samples. It is important to understand this difference to avoid falling into a false sense of privacy.

Currently, DP is one of the best ways to protect the privacy of individual samples and is still an area of ongoing research. An interesting discussion of open challenges and pitfalls of applying DP in real situations has been presented by Domingo et al. in [20]. Additional research focuses on reducing the amount of noise under the interactive nature of FL, where multiple rounds of communication between the aggregator and parties are often required.

13.3.3 Trusted Execution Environments (TEE)

Another set of defenses rely on *trusted execution environments* (TEE) [46]. A TEE is a secure area of computer's main processor that is designed to protect the confidentiality and integrity of the code and data loaded inside. Examples TEEs include IBM Hyper Protect™ [62], Intel SGX™ [52], and AME Memory Encryption™ [43]. One of the core features of TEEs is their ability to perform *remote attestation*. Remote attestation allows a remote client to verify that a specific software version has been securely loaded into the enclave. Hence, TEEs are suitable to run code in untrusted environments while ensuring the code run is the expected.

Thanks to the secure attestation feature, it is possible to ensure an aggregator running on a TEE uses the expected code preventing active attacks where a malicious aggregator may deviate from the expected behavior. Hence, we can ensure that the aggregator combines a minimum number of model updates preventing active attacks that isolate one or a few model updates. It is also possible to ensure it performs Global DP adequately and that, in general, it follows the right protocol. If deploying special hardware is possible, using TEE at the parties side can also improve the security of the system ensuring all participants run the pre-specified source code. This, however, may not be easy to achieve in settings where parties are consumer devices, run in legacy hardware or there is limited budget.

Potential vulnerabilities of the TEE include side-channel attacks and *cukoo attacks* [64] that enable an adversary to compromise the privacy of the data loaded by the TEE. To prevent potential side-channel attacks Law et al. [46] redesigned an adaptation of XGBoost for FL to be data-oblivious. Finally, using a TEE requires special hardware and setting up correctly the keys in the system to prevent cukoo attacks that compromise the cryptographic keys of the system [64].

In [15], data extraction attacks where a curious aggregator takes advantage of the gradients exchanged are deterred by decentralizing the aggregation process. The solution, called *Truda*, changes the FL architecture by introducing three TEE-aggregators that receive a partial view of the model. Parties agree on what pieces of the model sent to each aggregator, and no aggregator obtains all the model. Truda works for fusion algorithms that only require average of model updates. More advanced algorithms such as the ones required to train tree-boosting models based or PFMN [86] cannot be adapted to this architecture.

13.3.4 Other Techniques for Distributed Machine Learning and Vertical FL

In addition to the privacy-preserving approaches discussed above, there exist other techniques designed for distributed ML that have slightly different architectures to FL. These include Helen [90], Private Aggregation of Teacher Ensembles (PATE) [61] and its variant [49], as well as SecureML [55], where data is distributed among two non-colluding servers who jointly train a model using two-party computation. Because their architectures are different to FL, we do not expand on them in this chapter.

In this chapter, we focus our attention to the horizontal FL case, where all parties have the same input data and, thus, can train locally their own models. Vertical FL (Chap. 18) and split learning (Chap. 19) work for different setups. In vertical FL, each party holds only a partial set of the features and only one party typically holds the label while in split learning a different piece of the model may be trained in different parties. Thus, a single party cannot train a model on their own. Inference attacks and defenses have also been presented for these different setups [39, 47]. The threats in these settings may differ, for example, label inference is a potential attack [47]. Determining their vulnerabilities and designing defenses is still an open question.

13.4 Selecting the Right Defense

We have reviewed existing attacks and defenses and are now ready to define what defenses are applicable in different cases. Unfortunately, there is no free lunch when applying a defense as it was highlighted during their detail presentation in the previous section. Incorporating different defenses may lead to longer training times, lower performing models or more expensive deployment. Thus, it is necessary to find a sweet spot to prevent *relevant* attacks for each federation. We now analyze different scenarios.

