Client-specific Property Inference against Secure Aggregation in Federated Learning

Raouf Kerkouche
raouf.kerkouche@cispa.de
CISPA Helmholtz Center for
Information Security
Germany

Gergely Ács
acs@crysys.hu
Department of Networked Systems
and Services, CrySyS Lab, BME
Hungary

Mario Fritz
fritz@cispa.de
CISPA Helmholtz Center for
Information Security
Germany

ABSTRACT

Federated learning has become a widely used paradigm for collaboratively training a common model among different participants with the help of a central server that coordinates the training. Although only the model parameters or other model updates are exchanged during the federated training instead of the participant's data, many attacks have shown that it is still possible to infer sensitive information or to reconstruct participant data. Although differential privacy is considered an effective solution to protect against privacy attacks, it is also criticized for its negative effect on utility. Another possible defense is to use secure aggregation, which allows the server to only access the aggregated update instead of each individual one, and it is often more appealing because it does not degrade the model quality. However, combining only the aggregated updates, which are generated by a different composition of clients in every round, may still allow the inference of some client-specific information.

In this paper, we show that simple linear models can effectively capture client-specific properties only from the aggregated model updates due to the linearity of aggregation. We formulate an optimization problem across different rounds in order to infer a tested property of every client from the output of the linear models, for example, whether they have a specific sample in their training data (membership inference) or whether they misbehave and attempt to degrade the performance of the common model by poisoning attacks. Our reconstruction technique is completely passive and undetectable. We demonstrate the efficacy of our approach on several scenarios, showing that secure aggregation provides very limited privacy guarantees in practice. The source code is available at https://github.com/raouf-kerkouche/PROLIN.

CCS CONCEPTS

• **Security and privacy**; • **Computing methodologies** → **Machine learning**; **Distributed artificial intelligence**;

KEYWORDS

Federated learning, Secure aggregation, Client-specific property inference, Membership inference, Poisoning attacks

1 INTRODUCTION

Machine learning models have made their way into a broad range of application domains. However, accuracy of such models is typically dependent on the amount of available data. Often a lack of sufficient data prevents training of accurate models. One of the simplest solutions is collaboration between data holders by sharing data in order to train better models. Yet, this solution may not be viable if the data in question is sensitive and privacy is crucial. Federated Learning addresses the above constraints by allowing collaborative training of a model without sharing any data. Instead, only the model parameters are shared between a central server and the different entities that participate in the learning process. Federated learning has become a veritable paradigm and is used to train shared models for many applications, such as input text prediction[29], ad selection [55], drug discovery[60] or various medical applications [17, 19, 33] that use the confidential data of many different entities.

Unfortunately, even though private training data is not shared directly in federated learning, many attacks have shown that it is possible to infer sensitive information about the training data of each client. Membership attacks [45, 47] allow, for example, to infer whether a specific record is included in a participant's dataset. Similarly, the attack in [45] allows inferring whether a group of people with a specific property independent of the main task is included in any participant's dataset. Even worse, it is possible to reconstruct the training data [22, 25, 38, 39, 45, 61, 75, 76].

Solutions exist to remedy the above attacks, such as Differential Privacy. Although this can provide a strong privacy guarantee, it can also jeopardize the benefits of federated learning by severely deteriorating the accuracy of the commonly trained model. Hence, many companies are still reluctant to use Differential Privacy, especially in scenarios, where only a limited number of companies engage in training (process) and want to prevent the leakage of any, not only sample-specific information about their abundant training data[1]. However, the small number of clients is usually insufficient to counterbalance the negative effect of noise on model accuracy, which can eventually incur a (business) risk for the clients.

---

[1]https://www.melloddy.eu

Secure aggregation [12] is often used as an alternative (or complementary) mitigation technique against unintended information leakage. This cryptographic solution allows the protocol participants to access only the aggregated model updates but not the individual update of any client sent for aggregation. Indeed, most existing inference and reconstruction attacks rely on accessing the individual gradients (model updates) in order to succeed. Secure aggregation guarantees that even if any participant learns some confidential information from the aggregated model, they are still unlikely to attribute this information to any specific client without the necessary background knowledge [9]. Albeit providing strictly weaker confidentiality guarantees than differential privacy, secure aggregation does not degrade model accuracy, has small computational overhead, and has therefore become an indispensable part of any federated learning protocol. Although there exist active attacks [8, 10, 20, 50, 64] even against secure aggregation, which enforce information leakage by model or data poisoning, these attacks are either detectable, thereby providing evidence of the misdeed, or can be prevented [21, 26–28, 31, 41, 46, 70, 73]. This makes such an active attack less likely in practice, especially if clients can suffer a reputation loss due to the potential repercussions that can easily outbalance the benefit of a successful attack.

