Is Federated Learning a Practical PET Yet?

Abstract—**Federated learning (FL) is a framework for users to jointly train a machine learning model. FL is promoted as a privacy-enhancing technology (PET) that provides data minimization: data never "leaves" personal devices and users share only model updates with a server (e.g., a company) coordinating the distributed training. We assess the realistic (i.e., worst-case) privacy guarantees that are provided to users who are unable to trust the server. To this end, we propose an attack against FL protected with distributed differential privacy (DDP) and secure aggregation (SA). The attack method is based on the introduction of Sybil devices that deviate from the protocol to expose individual users' data for reconstruction by the server. The underlying root cause for the vulnerability to our attack is the power imbalance. The server orchestrates the whole protocol and users are given little guarantees about the selection of other users participating in the protocol. Moving forward, we discuss requirements for an FL protocol to guarantee DDP without asking users to trust the server. We conclude that such systems are not yet practical.**

1. Introduction

Federated Learning (FL) [30] is a widely deployed protocol for collaborative machine learning (ML). It allows a server to train an ML model on data of different users without requiring direct access to that data. For this reason, FL is often promoted as data minimizing [31]: instead of sharing their data, users calculate model updates, usually gradients, on a shared model obtained from the server. These model updates are then aggregated and applied iteratively to train this shared model.

Prior work has demonstrated that the model updates leak sensitive information on the users' local training data [24], [39]. This is to be expected since nothing in the design of FL prevents information leakage. Indeed, the root cause of why FL is inherently vulnerable to data reconstruction attacks is that it is designed to provide confidentiality (data does not leave user devices) rather than privacy (outputs of the computation do not leak sensitive attributes from the users' input). Without additional privacy measures, FL cannot protect users from the server reconstructing their data.

Fortunately, *if* the server is trusted to follow the protocol as prescribed, a relatively cost-effective mitigation exists: the server can add noise to the model updates and, thereby, implement differential privacy (DP) [12] during the aggregation [32]. This degrades the performance of learned models [50], but adds strong privacy guarantees. Unfortunately, the server cannot always be trusted. Worse yet, by observing local users' model updates, an untrusted server can mount powerful attacks to reconstruct users' training data points [7], [14], [16], [49], [54], [56], [57]. Contemporary attacks exploit the fact that neurons compute a weighted sum of their input; thus the corresponding gradients contain rescaled versions of the input.

In this vein, this work raises a simple pragmatic question that has far-reaching consequences: *can modern FL deployments offer meaningful privacy guarantees while efficiently and effectively learning from thousands of users' data, if the server is* ***not*** *trusted?*

We investigate this by mounting an attack (see Figure 1) that exerts the capabilities of an untrusted server in FL to extract individual user data points. We show that the attack is even successful when FL is combined with secure aggregation (SA) [9] and distributed differential privacy (DDP) [46]. In SA, gradient aggregation is performed via a decentralized multiparty computation protocol. In DDP, each user adds a small amount of noise to their gradient updates. Through the aggregation of user updates, the cumulative noise of DDP provides sufficiently high privacy guarantees to protect user data from leaking sensitive information. The two techniques were designed to decrease the amount of trust placed by users in the central party in FL [2], [17], [27].

To be clear, we are, of course, not claiming that the cryptographic primitives behind SA and DDP are broken. We merely notice that the trust model that they assume, where in each round enough honest users contribute noised gradient updates, is not necessarily realized in practice within FL. In reality, the server is entrusted with *provisioning* users and *sampling* them in each round, and can easily inject an arbitrary number of malicious sybils under their control into any round, as demonstrated by industry actors actively deploying FL [40].

By sampling a target user together with a group of sybils that act maliciously, the server can acquire direct access to the target user's non-aggregated update. This allows the server to eliminate any effect of SA, and effectively reduces the protection of DDP to a minimum because the noise added by a single user is not designed to offer the claimed privacy guarantees.

Next, we observe that the server is also typically entrusted with *controlling the shared model*. Prior work [7], [14], [51] has shown that this ability enables the server to extract large amounts of the users' individual training data points from gradients in vanilla FL. By integrating the trap weight approach from [7] into our attack, we show that an untrusted server can extract high-fidelity user data points for common learning tasks despite DDP and SA. This highlights that *even elaborate combinations of techniques like DDP and SA with FL fail to prevent data reconstruction when the server's real-world capabilities are taken into account*.

While exploring data reconstruction under DP, we make another observation that advances our understanding of which users are more vulnerable to data reconstruc-

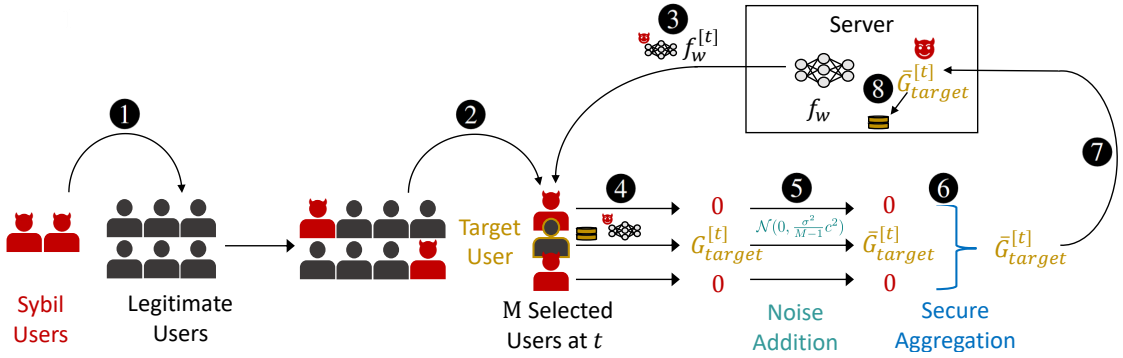


Figure 1: **Course of our attack against FL protected by SA and DDP.** ❶ The server introduces a small fraction of sybil users into the FL application. ❷ The server selects M users for participation in training round t: one target user and $\mathsf{M}-1$ sybils. ❸ The server manipulates the shared model with trap weights [7] and sends it out to the selected users. ❹ The target user locally calculates its gradients on the manipulated model while the sybil users return zero or constant value gradients that are known by the server. ❺ Only the target user locally applies a small amount of noise to its gradients to implement DDP. ❻ The target user's local noised gradients are aggregated with the sybil users' values. ❼ The resulting aggregate which effectively contains solely the target user's gradients is sent to the server. ❽ The server extracts the target user's training data from the received gradients.

tion attacks: the noise addition does not guarantee equal protection over all model gradients. As we visualize in Figure 3, some data points can be extracted with higher fidelity than others. This is despite the fact that all corresponding gradients were protected with the same clipping and noising operations. We identify the gradient norm as the reason behind disparate protection. For small-norm gradients, noise dominates the signal of the extracted data more than for gradients with a large norm. In principle, clipping in DP for ML is supposed to bound the gradient norm to control for this. However, the gradient norm is calculated globally whereas data is extracted locally from the components of the gradient that correspond to the weighted input of a single neuron. When the gradient corresponding to a neuron is large but all other neurons in that layer have small gradients, the overall gradient can still be below the clipping norm. Therefore, no clipping is performed, the same amount of noise is added to neurons with large and small gradients, and data from the neurons with larger gradients can be extracted with higher fidelity. We also sketch a construction that amplifies this effect and makes a few individual data points nearly perfectly extractable, even under noise addition.

We then discuss the centralization and resulting power imbalance between the server and users as the root cause of FL's vulnerability against attacks like the one proposed in this work. This motivates us to explore the requirements for building a variant of FL that practically prevents attacks by a malicious server. We consider three approaches based on (1) decentralization, (2) user verification, and (3) the support of specialized hardware. We find that one promising direction is to add adequate amounts of noise to users' gradients via a cryptographic protocol such as secure multiparty computation (SMCP). However as of yet, due to the gradients' high dimensionality, known constructions' communication costs are prohibitive.

As an alternative direction, users in FL can take responsibility for implementing their full privacy protection locally, for example, by adding enough noise to individually implement strong privacy guarantees. This approach is commonly referred to as local DP (LDP). As a last resort, users can decide not to participate in FL protocols if they do not trust the server. However, the last two options are only available if users have the required control over their participation in the protocol, which is not always the case.

In summary, we make the following contributions:

- We design an attack that enables a malicious server to reconstruct individual training data points from users when FL is protected by SA and DDP. See Figure 1 for an illustration of our attack flow.
- We experimentally validate the attack's ability to reconstruct image and textual data with high fidelity in different DDP setups. We show how to increase the fidelity of the reconstruction of users' data points by reducing the effect of additive noise at the level of individual neurons. We thus observe disparate leakage over gradients under DDP.
- We discuss centralization in FL and its resulting power-imbalance between the server and the users as the root cause of FL's vulnerability and analyze potential mitigations.

2. Background

This section provides background on FL, describes user-data extraction from gradients in the vanilla version of the protocol and introduces SA and DP—extensions implementing dedicated defenses against privacy leakage.

2.1. Federated Learning

FL [30] is a communication protocol that allows a group of N users to jointly train an ML model f, such that data never leaves the respective users' devices. FL involves a *server*, who coordinates the training in an iterative process, as follows: at round $t = 0$, the server initializes the shared model $\mathcal{W}^{[0]}$ at random, typically following common weight-initialization practice [19], [20], [23]. At

every round t, the server chooses a subset of $\mathsf{M} \leq \mathsf{N}$ users to contribute to the learning round. The server then sends each of these users the model $f_{\mathcal{W}^{[t]}}$. Each user chooses a subsample of their training data termed a *mini-batch*, computes the gradients of an objective function over $f_{\mathcal{W}^{[t]}}$ for each of these samples, and returns these gradients, which we call a (local) *update*, to the server. To conclude the round, the server updates the shared model by aggregating the received gradients, multiplying them by a learning-rate parameter and applying the change to the shared model.[1]

It follows from the description above that users' training data significantly affects the values of local updates. This enables a variety of privacy attacks that extract individual user-data points directly from the model updates, as highlighted in the following section.