13.4.1 Fully Trusted Federations

Consider a scenario where all the parties engage in the federation are owned by the same company. This case is embodied, for example, by a company who has stored data in different clouds, data warehouses, countries or has acquired other companies resulting in fragmented datasets. Another example of this type of scenario is a federation where the training task does not involve the use of sensitive data; yet, participants do not want to transfer the data to a single place due to its large volume.

These are low risk scenarios where there is no reason to mistrust each of the parties or the aggregator. Therefore, it is possible to employ plain FL without other protections, other than our assumed secure-and-authenticated channels. With respect to potential outsiders, it is possible to use Global DP to minimize attacks based on the final model. Otherwise, no DP needs to be added.

13.4.2 Ensuring that the Aggregator Can Be Trusted

There are multiple ways to ensure a federation can trust an aggregator. A common way to ensure this is the case is to run the aggregator by a trusted party. Alternatively, the aggregator may be run as a service where through contractual clauses trust may be achieved. The aggregator can also be required to be run using TEE so that parties can verify the aggregator is running the correct code through attestation. Such an aggregator as a service is a practical way to ensure fast deployment in consortium cases where all participants can find trusted company to host the aggregator.

In some cases, adding proper *accountability* to the process may also help boost trust in the aggregator and parties. Recently, an accountability framework for FL was proposed in [9], where all entities engage can subsequently audited if needed. It is in the best interest of a company running aggregators as a service to comply with its contracts, and it is even more critical to do so if it can be audit. Accountability services help ensure there is a way to verify different entities behaved as expected while offering a way for potentially mistrusting parties to verifying during the training process the system is behaving properly. Although accountability cannot prevent inference of information by inspecting the results of the well-executed process, it can help ensure the aggregator is a honest-but-curious adversary, meaning that it adheres to the protocol but may try to infer information based on the information it obtains in the process.

Whenever possible it is beneficial to trust the aggregator. One big advantage of this type of deployment is that the aggregator can offer additional features that require the inspection of individual replies. Among them are running robust learning algorithms that are resilient against noisy model updates or failures in the setup, as well as algorithms to detect and mitigate potential active attacks performed by misbehaving parties. In other words, a risk assessment to see what is more important for the federation is needed. In some cases, mitigating the aggregator's capabilities to perform inference by using the above listed techniques is deemed as enough mitigation to have it as an ally to prevent attacks from parties. In some other scenarios, the risk exposure may be unbearable.

13.4.3 Federations with an Untrusted Aggregator

In some cases, the aforementioned provisions may not be enough to trust the aggregator to obtain individual parties' model updates. For instance, the consumer space, where users may fear their private information is obtained by big companies. In these cases, SA techniques and DP can be applied. In fact, Google, Apple, and other consumer companies are already using local DP to provide privacy of the input and create trust among their users while enabling service improvement [66, 77].

Not all SA techniques offer the same protection. Different SA mechanisms are more suitable than others depending on whether the aggregator and other parties are mistrusted simultaneously. Let us analyze these cases.

The Aggregator Is Trusted to Offer Additional Functionality that Requires Access to the Model Consider the case where the federation wants to make use of an extended set of services offered by the aggregator that require this entity to access the model in plaintext. For example, the federation may want the aggregator to evaluate the performance of a model based on public data or to deploy the resulting model as a service. Solutions where the SA enables the aggregator to see the model in plaintext are adequate for these use cases.

We also highlight that SA mechanisms that enable the aggregator to obtain the model in plaintext need to be complemented with additional provisions to prevent disaggregation attacks. To mitigate disaggregation inference risks, it is necessary to ensure that *all* parties are selected and their model updates aggregated, or that sub-sample parties selections of multiple rounds do not lead to inference. Clearly, the first solution only works for small federations, while the second one can only be applied to large federations. For small federations, the modification is not particularly taxing, as it is typical to include all parties in all rounds to fully leverage their data.

To mitigate the risk of a malicious aggregator isolating replies of a few parties, HybridAlpha [84], the functional encryption-based SA presented in the previous section provides an inference module that verifies a minimum number of replies that have been aggregated before providing a functional key to obtain the model. This module prevents this type of attack. Another potential solution is to run in a TEE to verify the specified number of parties is indeed aggregated.