In this paper, we show that secure aggregation often fails to prevent the attribution of confidential information to a client, even if the adversary is only a *passive* observer who faithfully follows the federated learning protocol. Our attribution technique shows that the server or a client who can access only the common model in each training round can learn accurate client-specific information (i.e., a property of the client, such as whether its training data includes some specific samples) without being detected, even if secure aggregation is employed. Our technique does not need any background knowledge about any specific client to succeed, just the aggregated common model observed per round, and the identity of the clients participating per round.

We exploit the fact that the composition of participating clients changes in almost every round to decrease communication costs and guarantee convergence. This optimization allows us to solve a system of linear equations, where the unknowns are some (private) contributions of the clients whose sums are observable. Prior work [36] has shown that if these contributions are the gradients, then simple linear regression can be used to reconstruct the mean gradient vector of every client as long as the variance of the gradient is small per client and there are a sufficient number of rounds (equations). We show that, instead of disaggregating the sum of gradients and then launching a supervised inference attack on the reconstructed individual gradients per client, it is more effective to directly reconstruct the linear features used by this inference attack. In particular, we substantially improve on [36] by leveraging the linearity of model aggregation: the unknowns are the linear features of an individual model update that effectively capture property information and are reconstructed from the observed model aggregates. The (private) property value of every client is computed by maximizing the likelihood of these reconstructed features over the rounds given their prior distributions on the auxiliary dataset. Since only a small number of features are reconstructed and used for inference instead of the potentially large gradient vectors, our approach has a significantly smaller variance compared to [36] at

the cost of a slight bias. Our approach is general and can be utilized to infer various client-specific properties only from the observable aggregations of model updates. Moreover, it is completely passive, unintrusive, and does not intervene in the normal operation of federated learning.

Our main contributions are the following:

- We show that secure aggregation is not sufficient to prevent the reconstruction of client-specific information. We propose a general, completely passive reconstruction technique called PROLIN, which, exploiting the linearity of model aggregation, uses linear models to capture property information from aggregated model updates and attributes them to specific clients. We demonstrate our general approach on two detection tasks.
- We identify clients whose training data includes a specific target sample. We disaggregate the linear features used by the membership inference attack that yields increased attack accuracy compared to related work [36]. This negative result shows that accurate private information leakage is still possible with secure aggregation, even without client-specific background knowledge, and that membership inference attacks remain a significant risk, even with a passive adversary.
- We detect clients that exhibit malicious behavior by launching (untargeted) poisoning attacks. To the best of our knowledge, prior works have only addressed poisoning detection *without secure aggregation* [13, 24, 48, 52]. This positive result shows that secure aggregation is not enough to hide poisoning attacks, which decreases the incentive and therefore the risk of such attacks.

The operation of PROLIN is illustrated in Figure 1.

Figure 1: Illustration of PROLIN**.**

2 BACKGROUND

2.1 Federated Learning

In federated learning [43, 57], multiple clients build a common machine learning model from the union of their training data without

sharing the data with each other. At each round of the training, in order to reduce communication costs, only a fraction C of all N clients are randomly selected to retrieve the common model from the parameter server, update the global model based on their own training data, and send back their updated model to the server. The server aggregates the updated models of all clients to obtain a global model that is re-distributed to some selected parties in the next round. Different aggregation techniques [16, 43, 54, 72] have been proposed, we consider federated averaging (FedAvg) [43] in this paper.

More specifically, let $\mathbf{A} \in \{0, 1\}^{n \times N}$ denote the participation matrix, where $\mathbf{A}_{r,i} = 1$ if client i participates in round r, and 0 otherwise. As the total number of participants is the same in each round, it holds that $\sum_i \mathbf{A}_{r,i} = C \cdot N$ for all r. At round r, a participating client i (i.e., $\mathbf{A}_{r,i} = 1$) executes T_{gd} local gradient descent iterations on the common model T_{r-1} with parameters $\mathbf{w}_{r-1}$, using its own training data D_i, and sends the model update $\Delta \mathbf{w}_r^i = \mathbf{w}_r^i - \mathbf{w}_{r-1}^i$ to the server, which then obtains the new common model T_r by aggregating the received updates as $\mathbf{w}_r = \mathbf{w}_{r-1} + \sum_i \frac{|D_i|}{\sum_j |D_j|} \mathbf{A}_{r,i} \Delta \mathbf{w}_r^i$ (a client's update is weighted with the size of its training data), where z denotes the model (update) size [43]. Finally, the server re-distributes $\mathbf{w}_r$ to the clients selected in the next round. The server stops training after a fixed number of rounds n or when the performance of the common model does not improve on held-out data.