2.2. Data Extraction from Vanilla FL

Prior work [7], [14], [51] has shown that an untrusted server can directly extract user-data from the model gradients. In these attacks, the server leverages its control over the shared model.[2] In [14], the server exploits this ability to insert a fully-connected model layer as an extraction module where individual data points can be directly extracted. [7] manipulates the model weights with an attack they call trap weights. This attack increases natural data leakage from fully-connected model layers. In [51], the server instead modifies model parameters to extract single data points by increasing their gradient contribution. The attack operates in several iterations of the protocol to collect multiple gradient updates from the same user.

The presence of such data extraction attacks highlights the vulnerability of vanilla FL to privacy-leakage. To prevent training data from leaking onto updates and straight to the hands of the server, various extensions have been proposed, as we now briefly describe.

2.3. Secure Aggregation

In SA, due to Bonawitz et al. [9], users do not send their individual updates to the server. Instead, they perform, along with the server, a multiparty computation (MPC) protocol that ensures the server only receives the average of all updates in the round. Various improvements of the original protocol were suggested, for example, to allow the server to prove the correctness of the aggregate computation [52], increase robustness against malicious users' manipulated gradients [10], or improve communication efficiency [5], [22].

2.4. Differential Privacy in FL

Nothing in the design of FL prevents information leakage: FL is designed to provide confidentiality (data does not leave user devices) rather than privacy (outputs of the computation do not leak sensitive attributes from the users' input). As discussed in Section 2.2, this leaves vanilla FL vulnerable to data reconstruction attacks. To ensure the privacy of users' sensitive training data, it is natural to consider DP approaches that work by adding noise to user updates. DP is a gold standard in privacy technology because proper application of it comes with a theoretical bound on the probability of an adversary being able to distinguish adjacent datasets, *i.e.* datasets that differ solely in one data point. This implies a bound on the probability of data point extraction. In other words, a DP approach properly applied to FL updates could, in principle, ensure that individual user data points are not revealed to whoever observes the noised updates. See Appendix A for more background on DP and its integration to ML.

One possible approach to integrate DP into FL is *centralized DP* (CDP) [4], [28], [40], where the server adds noise to the mini-batch gradients received by users. CDP assumes that the server is trusted to add noise, which is not true in the threat model of this work, see Section 3. To address this, *local DP* (LDP) [47] was proposed, where each user adds noise to its local update before sending it out for aggregation, in a way that ensures the user's own dataset is protected from extraction. Unfortunately, LDP generally results in degraded model utility due to the addition of large amounts of noise to every user's update [50]. *Distributed DP (DDP)* was proposed as a popular middle ground between CDP and LDP, where multiple users independently add noise to their update, that is sufficient to ensure their datasets are protected from an extraction adversary, but only as long as their updates are aggregated before the adversary observes them. Through combination with SA [2], [11], [27] or similar aggregation methods [6], where the server can only view aggregated updates, DDP ostensibly ensures that the server cannot extract individual data points. But, importantly, this assumes that a large fraction of users participating in the FL round are honest and add their share of the noise. We challenge the applicability of this assumption in the real world.

3. Threat Model and Assumptions

We characterize our threat model and assumptions in terms of our adversary and the considered FL setup.

3.1. Adversary

Our adversary is the server, and their goal is to infer individual users' local sensitive data points. Note that the background in Section 2.1 implies that the server can—whenever they choose to—(1) control the weights of the shared model, (2) select which of the N users participate in each round, and (3) provision new users into the pool (including *sybils* controlled by the server).[3] We assume that the total fraction of sybils out of the N users is small.

1. For readers unfamiliar with gradient optimization: such gradient-based weight updates are intended to prescribe which direction $\mathcal{W}^{[t]}$ needs to move towards, for the objective to be minimized. Typically, the objective is a value measuring the model's level of prediction error across a given mini-batch.

2. There also exist optimization-based data reconstruction attacks operating on model updates. These attacks can be conducted by a passive attacker solely observing the gradients. However, computation is expensive and the reconstructed data is not necessarily of high-fidelity. We provide a brief overview of this type of attack in Section 7.

3. Capabilities (2) and (3) have been demonstrated in the real world as Google researchers introduced 189 sybils devices into the Gboard FL system and made them participate in the protocol along with real users [40].

We, furthermore, assume the server is *occasionally malicious* (OM), meaning they behave maliciously in only a few rounds of the protocol. When this happens, they can exert the above capabilities (1-3) adversarially. Do note that the server here does not deviate from the protocol and restricts themselves to only use valid operations (1-3) in the FL protocol. An OM server ensures the attack remains stealthy, and also allows the server to train a model that has high utility over the non-malicious rounds, which is an expected product of FL.

3.2. FL Setup

Our departure point are FL protocols deployed in real-world applications, such as the one described in [8]. These protocols initially focused on data minimization only. We extend them with SA and DDP, two defenses dedicated to additionally providing privacy protection for the FL protocol. We note that we chose to study an instantiation of FL with SA and DDP because it is the combination of published techniques that holds the strongest promise in the presence of an untrusted server.[4] Our goal in studying FL with SA and DDP is to show that, even if servers (*e.g.*, companies) adopted currently available best practices from the academic community, end users might not get the promised privacy protection from FL. In addition to these challenges, we note that FL with SA and DDP is not as widely deployed as vanilla FL is. This is mainly due to some increased communication costs [5] and the computational overhead of adapted mechanisms [2], [27]. Thus, our work *conservatively* characterizes the risk of privacy leakage from many instantiations of FL.

4. Attacking FL under SA and DDP

In this section, we study FL extended by DDP and SA—considered as a strongly privacy-protective instantiation of the protocol—and show that the server can still reconstruct sensitive information about the users' training data. We also forge an intuition of what factors contribute most to the leakage. Based on our findings, in Section 8, we discuss future research and implementation towards private FL.

For successful data reconstruction under DDP and SA, the server has to make use of the following three capabilities which it naturally holds in FL:

1) *Introducing sybil devices:* The server needs to be able to introduce a fraction of manipulated devices in the FL protocol. These devices can return arbitrary gradients, chosen by the server. In particular, they can contribute zero gradients to the SA.
2) *Controlling the user sampling:* To ensure that the sybil devices are sampled for SA together with a target user, the server needs to control the user sampling.
3) *Manipulating the model weights:* For improved data reconstruction performance, the server can manipulate the shared model's weights, for example, relying on one of the methods discussed in Section 2.2.

While the first two capabilities enable the server to circumvent SA and to leave the gradients of a target user with insufficient amount of noise for privacy protection under DDP, the third capability increases the amount of individual training data that can be reconstructed and extends the attack to other model architecture types.

Attack flow. Our attack aims at reconstructing the private data of one target user per malicious round in the FL training. To do so, the attack needs to circumvent the SA (Section 4.1), then exploit the weak privacy guarantees of DDP from a user's perspective (Section 4.2), and finally reconstruct the target user's individual training data points (Section 4.3). See Figure 1 for the course of our attack.

4.1. Circumventing SA

In our attack, the server circumvents SA by sampling the target user together with maliciously controlled sybil devices for the given training round. Since for each round, M users are sampled for participation, the server needs to control at least $\mathsf{M}-1$ sybil devices. It has been shown in previous work [40] that inserting an arbitrary number of sybil devices into real-world FL deployments is practically feasible. Moreover, the assumption that the server controls all-but-one participants in a specific protocol round is not a strong one. The fraction of controlled sybil devices (thousands) in comparison to the number of total users (millions to billions) is negligibly small.

Since SA-protocols provide their guarantees under the assumption that a certain fraction of users is honest [5], [9], it follows naturally that in the presence of the sybil devices, no guarantees can be provided to the target user. This is because when the gradients are aggregated over multiple users, and all but the target user contribute arbitrary gradients, known to the server, the server can extract the target user's gradients perfectly. In the easiest case, the sybil devices contribute all zero-gradients, such that the final aggregate directly only contains the target user's gradients.

Note that we do not claim that the SA or any of the underlying cryptographic primitives are broken. We solely observe that SA relies on the assumption that the clients participating in the execution are real clients and not maliciously controlled by the server [5]. However, in FL with an untrusted server, the users cannot verify this assumption. We show through our attack that this has severe implications on their privacy guarantees.

Pasquini *et al.* [37] describe a different way to circumvent SA based on the server sending out different models to different users. While the models for non-target users produce zero-gradients, the target user's model produces non-zero gradients which can be exploited for data reconstruction. An advantage of this method is that it does not require the server to control the user sampling, or to manipulate a fraction of users. Note however that in their scenario, DDP can still be efficiently applied if every user adds some noise to their (potentially zero) gradients. As a consequence, the total amount of noise can be sufficient to protect the gradients of the target user.

4. Indeed, the key alternative to FL with SA and DDP would be FL with LDP (local DP guarantees) but this alternative is not appealing because it comes at a significant utility cost for the server.

Therefore, in our attack, we rely on the controlled sybil devices to circumvent the SA.

Note that also alternative mechanisms to implement DDP, such as shuffling [6], [13] which can be put into place instead of SA, can be circumvented by our approach of inserting sybil devices with server-controlled gradients into the protocol.