Now let us consider a federation where parties do not fully trust each other and are concerned other parties may try to obtain private information. Example scenarios in this category include multiple competitors collaborating to detect fraud, where each competing party may benefit from learning data about other parties. In the case the federation fears inference from different parties, for example, fearing attacks where a few curious parties may collude. In those cases, the solution presented in [80] which encompasses SA and DP is particularly useful. The cryptosystem of choice, threshold Paillier, allows for verification that ensures a subset of t trusted out of n total parties need to contribute to obtain the model in plaintext. As a bonus, the solution prevents inference by reducing the noise compared to local DP as outlined in the previous section.

The Aggregator Is Not Trusted with the Model In certain cases, a federation may find too risky to provide the model to the aggregator. In these cases, HE techniques are recommended. Notice that the final model is going to be accessible by parties owning the cryptographic keys. If inferences over the model are relevant, then DP may be added.

In the above analysis, we have avoided discussing particular regulations, as regulations keep evolving and, to date, there is no clear mapping between regulation requirements and technical solutions. This is a relevant open question that we expect will be solved as FL is increasingly applied in regulated settings.

13.5 Conclusions

FL is a privacy by design system that has substantially improved the state-of-the-art techniques that require transmitting private data to a central place. From the privacy perspective, there is a clear benefit over other approaches that move data, as the data can always remain with its rightful owner. Although, some inference attacks have been demonstrated, current defenses and research efforts can be incorporated to mitigate them. Inference of private data in FL systems is a relevant risk for *some* federations where exposing private data is an important consideration. In this chapter, we have overviewed the attack surfaces, the threats, and the defenses to help provide a holistic view of the risk inherent to participating in FL. We also presented multiple attacks characterizing them based on the attack surface, the objective the adversary had and also providing some details on how they may be carried out. As highlighted by our literature review, some attacks may not be realistic as simply changing hyperparameters of the training process can easily deter them, while in some other cases FL creates new relevant threats.

We also presented multiple defenses and highlighted their benefits and drawbacks showing that one size does not fit all. The details of the design of each defense imply various trust assumptions and made them suitable to different applications as they have inherently diverse computational and transmission costs. Matching the right level of protection to the use case is imperative to ensure only necessary overheads are incurred by adding defenses, while deterring relevant risks. Without a doubt, new attacks will emerge creating an arm's race between defenders and adversaries. As future work, enhancing different techniques to reduce overheads and understanding how different legislation and regulation can be mapped to concrete technologies is required. We hope this chapter has helped clarify the state of the art of attacks and available defenses to improve and facilitate the decision making process.