Federated learning is often combined with secure aggregation to prevent the server and any client from accessing the individual updates $\Delta \mathbf{w}_r^i$ rather than just their aggregation $\sum_i \frac{|D_i|}{\sum_j |D_j|} \mathbf{A}_{r,i} \Delta \mathbf{w}_r^i$ per round [1, 11]. When secure aggregation is used, each client encrypts its individual update before sending it to the server. Upon reception, the server sums the encrypted updates as:

$$\sum_{i \in \{j: \mathbf{A}_{r,j}=1\}} \text{Enc}_{\mathbf{K}_i} \left(\frac{|D_i|}{\sum_j |D_j|} \Delta \mathbf{w}_r^i \right) = \sum_i \frac{|D_i|}{\sum_j |D_j|} \mathbf{A}_{r,i} \Delta \mathbf{w}_r^i \quad (1)$$

where $\text{Enc}_{\mathbf{K}_i}(\mathbf{x}) = \mathbf{x} + \mathbf{K}_i \mod p$ and $\sum_{i \in \{j: \mathbf{A}_{r,j}=1\}} \mathbf{K}_i = 0$ (see [1, 11] for details). Here the modulo is taken element-wise and $p = 2^{\lceil \log_2(\max_k ||\Delta \mathbf{w}_r^k||_\infty |\mathbb{K}|) \rceil}$.

2.2 Linear regression

Given a linear model as

$$\mathbf{b} = \mathbf{A}\hat{\mathbf{x}} + \boldsymbol{\delta} \quad (2)$$

where $\mathbf{A} \in \mathbb{R}^{n \times N}$ is a known matrix, $\mathbf{b} \in \mathbb{R}^n$ are the observed (noisy) aggregates, and $\boldsymbol{\delta} \in \mathbb{R}^N$ are random variables describing the noise with zero mean and finite variance. In machine learning parlance, each row of $\mathbf{A}$ corresponds to a training sample with N input variables (features), and $\hat{\mathbf{x}}$ is the unobserved parameter vector of the linear model to be determined. Eq. (2) defines a system of n linear equations for $\hat{\mathbf{x}}$ as unknowns, and the method of ordinary least squares (OLS) provides an unbiased estimate $\widetilde{\mathbf{x}}$ of $\hat{\mathbf{x}}$ as

$$\widetilde{\mathbf{x}} = \underset{\mathbf{x}}{\text{argmin}}(\mathbf{b} - \mathbf{A}\mathbf{x})^2 \quad (3)$$

that is, $E[\widetilde{\mathbf{x}}] = \hat{\mathbf{x}}$ regardless of $\mathbf{A}$. Eq. (3) has the closed-form solution of $\widetilde{\mathbf{x}} = \mathbf{A}^+ \mathbf{b}$, where $\mathbf{A}^+$ is the Moore-Penrose inverse of $\mathbf{A}$. According to the Gauss-Markov theorem, $\widetilde{\mathbf{x}}$ has the smallest variance among

all unbiased estimators if $\boldsymbol{\delta}_i$ are uncorrelated, have zero mean, and equal variance. Although $\widetilde{\mathbf{x}}$ is an unbiased estimate, there are other estimators that exploit the bias-variance trade-off and decrease the variance of the estimate at the cost of introducing some bias by regularization, so that the total error (the sum of squared bias and variance) is still smaller than for any unbiased estimator, including OLS.