4.2. Exploiting DDP Guarantees

If DDP is in place, the gradients of the target user will be slightly noisy—even with successful circumvention of SA. However, by design of DDP, the amount of noise added by each user is typically insufficient to provide a meaningful privacy guarantee *from the user's perspective* [27]. By meaningful privacy guarantees we mean, guarantees equivalent to what one would obtain in the LDP definition. This is in fact how DDP obtains a utility gain over LDP, which would have inserted sufficient noise to obtain per-user privacy guarantees that are independent of other users: DDP assumes all users will add enough noise so that the *aggregate* is sufficiently noised whereas LDP assumes each user adds enough noise to obtain *privacy in isolation*. As a consequence, in DDP, each user can add less noise locally than required for the desired total privacy level, resulting in more utility. In contrast, the guarantee provided by LDP allows the user to not trust the server or other users.

Concretely, in an LDP version of FL, the noise added by each user depends solely on the noise scale σ and the clipping parameter c of the application. As a consequence, the local noise is sampled from a Gaussian distribution according to

$$\mathcal{N}(0, \sigma^2 c^2). \tag{1}$$

In contrast, in DDP, the amount of noise added by each individual user additionally takes the number of users who participate in the round into account [46]. Assuming that M users are sampled for participation, this results in a local addition of Gaussian noise sampled from

$$\mathcal{N}\left(0, \frac{\sigma^2}{\mathsf{M}-1} c^2\right) \text{ [46].} \tag{2}$$

In Figure 2, we present the privacy-utility trade-offs resulting from training models on the CIFAR10 [29] dataset as a function of the total noise scale σ and the resulting models' accuracy on a test set. We train the private models with a state-of-the-art framework for DP-training[5] in which all hyperparameters and model architecture are tuned for the task.

Figure 2 provides two main insights. First, unsurprisingly, given the privacy-utility trade-offs mentioned above, the model utility decreases when the total noise scale σ increases. Second, the figure shows that the more users participate in a given training round, the less noise each user needs to add locally. This results from Equation (2) which relies on the total noise being aggregated over all participating users before sharing the aggregated gradients with the server.

DDP assumes that each user is *honest* and adds the required noise to their gradients. However, if even one of the users adds less than the amount of noise it should add, the desired total privacy guarantees cannot be reached. Even worse, if, as described in Section 4, a target user in FL is sampled for participation solely with controlled sybil devices that do not provide any noise for aggregation, the local noise added by the target user represents the only protection for its gradients.

These results mean that there is a tension between (1) the guarantee claimed by the server (and other users) in DDP and (2) the guarantee that a user who does not trust this server can rely on. This will lead the server optimizing for model utility to request that users add less noise to their gradients than what is needed for individual users to protect their data from leaking to an untrusted server.

4.3. Reconstructing Data

In Section 2.2, we presented different attacks that rely on manipulations of the shared model to extract individual users' training data points. In principle, each of these attacks can be included to perform data reconstruction in our FL+SA+DDP setup. However, the attack by [51] extracts individual data points over several rounds of the FL protocol. In our setup, due to the server's OM nature, single-round attacks are preferable. These allow the adversary to stay more inconspicuous and to train a more meaningful shared model throughout the benign rounds. The attack by [14] relies on manipulations of the model architecture, which are more detectable than manipulations of model parameters, such as in [7]. Finally, our experimental evaluation highlighted a significant advantage of [7] in comparison to [14] for data reconstruction under noise. [7]'s trap weights yield redundancy in the extracted data, *i.e.* the same data point can be extracted multiple times from gradients of different weight rows. We thoroughly investigate this effect in Section 5.3. The redundancy of extracted noisy data can be exploited to average out the effect of the noise and yield higher-fidelity data reconstruction, as we will show in Section 5.4. In contrast, due to the nature of their attack, in [14], each data point is only extractable once.

5. Evaluation of the Attack against SA+DDP

In this section, we present a practical evaluation of our attack against FL protected with SA and DDP. We first present our experimental setup. Then, we evaluate direct extraction of noisy gradients under DDP. We move on to experimentally evaluate the redundancy of extracted data with [7]'s trap weight approach, and discover how this redundancy can be exploited for higher-fidelity data reconstruction under noise. We illustrate this with empirical results reconstructing image and text data.

5.1. Experimental Setup

We operate in a cross-device FL setup and perform training on the CIFAR10 [29] dataset. We evaluate extraction on different mini-batch sizes $B \in \{10, 20, 100\}$. We

5. https://github.com/ftramer/Handcrafted-DP. Note, however, that our reported accuracy and achieved privacy levels ϵ cannot directly be compared with the values reported in the repository. This is because we use different noise scales than they do and train the model for 100 epochs while they only train for 30 epochs.

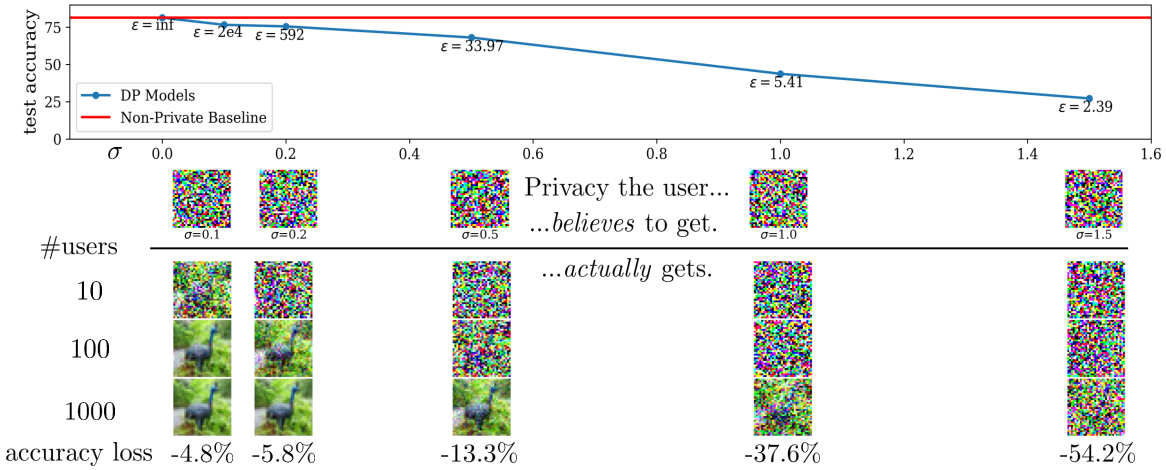


Figure 2: **Privacy vs. utility trade-offs under DDP/LDP.** Each point on the blue line corresponds to a model trained on CIFAR10 with a clipping parameter $c = 1$, and the total noise scale σ indicated on the x-axis. Training was conducted over 100 epochs, the resulting privacy guarantees ϵ are reported. For the non-private baseline $\epsilon = \infty$ (red line). We report accuracy loss with respect to this non-private baseline. The images depicted below the line plot display the rescaled noisy gradients of one data point of an individual user with noise calculated according to Equation (2) as a function of σ, c, and M in the training round. The more users participate, the less noise every user needs to add because, during aggregation, the total noise is determined by the sum of the individual noises. If, however, other users do not add noise, the locally added noise is the only privacy protection every individual user has (DDP reduces to LDP with weak privacy guarantees). As a consequence, there is a discrepancy between the privacy the user believes to get and the privacy they actually get: The images above the black line visualize the user's belief on what the server can extract from their gradient, the images below the black line visualize what the server can actually extract under our attack.

split the CIFAR10 training data at random and iid between the users. [7] shows that their trap weights' extraction success is equal for iid and non-iid distribution, even for the most extreme scenario where every data point in a given mini-batch stems from same class. Following [7], we also experiment with the IMDB dataset for sentiment analysis and distribute data the same way.

To evaluate different setups for DDP, we select $\{10, 100, 1000\}$ users for participation in a given round of the FL protocol. To circumvent the SA, as described in the previous section, we sample one target user together with sybil devices which all return zero gradients. Other than that, we follow [7]'s experimental setup, use their six-layer fully-connected neural network and embedding architecture (their Table 7 and 8), initialize the first fully-connected layer with their trap weights for individual data point extractability. To project received gradients of the first fully-connected layer's weight matrix back to their input domain, we rely on [7]'s Equation (5) which shows that it is sufficient to rescale the gradient of the weights with the inverse of the gradient of the bias for perfect data extraction. Note that in our case, due to DDP, noise is added not only to the gradient of the weights but also to the gradient of the bias. Therefore, not only the extracted gradients but also our scaling factor are noisy, resulting in the rescaled gradients not being a perfect reconstruction of the original input data.

We report the DDP-setup per-round through three parameters required to determine the noise magnitude according to Equation (2), namely the DP clip norm c, the DP noise multiplier σ, and the number of selected users in this round M. This enables us to understand how sensitive the attack is to choices for these hyperparameters without making any assumptions about other hyperparameters of the training run (e.g., the number of training steps).

Figure 3: **Directly Extracted Data under DDP.** Rescaled clipped and noised gradients from a mini-batch with 20 data points from CIFAR10 dataset. DDP setup: $c = 1$, $\sigma = 0.1$, and $\mathsf{M} = 100$.

5.2. Noisy Data Extraction

We first perform direct extraction from noisy gradients. The extraction of data points works exactly the same way as for vanilla FL [7], by initializing the model with trap weights before sending it to the target user, and then projecting their received gradients back to the input domain. Figure 3 shows the full resulting extracted data in a setup with a mini-batch of 20 data points, $c = 1$, $\sigma = 0.1$, and $\mathsf{M} = 100$. While some noisy reconstructed data points resemble the original training data, other are less recognizable because their are an overlay of multiple data points, or dominated by the added noise. DDP setup: $c = 1$, $\sigma = 0.1$.