References

1. Abadi M, Chu A, Goodfellow I, McMahan HB, Mironov I, Talwar K, Zhang L (2016) Deep learning with differential privacy. In: Proceedings of the 2016 ACM SIGSAC conference on computer and communications security, pp 308–318
2. Abdalla M, Bourse F, De Caro A, Pointcheval D (2015) Simple functional encryption schemes for inner products. In: IACR international workshop on public key cryptography. Springer, pp 733–751
3. Abdalla M, Benhamouda F, Gay R (2019) From single-input to multi-client inner-product functional encryption. In: International conference on the theory and application of cryptology and information security. Springer, pp 552–582
4. Ananth P, Vaikuntanathan V (2019) Optimal bounded-collusion secure functional encryption. In: Theory of cryptography conference. Springer, pp 174–198
5. Aono Y, Hayashi T, Wang L, Moriai S et al (2017) Privacy-preserving deep learning via additively homomorphic encryption. IEEE Trans Inf Forens Secur 13(5):1333–1345
6. Asoodeh S, Calmon F (2020) Differentially private federated learning: An information-theoretic perspective. In: Proc. ICML-FL
7. Ateniese G, Mancini LV, Spognardi A, Villani A, Vitali D, Felici G (2015) Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers. Int J Secur Netw 10(3):137–150
8. Attrapadung N, Libert B (2010) Functional encryption for inner product: Achieving constant-size ciphertexts with adaptive security or support for negation. In: International workshop on public key cryptography. Springer, pp 384–402
9. Balta D, Sellami M, Kuhn P, Schöpp U, Buchinger M, Baracaldo N, Anwar A, Sinn M, Purcell M, Altakrouri B IFIP EGOV (2021) Accountable Federated Machine Learning in Government: Engineering and Management Insights (Best paper award), IFIP EGOV 2021
10. Bonawitz K, Ivanov V, Kreuter B, Marcedone A, McMahan HB, Patel S, Ramage D, Segal A, Seth K (2017) Practical secure aggregation for privacy-preserving machine learning. In: Proceedings of the 2017 ACM SIGSAC conference on computer and communications security, pp 1175–1191
11. Boneh D, Sahai A, Waters B (2011) Functional encryption: Definitions and challenges. In: Theory of cryptography conference. Springer, pp 253–273
12. Carlini N, Liu C, Erlingsson Ú, Kos J, Song D (2019) The secret sharer: Evaluating and testing unintended memorization in neural networks. In: 28th USENIX security symposium (USENIX Sec 19), pp 267–284
13. Carlini N, Tramer F, Wallace E, Jagielski M, Herbert-Voss A, Lee K, Roberts A, Brown T, Song D, Erlingsson U et al (2020) Extracting training data from large language models. Preprint. arXiv:2012.07805
14. Centers for Medicare & Medicaid Services: The Health Insurance Portability and Accountability Act of 1996 (HIPAA) (1996) Online at http://www.cms.hhs.gov/hipaa/
15. Cheng PC, Eykholt K, Gu Z, Jamjoom H, Jayaram K, Valdez E, Verma A (2021) Separation of powers in federated learning. Preprint. arXiv:2105.09400
16. Choquette-Choo CA, Tramer F, Carlini N, Papernot N (2021) Label-only membership inference attacks. In: Meila M, Zhang T (eds) Proceedings of the 38th international conference on machine learning, Proceedings of machine learning research. PMLR, vol 139, pp 1964–1974. http://proceedings.mlr.press/v139/choquette-choo21a.html
17. Choudhury O, Gkoulalas-Divanis A, Salonidis T, Sylla I, Park Y, Hsu G, Das A (2020) A syntactic approach for privacy-preserving federated learning. In: ECAI 2020. IOS Press, pp 1762–1769