Effect of Fraction of Sybils. We, furthermore, conducted experiments to quantify the effect of not replacing all other $\mathsf{M} - 1$ users by sybil devices, but only a fraction of the other users. Therefore, we conducted experiments with

Number of Benign Users	% Individually Reconstructable Data	Reconstruction SNRs
1	0.95	0.014
5	0.75	0.011
19	0.15	0.010
49	0.0	0.010

TABLE 1: **Influence of Fraction of Sybil Devices.** Results for FL+SSA+DDP setup with 50 participants, each holding $B = 20$ data points. We replaced a varying fraction of users by sybil devices and measured number of individually extractable data points from the target user and average SNRs over all their reconstructions. The more benign users participate in the round, the less effective data reconstruction becomes.

extracting data from one round of the FL protocol with 50 participants, each holding a mini-batch of 20 data points. Our goal was to study what privacy gain the presence of other benign users incurs on the target user. We visualize our results in Table 1. They suggest that when the target user is sampled solely with sybil devices (row 1), the server is able to extract 95% of their individual training data points individually, protected solely by the noise added locally according to DDP. The more benign users participate in the protocol round, the lower SNRs of the reconstructed data from the target user, and the fewer of the target user's individual data points can be individually extracted. This effect stems from the aggregation within the SA, which overlays gradients from all users before sharing them with the server. Our results are aligned with findings by [7] who showed that averaging over several mini-batches (which is precisely the effect of the SA) degrades extraction success.

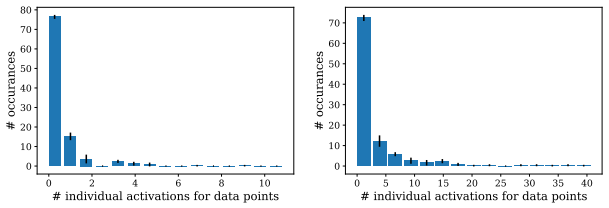
Sufficiently Protective Noise. Deciding at which point, *i.e.*, under the influence of how much noise, the reconstruction of a data point is sufficiently close to the original data point is orthogonal to this work. In particular, it will depend on the specific domain, task, and user-preference. However, users in FL should assume that the server can extract individual data points such as the ones depicted in Figure 3 from their gradients.

In the following, we will show how the server can improve the fidelity of extraction by leveraging the redundancy of extractable data due to the trap weights.

5.3. Redundancy in Extracted Data

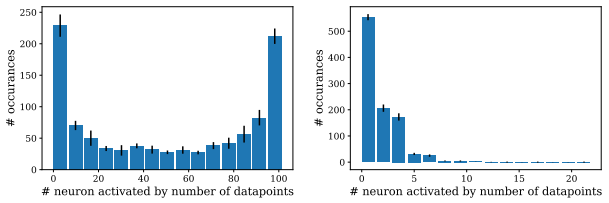
This section studies redundancy in extracted data of the trap weights method and their effect on the fidelity of reconstructed data.

Redundancy in Extractable Data Points. We first study direct redundancy by analyzing *how often* each data point in a mini-batch with $B = 100$ is individually extractable from the rescaled gradients. The results depicted in Figure 4 suggest that the trap weights, first of all, make more data points individually extractable in contrast to random model initializations, but also cause the same data points to be individually extractable from many more different weight rows' gradients (up to 70 times over the 1000 neurons and their respective weight rows).



(a) Random weights. (b) Trap weights.

Figure 4: **Number of Activations per Data Point.** The number of times each of the 100 data points is individually extractable from the model gradients. The same data points are individually extractable from many more different gradients when using trap weights which enable us to use redundancy for better data reconstruction. Results are averaged over five different random and trap weight model initializations.



(a) Random weights. (b) Trap weights.

Figure 5: **Number of Activations per Neuron.** Number of data points that activate each one of the 1000 neurons. Individual neurons are activated by fewer data points (less overlay) when using trap weights which enable better data reconstruction. Results are averaged over five different random and trap weight model initializations.

Sparsity in Extractability. We, furthermore, evaluate *by how many data points* each neuron gets activated. This is equivalent to the question how many data points cause a positive input to each neuron. Figure 5 highlights that with randomly initialized weights, neurons are activated by many more data points than with the trap weights, which causes that many data points overlay in a single gradient and we cannot extract them individually.

To improve fidelity of reconstruction, we can leverage both the redundancy of extractable data and the sparsity in the extracted data. By averaging redundant noisy data points, the signal-to-noise ratio (SNR) of reconstructed data increases as noise averages out. We visualize this effect in Figure 6. Also sparsity can be exploited. The fewer data points activate a neuron, the fewer data points contained in the overlay of the rescaled gradient. Hence, each individual data point's signal is more clearly present and identifiable in the rescaled gradients. In the following section, we will show how this can be used to yield higher-fidelity reconstruction in the image domain through clustering.

5.4. Improving Noisy Data Reconstruction

The previous sections highlight that DDP reduces to LDP with weak privacy guarantees from an individual user's perspective when other users are untrusted with their noise addition. In this section, we show how we

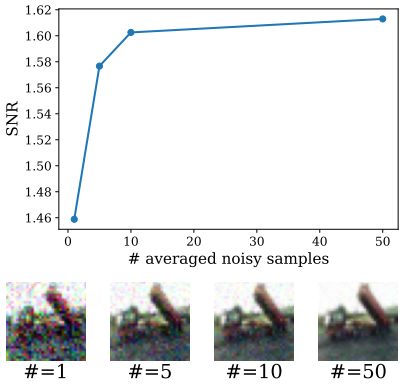


Figure 6: **Averaging out Noise.** Mean value over #-many noisy reconstructions of the same data point (bottom); corresponding mean image's SNR (top). DDP setup: $c = 1$, $\sigma = 0.1$, and $\mathsf{M} = 100$. Over an increasing number of reconstructions, the local noise averages out, yielding higher-fidelity images and increased SNR.

can even improve data reconstruction in this setup, further amplifying the small signal in the extracted gradients. We evaluate improvements for data reconstruction from noisy gradients computed under DDP on *image and textual data*. All improvements solely rely on post-processing steps to reduce the effect of the noise.

Image Data. Due to the local clipping and noise addition by the users, the data points extracted from the gradients are not perfect reconstructions of the original data points. We can still improve reconstruction quality by leveraging redundancy and sparsity in the gradients to average out the added noise, as highlighted in the previous section. However, without knowledge of the users data, the server has no means of determining which data points activate which neurons a priori. Therefore, it is unclear which rescaled gradients need to be averaged to improve reconstruction fidelity.

To overcome this limitation, we employ *similarity clustering*. In this approach, the server first filters out extracted data points with a SNR below 1. This prevents too noisy instances from degrading performance. In the following Section 6, we will discuss why different extracted data points have different SNRs. Then, the server runs a simple k-means clustering on the extracted data, and finally averages all per-cluster data points. Thereby, we do not only leverage redundancy in individually extracted data points, but also the sparsity. The signal from gradients that represent an overlay of very few data points can meaningfully contribute to the improved signal. We evaluate this approach with different noise scales and mini-batch sizes B. Note that the number k of clusters has to be chosen in accordance with the mini-batch size if we want to be able to reconstruct every data point. Our evaluation suggests that clustering works best when $k \geq 2B$.

In Figure 7, we depict the results of our clustering on data points from the CIFAR10 dataset with a DDP setup with $c = 1$, $\sigma = 0.1$ and $\mathsf{M} = 100$. The top row depicts 10 original data points, the mid and bottom rows show the closest averaged clusters for mini-batches of size 10, and 20 respectively. The more instances are available for averaging, the better the resulting per-cluster averages.

Textual Data. For the text classifier on IMDB, we initialize the weights of the embedding layer with a random uniform distribution (minimum=0.0,maximum=1.0) to create the inputs for the fully-connected layer, following [7]. We then adversarially initialize this fully-connected-layer's weights with the trap weights to perform extraction of the embeddings and invert the embeddings back to tokens using a lookup dictionary. In the presence of noise introduced for DDP, the extracted embeddings are slightly noisy. To overcome this, in presence of noise we perform the lookup by searching for the token with the closest embedding measured through the ℓ_2 distance. Figure 8 shows performance of a single mini-batch language extraction in presence of DP. Just as with image data, here an attacker is capable of extracting the original sentence of the users, despite the applied noise. We do observe however that there is stochasticity involved—when parametrization does well on the data point by default, extraction gets low performance since the received gradient has an extremely low magnitude and the corresponding signal gets dominated by the noise. We turn to this phenomena in the next section.

To summarize the results on image and textual data, we find that:

- The trap weights [7] cause input data-diversity and redundancy in resulting gradients, which can be used to cancel out some of the applied noise.
- NLP is not safe from attacks described in this paper, despite a more sophisticated input-embeddings mapping.
- Despite using DDP, an attacker often can reconstruct semantic information on the individual user data points. This is because in the presence of untrusted other users, DDP reduces to LDP with weak privacy guarantees from the perspective of an individual user.
- As shown in Figure 2, having users add more noise locally, without additional improvements of the protocol [44] comes with a significant decrease in utility which makes the solution less practical.

6. Disparate Leakage over Model Gradients

Throughout our experiments, we observe that with the exact same scale of noise added to all gradients, some extracted data points have a significantly higher SNR than others. This effect translates into different levels of semantic similarity in the extracted data with respect to the original data as we show in Figure 3. In this section, we explain this observation and sketch how it can be leveraged by the adversary to better extract data in the presence of noise.

6.1. Impact of Gradient Norm

We find that the SNR of an extracted data point is tightly bound to the magnitude, *i.e.* the norm, of the respective gradients. In Figure 9, we depict the SNR in the rescaled clipped and noised gradients, *i.e.*, the extracted data points, against their respective gradient norms. The figure shows that with higher magnitude

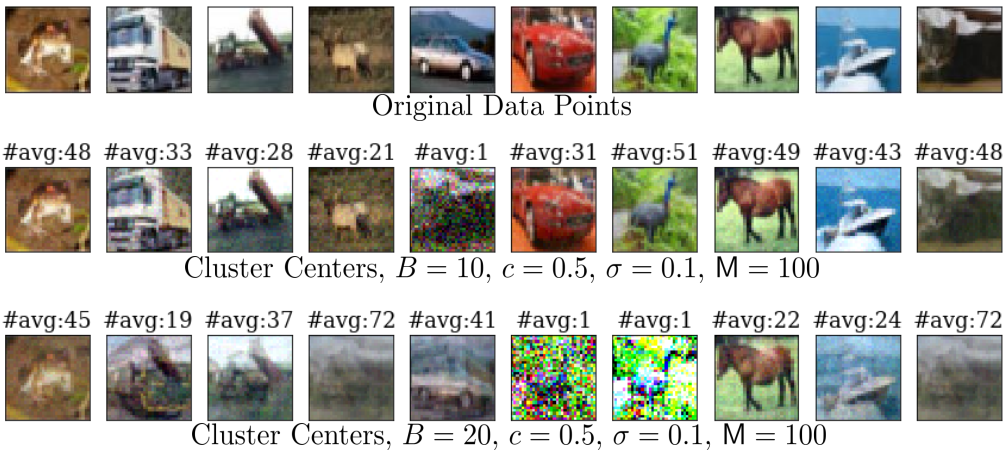


Figure 7: **Similarity Clustering** to improve noisy data extraction. Original data points and average clusters obtained from the rescaled gradients depicted in Figure 3. First 10 original training data points from the CIFAR10 dataset (top row). Averaged clusters of 10 data points reconstructed from the gradients for mini-batch size $B = 10$ (mid row), and $B = 20$ with the first 10 examples depicted (bottom row). The numbers above the images indicate how many noisy reconstructions were averaged to obtain that image.

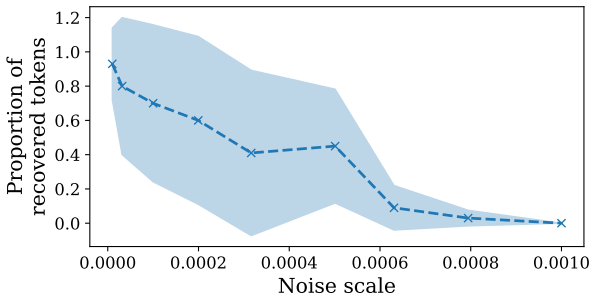


Figure 8: **Textual Data Extraction under Noise.** Extraction performance under noise for DDP from language model on the IMDB dataset. Extraction remains successful, even in presence of noise. Occasional drops in performance occur because of near-zero gradients resulted from correct data classification, *i.e.* data points with very low original loss. Error bars correspond to a single standard deviation.

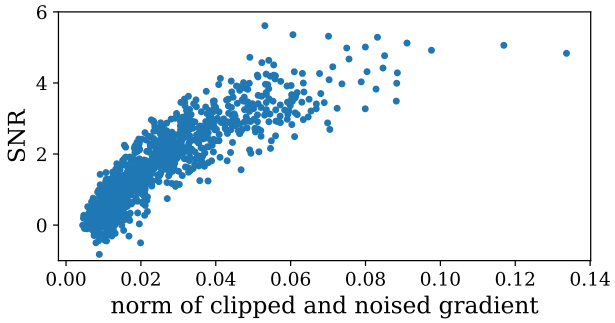


Figure 9: **Gradient Norm vs. SNR.** Norm of the clipped and noised gradients of 1000 weight rows against SNR in corresponding extracted data point, *i.e.* the rescaled gradients. With higher gradient norms, the SNR in the extracted data increases. DDP setup: $c = 1$, $\sigma = 0.1$, and $\mathsf{M} = 100$.

gradients, the same amount of noise has less impact on the signal, whereas, with smaller magnitude gradients, the same amount of noise largely dominates the signal. Therefore, increased magnitude of model gradients results in an increased data leakage.

The norm of a weight row's gradients in the model depends on the model's loss. In general, higher loss results in higher magnitude gradients, in particular for the weight rows that most contribute to the loss. Intuitively, to increase data leakage from noisy gradients, the server could, therefore, manipulate the shared model to produce higher loss. In the best case, the loss would be caused by all weight rows in the fully-connected layer used for extraction with the trap weights. This ensures high-magnitude gradients at all the weight rows' gradients and, thereby, enables enhanced extraction at all of them.

6.2. Global vs. Local Effect of Clipping

However, in DDP, before noise addition, users perform a clipping step, bounding the maximum per-layer gradient norm, and hence the extractable signal from a gradient update. More precisely, clipping bounds the total norm of a model layer's gradients to the clipping parameter c. If all weight rows have high gradients, their joint norm will exceed c, and therefore, all of them will have to be scaled down to reduce the total norm to c. The effect is visualized in the middle row of Figure 10. It shows that in this scenario, the extracted data over all gradients has a relatively low signal, which yields low-fidelity reconstruction.

Even though with DP and clipping, it is not possible to have high magnitude gradients over *all* weight rows, we note that the clipping is performed *globally* per-model layer. Hence, if only a few weight rows *locally* have a high magnitude gradient but all other weight rows have

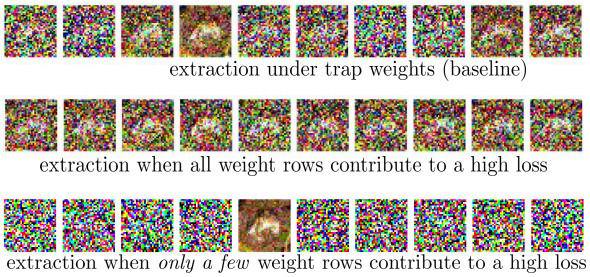


Figure 10: **Extraction Success with Additional Model Manipulations. Row 1 (top):** Under trap weights only (baseline), gradients at different weight rows have varying SNRs under the same amount of added noise, depending on their magnitudes. **Row 2 (middle):** When the shared model is further manipulated (Section 6.3) and all weight rows contribute equally to a high loss, their gradients will be clipped, which results in equal information loss for all of them. **Row 3 (bottom):** When only a few weight rows contribute to a high loss, their gradients preserve a high magnitude over clipping, which allows for higher fidelity extraction. DDP setup: $c = 1$, $\sigma = 0.1$, and $\mathsf{M} = 10$.

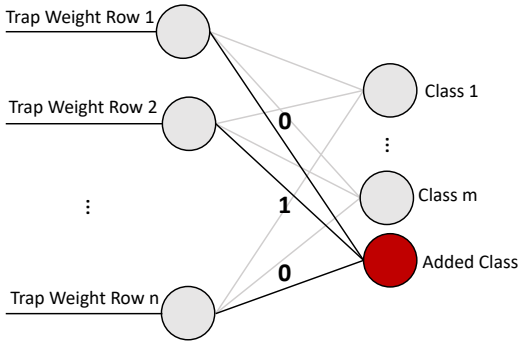


Figure 11: **Amplification of Gradient Magnitudes**. The misclassification of an example to the added class increases the magnitude of its gradient for weight rows that contribute to the loss.

a low magnitude gradient, then their joint norm can be below c. As a consequence, no clipping will be performed. The effect of this scenario is visualized in the bottom row of Figure 10. It highlights that while most gradients yield pure-noise reconstruction, a few gradients contain a very high-fidelity reconstruction of the input data. These gradients correspond to individual neurons whose gradients were less affected by the clipping operation due to the local vs. global effect we described. This higher vulnerability of certain neurons is desirable for improved data reconstruction under DDP.

6.3. Exploiting Global Clipping for Increased Extractability

The previous section highlights that the *global* per-layer clipping in DP still allows *local* parts of the gradients to be large. This can be exploited for higher-fidelity data extraction from neurons corresponding to these gradients. In this section, we sketch the idea for a possible model manipulation that gives an attacker the control on local gradient magnitudes to amplify this effect.

We base our manipulation on a fully-connected neural network with two layers. The first layer is initialized with the trap weights for extraction and has a ReLU activation function. The second classification layer is modified to yield high loss (without knowledge of the user data). Therefore, we add an additional neuron to the classification layer, *i.e.*, an additional class that does not occur in the data distribution. Then, we set most of the weights connecting the output of the previous layers' neurons to this additional class to very small values, *e.g.* zero, and the weights for a few neurons' output to high values, *e.g.* one.

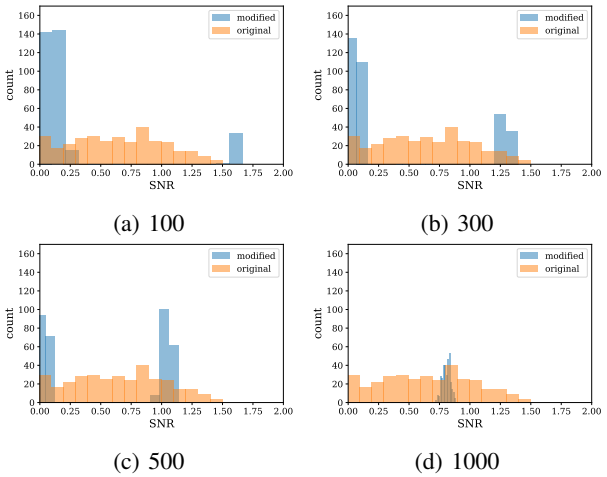
Due to the ReLU activation of the first layer, this layer's outputs are positive. High weights, connecting neurons to the added class in the second layer, "attribute" the loss of the misclassification to a few trap weight rows. As a consequence, only the gradients of these few trap weight rows obtain a high magnitude (as illustrated in Figure 11 for the *Trap Weight Row 2*). All other trap weight rows (row 1 and n in Figure 11) have low-magnitude gradients. The overall norm of the layer's gradients will stay below the clipping parameter c, hence no clipping will be applied, enabling high-fidelity data extraction from the *Trap Weight Row 2*.