18. Clifton C, Tassa T (2013) On syntactic anonymity and differential privacy. In: 2013 IEEE 29th international conference on data engineering workshops (ICDEW). IEEE, pp 88–93
19. Damgård I, Jurik M (2001) A generalisation, a simpli. cation and some applications of paillier's probabilistic public-key system. In: International workshop on public key cryptography. Springer, pp 119–136
20. Domingo-Ferrer J, Sánchez D, Blanco-Justicia A (2021) The limits of differential privacy (and its misuse in data release and machine learning). Commun ACM 64(7):33–35
21. Dwork C (2008) Differential privacy: A survey of results. In: International conference on theory and applications of models of computation. Springer, pp 1–19
22. Dwork C, Lei J (2009) Differential privacy and robust statistics. In: STOC, vol 9. ACM, pp 371–380
23. Dwork C, Roth A et al (2014) The algorithmic foundations of differential privacy. Found Trends Theor Comput Sci 9(3–4):211–407
24. FederatedAI: Fate (federated AI technology enabler). Online at https://fate.fedai.org/
25. Fredrikson M, Lantz E, Jha S, Lin S, Page D, Ristenpart T (2014) Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing. In: 23rd {USENIX} Security Symposium ({USENIX} Security 14), pp 17–32
26. Fredrikson M, Jha S, Ristenpart T (2015) 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
27. Ganju K, Wang Q, Yang W, Gunter C.A, Borisov N (2018) Property inference attacks on fully connected neural networks using permutation invariant representations. In: Proceedings of the 2018 ACM SIGSAC conference on computer and communications security, pp 619–633
28. Geiping J, Bauermeister H, Dröge H, Moeller M (2020) Inverting gradients–how easy is it to break privacy in federated learning? Preprint. arXiv:2003.14053
29. Gentry C (2009) A fully homomorphic encryption scheme. Ph.D. thesis, Stanford University
30. Geyer RC, Klein T, Nabi M (2017) Differentially private federated learning: A client level perspective. Preprint. arXiv:1712.07557
31. Goldwasser S, Gordon S.D, Goyal V, Jain A, Katz J, Liu FH, Sahai A, Shi E, Zhou HS (2014) Multi-input functional encryption. In: Annual international conference on the theory and applications of cryptographic techniques. Springer, pp 578–602
32. Hayes J, Melis L, Danezis G, De Cristofaro E (2017) Logan: Membership inference attacks against generative models. Preprint. arXiv:1705.07663
33. Henderson P, Sinha K, Angelard-Gontier N, Ke NR, Fried G, Lowe R, Pineau J (2018) Ethical challenges in data-driven dialogue systems. In: Proceedings of the 2018 AAAI/ACM conference on AI, ethics, and society, pp 123–129
34. Hitaj B, Ateniese G, Perez-Cruz F (2017) Deep models under the GAN: information leakage from collaborative deep learning. In: Proceedings of the 2017 ACM SIGSAC conference on computer and communications security, pp 603–618
35. Holohan N, Leith DJ, Mason O (2017) Optimal differentially private mechanisms for randomised response. IEEE Trans Inf Forens Secur 12(11):2726–2735
36. Holohan N, Braghin S, Mac Aonghusa P, Levacher K (2019) Diffprivlib: the IBM^{TM} differential privacy library. Preprint. arXiv:1907.02444
37. Huang Y, Su Y, Ravi S, Song Z, Arora S, Li K (2020) Privacy-preserving learning via deep net pruning. Preprint. arXiv:2003.01876
38. Jia J, Salem A, Backes M, Zhang Y, Gong NZ (2019) Memguard: Defending against black-box membership inference attacks via adversarial examples. In: Proceedings of the 2019 ACM SIGSAC conference on computer and communications security, pp 259–274
39. Jin X, Du R, Chen PY, Chen T (2020) Cafe: Catastrophic data leakage in federated learning. OpenReview - Preprint
40. Kadhe S, Rajaraman N, Koyluoglu OO, Ramchandran K (2020) Fastsecagg: Scalable secure aggregation for privacy-preserving federated learning. Preprint. arXiv:2009.11248
41. Kairouz P, Oh S, Viswanath P (2014) Extremal mechanisms for local differential privacy. Preprint. arXiv:1407.1338