We instantiated the above construction with a fully-connected neural network consisting of 1000 neurons in the first layer and eleven neurons in the second layer for experimental evaluation. The first layer's weights were initialized with trap weights, while the second layer was initialized with a Glorot uniform distribution. We then manipulated the weights connecting to the eleventh (added) class and set varying fractions (10%, 30%, 50%, and 100%) of them to one and the rest to zero. Using the CIFAR10 dataset, we performed data reconstruction.

In Figure 10, we visualize the extracted data. For the top row, all weights of the second layer are initialized at random. In the middle row, 100% of the weights connecting to the added class are set to one, and in the bottom row, 10% of these weights are set to one and the rest to zero.

We furthermore measured the SNRs of the extracted data and depict the results in Figure 12. The results are consistent to the observations from Figure 10: When few weight rows contribute to the high loss, their respective rescaled gradients, *i.e.* the extracted data points, have a higher SNR. This allows for higher-fidelity extraction. In contrast, when all weight rows contribute equally to the high loss, all their rescaled gradients have a similar (lower) SNR. This is because of all their gradients being (equally) affected by the clipping. In general, the more weight rows contribute to a high loss, the lower their individual SNRs.

Our two-layer construction naturally integrates with other architectures starting with convolutional and embedding layers described in this work. We leave fine-tuning and the extension of our construction to architectures that end with more than two fully-connected layers for future work. However, the approach highlights that even when DDP is in place, the server can initialize a model to increase the likelihood of reconstructing points with high fidelity.



(a) 100 (b) 300

(c) 500 (d) 1000

Figure 12: **SNRs of Rescaled Clipped and Noised Gradients,** *i.e.* the extracted data points. The first fully-connected layer used for extraction consists of 1000 neurons. The respective weight rows are initialized with our trap weights. The second layer is manipulated by adding an additional class neuron, and setting $\{100, 300, 500, 1000\}$ of the 1000 weights that are connected to this neuron to one. The remaining weights that go to this neuron are set to zero. The *original* baseline consists in randomly initialized weights for second fully-connected model layer. Noise with scale 0.001 is added to all gradients; clipping parameter $c = 1$. When fewer weight rows (in this case 100) contribute to the high loss, the data points extracted from the respective rescaled gradients have the highest SNR (> 1.5), which allows for higher fidelity extraction (compare to Row 3 (bottom) in Figure 10).

7. Related Work

We survey related work on privacy attacks against model gradients, in particular in the setup of FL.

Passive Attacks against Vanilla FL. Phong *et al*. [39] were the first to show how gradients leak information that can be used to recover training data at single neurons or linear layers. Recent work [16], [39], [49], [54], [56], [57] proposed exploiting this leakage for data reconstruction, for example, through Generative Adversarial Networks [21] (GANs), or by solving a second order optimization problem.

Active Attacks against Vanilla FL. Melis *et al*. [33] proposed membership [42] and property inference [3] attacks based on periodically analyzing the model updates in an FL setup. They consider both passive and active attackers in vanilla FL. Nasr *et al*. [35] assume active attackers (both users and server) for membership inference. Similarly to Melis *et al*., their attackers do not directly alter the shared model, but rather manipulate it through model updates (users through gradients, and the server by modifying the aggregated model update). Recent work [7], [14], [51], have shown that manipulating the shared model allows a malicious server to extract user data perfectly from the model gradients. In contrast to all these attacks that consider vanilla FL, our work attacks FL with additional extension for dedicated privacy protection through DDP and SA. As we discussed in Section 4.3, the extraction attacks for vanilla FL can be integrated into our attack flow for improved data extraction. We show this using [7]'s trap weights.

Attacks against Hardened FL. To our knowledge, the only prior work in this vein is due to Pasquini *et al*. [37]. This work noticed that in SA the server has the ability to dispatch inconsistent models to different users, and used this to circumvent SA entirely. As we note in Section 4.1, their attack against SA is not able to circumvent DDP since even when all users's models but the target user's model produce zero gradients, benign users would still add their share of noise. Thereby, privacy guarantees through DDP can still be achieved. Finally, while Pasquini *et al*. focuses on leveraging a specific capability of the server to circumvent a specific mechanism, we systematically study the server's capabilities arising from FL's pervasive centralization, and consequently offer a rich breadth of contributions over this work. We suggest a simpler and more powerful attack that relies on sybils introduced by the server to circumvent SA, compose this attack with other attacks relying on other capabilities to extract individual data points (the attack by [37] only extracts updates, not individual user data points) and attack a variant that also includes DDP.

8. Discussion: Privacy-Preserving FL

In this section, we discuss the following three main questions:

- *Q1: What is the core reason behind FL's vulnerability to privacy attacks as the one of this work?*
- *Q2: How can the vulnerability be fixed?*
- *Q3: What privacy guarantees can, as of now, be provided to FL users?*

8.1. Q1: What is the Root Cause of Vulnerability?

We posit that *the root cause of FL's vulnerability to attacks, like the one introduced in this work, is the* ***power imbalance*** *in the centralized design of FL*. At its core, FL is a highly centralized protocol where the server makes the final decisions. For example, DDP+SA is commonly regarded as a privacy-enhancing solution in modern FL, adding local noise and decentralizing the aggregation step of the original design, however, the server can circumvent the defense by controlling users and manipulating the shared model. In this paper, we demonstrate that even though the protocols behind DDP+SA and their fundamental cryptographic primitives are correct, the underlying assumptions are not met when the server is malicious.

We detail the factors enabling the different aspects of the attack presented in this work:

1) The server is able to control a fraction of users.
2) The server provisions users for participation in each protocol-round.
3) The server holds the power to manipulate the shared model.
4) The users have no inherent way of verifying each other's correctness.
5) The users cannot meaningfully validate model updates.

Decentralization can indeed help balance the power disparity, but it should be introduced very carefully and consider the system as a whole [38]. Next, we elaborate more on decentralization and other methods that can address the vulnerability.

8.2. Q2: How to Fix the Vulnerability?

We analyze how to fix the vulnerability by considering the following three approaches: (1) decentralization, (2) verification on the user-side to decrease trust assumptions made about the server, and (3) application of external hardware and (cryptographic) protocols that implement guarantees under the presence of an untrusted server.

Decentralizing FL. Several approaches were introduced to decentralize parts of FL, or the entire protocol. We survey them and discuss their practical applicability.

Decentralized methods for selecting a protocol round's participants are available [4], [18], [41], [53]. For example, mechanisms where the users perform a self-sampling, such as [18], have been put forward. These allow users to decide about their participation and, thereby, reduce the power of the server. Anarchic FL [53] goes even further and lets users choose an arbitrary point in time to update the shared model with fully-individualized training parameterization; the server observes user updates directly, but each user can individually determine their required privacy level and act accordingly (for example, train on a lot of data, or add a lot of noise). This does not only mitigate the attack vector where the server samples a target user along with sybil devices. If the users implement enough protection locally, the server cannot extract their private data. Yet, such mechanisms come with a significant increase in operational complexity and are faced with the major challenge of motivating users to provide truthful private data, compromising overall system utility.

An interesting decentralized-FL attempt, Biscotti [41], addresses the issue of sybil attacks by offering a blockchain based protocol that selects users in a decentralized fashion based on past behavior that appears to demonstrate honesty. This is called Proof-of-Federation (PoF), and honesty is indicated by contributing updates to the model that appear close to many other updates (and are thus assumed to be of high quality). However, users can try to appear honest and yet act maliciously in ways that will not affect the update-quality metric, for example, by not performing their role when they are supposed to noise or verify updates (roles for round participants in the Biscotti protocol which have no performance quality metric). Biscotti would have to be extensively audited for vulnerabilities to this and other attacks before it can be safely and widely deployed. Ultimately, the designs of these decentralized FL systems are very different from FL's original design, and usually from each other's designs. At the time of writing, we are not aware of any prominent real-world FL deployments that adopt such decentralization.

User-Side Verification. Alternatively to decentralization, or in addition to it, we can try to reduce the trust that users have to place in the server by performing verification on the user side.

First, we discuss the verification of the shared model. Giving users the ability to verify the shared model's integrity can prevent malicious manipulations against it, such as the ones of the trap weights that we integrated into our attack flow. Unfortunately, without any changes to FL, there is no full-proof way to distinguish weight manipulation from model weights resulting from legitimate previous training. Users have no insights on the data of other users, updates are affected by stochasticity including sampling [43], non-deterministic hardware elements [26], and large-scale FL applications with control flow mechanisms [8] that do not sample every user in every round [40]. Ultimately, this makes it impossible to track how the model evolves over time since a user gets only intermittent views of the shared model. Given these difficulties, we might consider solutions that modify FL to make manipulation-detection easier.

Second, we discuss the option of users verifying other users. Privacy guarantees in DDP result from all users adding their share of noise to protect the aggregate of all gradients. If only one user does not add their share of noise, the claimed theoretical overall privacy guarantees cannot be reached. Our attack exploits this effect and exposes the gradients of a target user by omitting noise addition of all other users sampled in the protocol round.

To prevent this attack vector, users have to verify that other users calculate their gradients correctly and add the correct amount of noise. However, due to the centralization in standard FL protocols, users do not have direct communication channels with each other. The central party acts as an intermediary in their interactions. As an alternative, FL could be deployed within a Public Key Infrastructure (PKI) [5]. However, a server that behaves semi-honestly is required during the key collection phase of PKI. Hence, users rely again on their trust in the server while nothing in the protocol prevents this server from acting maliciously and registering their introduced sybil devices to the PKI prior to training.

In fact, protocols like SA rely on the assumption that users participating in the protocol are actual users and no sybil devices controlled by the server [5]. Yet, nothing in the integration of the protocol into FL verifies or enforces this assumption. As a consequence, users who do not trust the server will have to verify that other users participating with them in a protocol round follow the protocol, correctly calculate their gradients, and add their share of noise. Verifying correct gradient calculations without users having to reveal their private data to each other, can, in principle, be implemented through zero knowledge proofs (ZKPs) where users commit to their private data and prove correct gradient calculation. However, in our attack, the sybil devices are controlled by the server. As a consequence, even when they compute correct gradients, the server will be able to subtract these from the aggregate gradient to obtain the target user's gradients and conduct extraction as described in this work. The same argument can be applied to correct noise addition.

This motivates the need for users to verify that other users are no sybil devices. Prior sybil device detection in FL relies on analyzing gradients [15] under the assumption of a non-malicious server. This approach fails to prevent our attack where sybils are controlled by the server and can, as mentioned above contribute meaningful gradients,

as long as the server can subtract these from the aggregate. Yet, *if* users in FL have a way to confidently determine if other users are sybil devices, and *if* users have a chance to refrain from contributing to the protocol under their presence, our attack can be mitigated.

Finally, it is desirable to enable users to verify the whole FL application and its local execution. This allows them to make sure that their local client handles their data correctly, adds enough local noise, and is not manipulated by the server. However, in modern FL ecosystems the FL client applications are proprietary software encapsulated in dedicated partitions [48] largely inaccessible to users. Additionally, the developers of verification software are usually the same entity acting as the server, bestowing it with even more power. Therefore, we recommend that *applications of purportedly privacy-preserving protocols should be open-source* so that they are available for audits and verification by the community.

Hardware and Protocol Support. As the last approach to fixing the vulnerability of FL, we discuss the support of dedicated hardware and (cryptographic) protocols to implement guarantees for the users.

By relying on protocols that are based on trusted execution environments (TEE), *e.g.* [34] the server can be prevented from manipulating the shared model. This reduces the success of data extraction as the one underlying our attack. Additionally, to make sure that users always receive a well-controlled and no a manipulated model, we suggest releasing the shared model publicly, for example, in a block-chain. This makes it impossible to manipulate and change a shared model after release.

A drawback of this solution is that it offers outside attackers access to several intermediate model states. We argue that this is not too restricting, though, since the shared model is also sent out to a few hundreds or thousands of users during each round. Hence, there exists the possibility of the internal model states being leaked, anyways. Yet, [7] has shown that even non-manipulated ML models, under certain conditions, leak their private training data. Moreover, when it comes to TEEs, they are prone to side-channel attacks, *e.g.* [25]. Hence, we conclude that even under such hardware protection some privacy risk for users remains.

Homomorphic encryption (HE) is a candidate solution to protect user gradients against leakage to a malicious server. However, existing instantiations [55] do not prevent our attack since, for efficiency in training, users decrypt the aggregated gradients and apply them to their (unencrypted) local model. In our attack, the server controls sybil devices, hence, it obtains access to the decrypted aggregate and can extract the target user's gradient by subtracting the sybil devices' contributions.

Finally, a cryptographic protocol that always adds enough noise to user gradients in an aggregation step would mitigate the privacy leakage of the attack presented in this work. The protocol should offer DP, but without making any questionable trust assumptions on other users. We envision a secure multiparty computation (SMPC) protocol that performs update aggregation, much like SA, but also ensures that the output is added with a sufficient amount of noise to implement DP before it is dispatched to the server. As long as within this protocol users do not learn about each others' inputs, and the server only learns the aggregated (and noised) output, the protocol may be able to offer a comparatively favorable privacy-utility trade-off. To the best of our knowledge, so far, no protocol for jointly adding sufficient amounts of noise to user gradients in SMPC exists and given the gradients' high dimensionality, the costs of any such approach will most likely not be practical, yet.

8.3. Q3: What Users Can Do Today?

Let us assume that a user wishes to attain DP guarantees, and with good reason, as DP is the most viable and protective form of privacy guarantee in use today. Our study of FL and its extensions demonstrates that the mere inclusion of a DP mechanism does not necessarily provide protection if this mechanism makes assumptions that are inaccurate in the context of the given system.

We currently see two promising directions for users to attain strong privacy guarantees.

Local Differential Privacy. The first one consists of implementing their full privacy protection locally, without trusting any other participant of the protocol, neither the server nor other users to contribute to their protection. One way to implement such protection is LDP with conservative parameterization, and while ensuring that data point reuse is accounted for and prevented when needed [45]. While LDP comes at cost of the utility of the shared models, approaches to successfully improve the trade-offs exist for FL deployments with large numbers of participants [44]. We think that improving LDP to desirable privacy-utility trade-offs is a promising direction.

FL Protocols with Trusted Servers. The second option for users to obtain privacy guarantees is to opt-out of FL altogether, if they do not trust the server, potentially while trying to hide this, for example, by providing randomized "garbage" updates to the server. However, this approach comes with a corresponding utility cost and undermines the purpose of collaborative learning to produce a performant model that fits many diverse individuals.

Finally, these options are only available if users can control the local FL application software, which is usually not the case (unless an opt-out option is provided).

9. Conclusion

Truly privacy-preserving ML must defend itself from attackers that are malicious and hence do not follow protocol. In this work, we presented a highly efficient data extraction attack against FL in a strongly protected deployment, namely with SA and DDP. Based on the attack's success, we analyzed the question of the minimum trust model that is required to obtain meaningful privacy guarantees for the users. We showed that FL can provide privacy guarantees if the users trust the server, or if they rely on adequate cryptographic protocols, and if adequate additional protection methods are in place. Most of the methods for protection aim at shifting power from the server to the conglomerate of users and come with significant costs or overhead. Therefore, such systems are not yet practically in place. As a consequence, as of yet, we recommend that users only participate in FL protocols that are orchestrated by a trusted server.

References

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In Edgar Weippl, editor, *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016.

[2] Naman Agarwal, Peter Kairouz, and Ziyu Liu. The skellam mechanism for differentially private federated learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[3] Giuseppe Ateniese, Luigi V Mancini, Angelo Spognardi, Antonio Villani, Domenico Vitali, and Giovanni Felici. Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers. *International Journal of Security and Networks*, 10(3):137–150, 2015.

[4] Borja Balle, Peter Kairouz, Brendan McMahan, Om Thakkar, and Abhradeep Guha Thakurta. Privacy amplification via random check-ins. *Advances in Neural Information Processing Systems*, 33:4623–4634, 2020.

[5] James Henry Bell, Kallista A Bonawitz, Adrià Gascón, Tancrède Lepoint, and Mariana Raykova. Secure single-server aggregation with (poly) logarithmic overhead. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1253–1269, 2020.

[6] Andrea Bittau, Úlfar Erlingsson, Petros Maniatis, Ilya Mironov, Ananth Raghunathan, David Lie, Mitch Rudominer, Ushasree Kode, Julien Tinnes, and Bernhard Seefeld. Prochlo: Strong privacy for analytics in the crowd. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 441–459, 2017.

[7] Franziska Boenisch, Adam Dziedzic, Roei Schuster, Ali Shahin Shamsabadi, Ilia Shumailov, and Nicolas Papernot. When the curious abandon honesty: Federated learning is not private. *arXiv preprint arXiv:2112.02918*, 2021.

[8] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, Brendan McMahan, et al. Towards federated learning at scale: System design. *Proceedings of Machine Learning and Systems*, 1:374–388, 2019.

[9] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1175–1191, 2017.

[10] Lukas Burkhalter, Hidde Lycklama, Alexander Viand, Nicolas Küchler, and Anwar Hithnawi. Rofl: Attestable robustness for secure federated learning. *arXiv preprint arXiv:2107.03311*, 2021.

[11] Wei-Ning Chen, Christopher A Choquette-Choo, Peter Kairouz, and Ananda Theertha Suresh. The fundamental price of secure aggregation in differentially private federated learning. *arXiv preprint arXiv:2203.03761*, 2022.

[12] C Dwork. Differential privacy in the 40th international colloquium on automata. *Languages and Programming*, 2006.

[13] Úlfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Abhradeep Thakurta. Amplification by shuffling: From local to central differential privacy via anonymity. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2468–2479. SIAM, 2019.

[14] Liam H Fowl, Jonas Geiping, Wojciech Czaja, Micah Goldblum, and Tom Goldstein. Robbing the fed: Directly obtaining private data in federated learning with modified models. In *International Conference on Learning Representations*, 2021.

[15] Clement Fung, Chris JM Yoon, and Ivan Beschastnikh. The limitations of federated learning in sybil settings. In *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)*, pages 301–316, 2020.

[16] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. *Inverting Gradients – How easy is it to break privacy in federated learning?* 2020. 23 pages, 20 figures. The first three authors contributed equally.

[17] Antonious Girgis, Deepesh Data, Suhas Diggavi, Peter Kairouz, and Ananda Theertha Suresh. Shuffled model of differential privacy in federated learning. In *International Conference on Artificial Intelligence and Statistics*, pages 2521–2529. PMLR, 2021.

[18] Antonious M Girgis, Deepesh Data, and Suhas Diggavi. Differentially private federated learning with shuffling and client self-sampling. In *2021 IEEE International Symposium on Information Theory (ISIT)*, pages 338–343. IEEE, 2021.