42. Kairouz P, Liu Z, Steinke T (2021) The distributed discrete gaussian mechanism for federated learning with secure aggregation. Preprint. arXiv:2102.06387
43. Kaplan D, Powell J, Woller T (2016) Amd memory encryption. White paper
44. Kifer D, Machanavajjhala A (2011) No free lunch in data privacy. In: Proceedings of the 2011 ACM SIGMOD international conference on management of data, pp 193–204
45. Lam M, Wei GY, Brooks D, Reddi VJ, Mitzenmacher M (2021) Gradient disaggregation: Breaking privacy in federated learning by reconstructing the user participant matrix. Preprint. arXiv:2106.06089
46. Law A, Leung C, Poddar R, Popa R.A, Shi C, Sima O, Yu C, Zhang X, Zheng W (2020) Secure collaborative training and inference for xgboost. In: Proceedings of the 2020 workshop on privacy-preserving machine learning in practice, pp 21–26
47. Li O, Sun J, Yang X, Gao W, Zhang H, Xie J, Smith V, Wang C (2021) Label leakage and protection in two-party split learning. Preprint. arXiv:2102.08504
48. Liu Y, Ma Z, Liu X, Ma S, Nepal S, Deng R (2019) Boosting privately: Privacy-preserving federated extreme boosting for mobile crowdsensing. Preprint. arXiv:1907.10218
49. Liu C, Zhu Y, Chaudhuri K, Wang YX (2020) Revisiting model-agnostic private learning: Faster rates and active learning. Preprint. arXiv:2011.03186
50. Ludwig H, Baracaldo N, Thomas G, Zhou Y, Anwar A, Rajamoni S, Ong Y, Radhakrishnan J, Verma A, Sinn M et al (2020) Ibm$^{\text{TM}}$ federated learning: an enterprise framework white paper v0. 1. Preprint. arXiv:2007.10987
51. Machanavajjhala A, Kifer D, Gehrke J, Venkitasubramaniam M (2007) l-diversity: Privacy beyond k-anonymity. ACM Trans Knowl Discov Data (TKDD) 1(1):3–es
52. McKeen F, Alexandrovich I, Berenzon A, Rozas C.V, Shafi H, Shanbhogue V, Savagaonkar UR (2013) Innovative instructions and software model for isolated execution. In: Proceedings of the 2nd international workshop on hardware and architectural support for security and privacy, HASP '13. Association for Computing Machinery, New York
53. McMahan B, Moore E, Ramage D, Hampson, S, y Arcas BA (2017) Communication-efficient learning of deep networks from decentralized data. In: Artificial intelligence and statistics. PMLR, pp 1273–1282
54. Melis L, Song C, De Cristofaro E, Shmatikov V (2019) Exploiting unintended feature leakage in collaborative learning. In: 2019 IEEE symposium on security and privacy (SP). IEEE, pp 691–706
55. Mohassel P, Zhang Y (2017) Secureml: A system for scalable privacy-preserving machine learning. In: 2017 IEEE symposium on security and privacy (SP). IEEE, pp 19–38
56. Naccache D, Stern J (1997) A new public-key cryptosystem. In: International conference on the theory and applications of cryptographic techniques. Springer, pp 27–36
57. Nasr M, Shokri R, Houmansadr A (2019) Comprehensive privacy analysis of deep learning: Stand-alone and federated learning under passive and active white-box inference attacks. 2019 IEEE symposium on security and privacy (SP)
58. Nikolaenko V, Weinsberg U, Ioannidis S, Joye M, Boneh D, Taft N (2013) Privacy-preserving ridge regression on hundreds of millions of records. In: 2013 IEEE symposium on security and privacy. IEEE, pp 334–348
59. Okamoto T, Uchiyama S (1998) A new public-key cryptosystem as secure as factoring. In: International conference on the theory and applications of cryptographic techniques. Springer, pp 308–318
60. Paillier P (1999) Public-key cryptosystems based on composite degree residuosity classes. In: International conference on the theory and applications of cryptographic techniques. Springer, pp 223–238
61. Papernot N, Abadi M, Erlingsson U, Goodfellow I, Talwar K (2016) Semi-supervised knowledge transfer for deep learning from private training data. Preprint. arXiv:1610.05755
62. Park S, McMullen A (2021) Announcing secure build for ibm$^{\text{TM}}$ cloud hyper protect virtual servers. IBM$^{\text{TM}}$ Cloud Blog. https://www.ibm.com/cloud/blog/announcements/secure-build-for-ibm-cloud-hyper-protect-virtual-servers
63. Parliament E of the European Union C (2016) General data protection regulation (GDPR) – official legal text. https://gdpr-info.eu/

64. Parno B (2008) Bootstrapping trust in a "trusted" platform. In: HotSec
65. Qin Z, Yang Y, Yu T, Khalil I, Xiao X, Ren K (2016) Heavy hitter estimation over set-valued data with local differential privacy. In: Proceedings of the 2016 ACM SIGSAC conference on computer and communications security, pp 192–203
66. Radebaugh C, Erlingsson U.: Introducing TensorFlow privacy: Learning with differential privacy for training data. https://blog.tensorflow.org/2019/03/introducing-tensorflow-privacy-learning.html
67. Roth H, Zephyr M, Harouni A (2021) Federated learning with homomorphic encryption. NVIDIA^{TM} Developer Blog. https://developer.nvidia.com/blog/federated-learning-with-homomorphic-encryption/
68. Salem A, Zhang Y, Humbert M, Berrang P, Fritz M, Backes M (2018) Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models. Preprint. arXiv:1806.01246
69. Shamir A (1979) How to share a secret. Commun ACM 22(11):612–613
70. Shokri R, Stronati M, Song C, Shmatikov V (2017) Membership inference attacks against machine learning models. In: 2017 IEEE symposium on security and privacy (SP). IEEE, pp 3–18
71. So J, Ali RE, Guler B, Jiao J, Avestimehr S (2021) Securing secure aggregation: Mitigating multi-round privacy leakage in federated learning. Preprint. arXiv:2106.03328
72. So J, Güler B, Avestimehr AS (2021) Turbo-aggregate: Breaking the quadratic aggregation barrier in secure federated learning. IEEE J Sel Areas Inf Theory 2:479
73. Song C, Raghunathan A (2020) Information leakage in embedding models. In: Proceedings of the 2020 ACM SIGSAC conference on computer and communications security, pp 377–390
74. Song S, Chaudhuri K, Sarwate AD (2013) Stochastic gradient descent with differentially private updates. In: 2013 IEEE global conference on signal and information processing. IEEE, pp 245–248
75. Song M, Wang Z, Zhang Z, Song Y, Wang Q, Ren J, Qi H (2020) Analyzing user-level privacy attack against federated learning. IEEE J Sel Areas Commun 38(10):2430–2444
76. Sweeney L (2002) k-anonymity: A model for protecting privacy. Int J Uncertainty Fuzziness Knowl Based Syst 10(05):557–570
77. Team DP (2017) Learning with privacy at scale. Machine Learning Research at Apple^{TM}
78. Thakkar O, Ramaswamy S, Mathews R, Beaufays F (2020) Understanding unintended memorization in federated learning. Preprint. arXiv:2006.07490
79. Tramèr F, Boneh D (2020) Differentially private learning needs better features (or much more data). Preprint. arXiv:2011.11660
80. Truex S, Baracaldo N, Anwar A, Steinke T, Ludwig H, Zhang R, Zhou Y (2019) A hybrid approach to privacy-preserving federated learning. In: Proceedings of the 12th ACM workshop on artificial intelligence and security, pp 1–11
81. Wang Y, Deng J, Guo D, Wang C, Meng X, Liu H, Ding C, Rajasekaran S (2020) Sapag: a self-adaptive privacy attack from gradients. Preprint. arXiv:2009.06228
82. Warner SL (1965) Randomized response: A survey technique for eliminating evasive answer bias. J Am Stat Assoc 60(309):63–69
83. Wei W, Liu L, Loper M, Chow KH, Gursoy ME, Truex S, Wu Y (2020) A framework for evaluating gradient leakage attacks in federated learning. Preprint. arXiv:2004.10397
84. Xu R, Baracaldo N, Zhou Y, Anwar A, Ludwig H (2019) Hybridalpha: An efficient approach for privacy-preserving federated learning. In: Proceedings of the 12th ACM workshop on artificial intelligence and security, pp 13–23
85. Yang Z, Shao B, Xuan B, Chang EC, Zhang F (2020) Defending model inversion and membership inference attacks via prediction purification. Preprint. arXiv:2005.03915
86. Yurochkin M, Agarwal M, Ghosh S, Greenewald K, Hoang N, Khazaeni Y (2019) Bayesian nonparametric federated learning of neural networks. In: International conference on machine learning. PMLR, pp 7252–7261
87. Zanella-Béguelin S, Wutschitz L, Tople S, Rühle V, Paverd A, Ohrimenko O, Köpf B, Brockschmidt M (2020) Analyzing information leakage of updates to natural language models.

In: Proceedings of the 2020 ACM SIGSAC conference on computer and communications security, pp 363–375
88. Zhang C, Li S, Xia J, Wang W, Yan F, Liu Y (2020) Batchcrypt: Efficient homomorphic encryption for cross-silo federated learning. In: 2020 USENIX annual technical conference (USENIX ATC 20), pp 493–506
89. Zhao B, Mopuri KR, Bilen H (2020) IDLG: Improved deep leakage from gradients. Preprint. arXiv:2001.02610
90. Zheng, W Popa RA, Gonzalez JE, Stoica I (2019) Helen: Maliciously secure coopetitive learning for linear models. In: 2019 IEEE symposium on security and privacy (SP). IEEE, pp 724–738
91. Zhu L, Han S (2020) Deep leakage from gradients. In: Federated learning. Springer, pp 17–31