[19] Raja Giryes, Guillermo Sapiro, and Alex M Bronstein. Deep neural networks with random gaussian weights: A universal classification strategy? *IEEE Transactions on Signal Processing*, 64(13):3444–3457, 2016.

[20] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323. JMLR Workshop and Conference Proceedings, 2011.

[21] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, Montreal, Canada, December 2014.

[22] Xiaojie Guo, Zheli Liu, Jin Li, Jiqiang Gao, Boyu Hou, Changyu Dong, and Thar Baker. Verifl: Communication-efficient and fast verifiable aggregation for federated learning. *IEEE Transactions on Information Forensics and Security*, 16:1736–1751, 2020.

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2015.

[24] Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. Deep models under the gan: information leakage from collaborative deep learning. 2017.

[25] Patrick Jauernig, Ahmad-Reza Sadeghi, and Emmanuel Stapf. Trusted execution environments: properties, applications, and challenges. *IEEE Security & Privacy*, 18(2):56–60, 2020.

[26] Hengrui Jia, Mohammad Yaghini, Christopher A Choquette-Choo, Natalie Dullerud, Anvith Thudi, Varun Chandrasekaran, and Nicolas Papernot. Proof-of-learning: Definitions and practice. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 1039–1056. IEEE, 2021.

[27] Peter Kairouz, Ziyu Liu, and Thomas Steinke. The distributed discrete gaussian mechanism for federated learning with secure aggregation. In *International Conference on Machine Learning*, pages 5201–5212. PMLR, 2021.

[28] Peter Kairouz, Brendan McMahan, Shuang Song, Om Thakkar, Abhradeep Thakurta, and Zheng Xu. Practical and private (deep) learning without sampling or shuffling. In *International Conference on Machine Learning*, pages 5213–5225. PMLR, 2021.

[29] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[30] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. pages 1273–1282, 2017.

[31] Brendan McMahan and Abhradeep Thakurta. Federated learning with formal differential privacy guarantees. https://ai.googleblog.com/2022/02/federated-learning-with-formal.html?m=1, 2022. Last Acccessed: March 25, 2022.

[32] H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models. In *International Conference on Learning Representations*, 2018.

[33] Luca Melis, Congzheng Song, Emiliano de Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 691–706. IEEE, 19/05/2019 - 23/05/2019.

[34] Fan Mo, Hamed Haddadi, Kleomenis Katevas, Eduard Marin, Diego Perino, and Nicolas Kourtellis. Ppfl: privacy-preserving federated learning with trusted execution environments. In *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services*, pages 94–108, 2021.

[35] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE symposium on security and privacy (SP)*, pages 739–753. IEEE, 2019.

[36] Nicolas Papernot, Martín Abadi, Úlfar Erlingsson, Ian Goodfellow, and Kunal Talwar. Semi-supervised knowledge transfer for deep learning from private training data, 18/10/2016. Accepted to ICLR 17 as an oral.

[37] Dario Pasquini, Danilo Francati, and Giuseppe Ateniese. Eluding secure aggregation in federated learning via model inconsistency. *arXiv preprint arXiv:2111.07380*, 2021.

[38] Dario Pasquini, Mathilde Raynal, and Carmela Troncoso. On the privacy of decentralized machine learning. *arXiv preprint arXiv:2205.08443*, 2022.

[39] Le Trieu Phong, Yoshinori Aono, Takuya Hayashi, Lihua Wang, and Shiho Moriai. Privacy-preserving deep learning: Revisited and enhanced. In *International Conference on Applications and Techniques in Information Security*, pages 100–110. Springer, 2017.

[40] Swaroop Ramaswamy, Om Thakkar, Rajiv Mathews, Galen Andrew, H Brendan McMahan, and Françoise Beaufays. Training production language models without memorizing user data. *arXiv preprint arXiv:2009.10031*, 2020.

[41] Muhammad Shayan, Clement Fung, Chris J. M. Yoon, and Ivan Beschastnikh. Biscotti: A blockchain system for private and secure federated learning. *IEEE Transactions on Parallel and Distributed Systems*, 32(7):1513–1525, 2021.

[42] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In IEEE Symposium on Security and Privacy, editor, *2017 IEEE Symposium on Security and Privacy - SP 2017*. IEEE Symposium on Security and Privacy and SP and Symposium on Security & Privacy and S & P and IEEE S & P, IEEE, 2017.

[43] Ilia Shumailov, Zakhar Shumaylov, Dmitry Kazhdan, Yiren Zhao, Nicolas Papernot, Murat A. Erdogdu, and Ross Anderson. *Manipulating SGD with Data Ordering Attacks*. 2021.

[44] Lichao Sun, Jianwei Qian, and Xun Chen. Ldp-fl: Practical private aggregation in federated learning with local differential privacy. *arXiv preprint arXiv:2007.15789*, 2020.

[45] Anvith Thudi, Ilia Shumailov, Franziska Boenisch, and Nicolas Papernot. Bounding membership inference, 2022.

[46] Stacey Truex, Nathalie Baracaldo, Ali Anwar, Thomas Steinke, Heiko Ludwig, Rui Zhang, and Yi Zhou. A hybrid approach to privacy-preserving federated learning. In *Proceedings of the 12th ACM workshop on artificial intelligence and security*, pages 1–11, 2019.

[47] Stacey Truex, Ling Liu, Ka-Ho Chow, Mehmet Emre Gursoy, and Wenqi Wei. Ldp-fed: Federated learning with local differential privacy. In *Proceedings of the Third ACM International Workshop on Edge Systems, Analytics and Networking*, pages 61–66, 2020.

[48] Jack Wallen. Android 12 adds ai and machine learning with private compute core but keeps your data secure. https://www.techrepublic.com/article/android-12-adds-ai-and-machine-learning-with-private-compute\protect\discretionary{\char\hyphenchar\font}{}{}core-but-keeps-your-data-secure/, 2021. Last Acccessed: October 17, 2022.

[49] Zhibo Wang, Mengkai Song, Zhifei Zhang, Yang Song, Qian Wang, and Hairong Qi. Beyond inferring class representatives: User-level privacy leakage from federated learning. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*. IEEE, 2019.

[50] Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H Yang, Farhad Farokhi, Shi Jin, Tony QS Quek, and H Vincent Poor. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security*, 15:3454–3469, 2020.

[51] Yuxin Wen, Jonas A. Geiping, Liam Fowl, Micah Goldblum, and Tom Goldstein. Fishing for user data in large-batch federated learning via gradient magnification. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 23668–23684. PMLR, 17–23 Jul 2022.

[52] Guowen Xu, Hongwei Li, Sen Liu, Kan Yang, and Xiaodong Lin. Verifynet: Secure and verifiable federated learning. *IEEE Transactions on Information Forensics and Security*, 15:911–926, 2019.

[53] Haibo Yang, Xin Zhang, Prashant Khanduri, and Jia Liu. Anarchic federated learning. *arXiv preprint arXiv:2108.09875*, 2021.

[54] Hongxu Yin, Arun Mallya, Arash Vahdat, Jose M Alvarez, Jan Kautz, and Pavlo Molchanov. See through gradients: Image batch recovery via gradinversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16337–16346, 2021.

[55] Chengliang Zhang, Suyi Li, Junzhe Xia, Wei Wang, Feng Yan, and Yang Liu. {BatchCrypt}: Efficient homomorphic encryption for {Cross-Silo} federated learning. In *2020 USENIX annual technical conference (USENIX ATC 20)*, pages 493–506, 2020.

[56] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. idlg: Improved deep leakage from gradients. *arXiv preprint arXiv:2001.02610*, 2020.

[57] Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. *Advances in neural information processing systems*, 32, 2019.

A. Differential Privacy and DPSGD

DP [12] formalizes the idea that no data point should significantly influence the results of an analysis conducted on a whole dataset. Therefore, the analysis should yield roughly the same results whether or not any particular dataset is included in the dataset or not. Formally, this is expressed by the following definition.

Definition 1 ((ε, δ)-Differential Privacy)**.** Let $A\colon \mathcal{D}^* \to \mathcal{R}$ a randomized algorithm. A satisfies (ε, δ)-DP with $\varepsilon \in \mathbb{R}_+$ and $\delta \in [0, 1]$ if for all neighboring datasets $D \sim D'$, i.e. datasets that differ on only one element, and for all possible subsets $R \subseteq \mathcal{R}$

$$\mathbb{P}\left[A(D) \in R\right] \leq e^{\varepsilon} \cdot \mathbb{P}\left[M(D') \in R\right] + \delta\,. \tag{3}$$

In ML, DP formalizes the idea that any possible training data point in a training dataset cannot significantly impact the resulting ML model [1], [36]. One notable approach to achieve this is Differentially Private Stochastic Gradient Descent (DPSGD) [1]. DPSGD alters the training process to introduce DP guarantees for weight update operations, and thereby protects underlying individual data points. Particularly, the gradient computed for each data point or a mini-batch of data points is first clipped in their ℓ_2-norm to bound influence. Clipping of the gradient $g(\{x_i\}_b)$ for mini-batch b of data $\{x_i\}_b$ is performed according to a clipping parameter c, by replacing $g(\{x_i\}_b)$ with $g(\{x_i\}_b)/\max(1, \frac{||g(\{x_i\}_b)||_2}{c})$. This ensures that if the ℓ_2-norm of the gradients is $\leq c$, the gradients stays unaltered, and if the norm is $> c$, the gradient get scaled down to be in norm of c. After the clipping, Gaussian noise with scale σ is applied to the gradients of each mini-batch before performing the model updates. DP is commonly used to make FL private – as described in **??**.