Ridge Regression (RR) provides an L_2 regularized estimation of $\hat{\mathbf{x}}$ as

$$\widetilde{\mathbf{x}}' = \underset{\mathbf{x}}{\text{argmin}}[(\mathbf{b} - \mathbf{A}\mathbf{x})^2 + \lambda \cdot \mathbf{x}^\top \mathbf{x}] \quad (4)$$

where λ is the regularization parameter. RR introduces bias by constraining the set of feasible solutions of the least square problem into a zero-centered L_2 ball even if the real solution $\hat{\mathbf{x}}$ is outside this ball. Compared to OLS, this can significantly reduce the variance and hence the mean squared error of the final estimate $\widetilde{\mathbf{x}}'$, which is especially useful when the variance of $\boldsymbol{\delta}$ is too large (e.g., when the number of observations n is too small or the observations are too noisy). In general, the larger the variance of $\boldsymbol{\delta}$, the larger the regularization should be since increased λ causes the variance to vanish and the bias will dominate the total estimation error. Ultimately, the optimal choice of λ depends on the distribution of $\hat{\mathbf{x}}$ and $\boldsymbol{\delta}$.

3 RELATED WORK

Privacy attacks in Federated Learning: Several privacy attacks have been proposed to learn confidential information about the client's training data in federated learning [8, 22, 25, 36, 38, 39, 45, 47, 61, 75, 76].

In [47], membership inference attack (MIA) is proposed to infer if a specific record is included in the training dataset of the participants. At each round, the adversary first extracts a set of features from every snapshot of the trained global model received from each selected client, such as the output value of the last layers and the hidden layers, the loss values, and the gradient of the loss with respect to the parameters of each layer. These features are used to train a single membership inference model, which is a convolutional neural network, at the end of the training. The attack requires access to each individual update and is therefore ineffective when secure aggregation is used. Finally, the paper has also shown that the attack can be much more effective if the adversary is active instead of passive. [45] introduced the first membership attack under federated learning settings that consists of exploiting the non-zero values of the embedding layer and is therefore only valid for this specific type of layer. Moreover, it requires access to individual updates and is thus also ineffective when secure aggregation is used.

In [36], the authors reconstruct the participation matrix and then the average update per client. The reconstruction of the participation matrix is out of scope in our paper because, in federated learning, the server selects the participating clients according to their availability in each round and therefore knows the participation matrix. However, the reconstruction of the average update per client is naturally the baseline we will consider in our paper (see Section 5.1). To the best of our knowledge, [36] is the first and only work that performs disaggregation in federated learning against secure aggregation by considering a passive adversary (the

server). After reconstructing the average update per client using ordinary least squares (OLS), the server infers the membership information from these reconstructed updates. We show that disaggregating some linear features of the update vector used by the attacker/detector model (e.g., membership inference model) provides more accurate membership inference than disaggregating the whole update vector. Instead of training a single membership model at the very end of the training, we train a distinct membership inference model in each round and combine their inner representations (features) into a final decision with optimization. This approach is more robust especially if some rounds have very inaccurate inference models. Moreover, we also demonstrate the efficacy of our approach on identifying malicious clients.

Recently, a new line of research has focused on active privacy attacks [8, 10, 20, 50, 64], where a malicious server poisons the parameters of the global model in order to reveal a client's update vector. These attacks try to increase the norm of the update vector for a targeted client while decreasing it for non-targeted clients. Some attacks are more restricted than our proposal because they either require large linear layers after the input layer [8, 20] or are designed and evaluated only for FedSGD [43], where each client performs a single SGD update [20, 50, 64], unlike FedAvg [43]. In addition, except for [50], these active attacks only link the recovered update vector to the set of participating clients in a round and do not combine the recovered updates across rounds to infer a property of a client. Although a stealthier active attack has been proposed in [50] that is harder to detect, it is not undetectable, in contrast to passive attacks. In fact, at the cost of additional computational overhead but without harming model quality, all active attacks can be prevented by using cryptographic protocols to verify whether the server manipulates the common model [21, 26–28, 31, 41, 46, 70, 73, 74].

Since active attacks can be detected (or prevented) without degrading model accuracy, they are less practical than passive attacks. Moreover, passive attacks can even be launched offline on more powerful hardware after capturing the protocol messages.

Poisoning attacks in Federated Learning: We focus on integrity attacks [49] and more specifically on poisoning attacks and their defenses. Poisoning attacks are performed either by manipulating the training data (data poisoning) [6, 15, 23, 30, 34, 44, 53, 56, 58, 65] or by directly manipulating the model update (model poisoning) [3, 4, 7, 32, 47] . These attacks can be either targeted by aiming only at reducing the accuracy of the model on some target classes [2, 5] or untargeted, in which case they aim at reducing the accuracy of the model globally without any distinction between the classes [3, 4, 7, 32, 47].

Numerous defenses exist against these attacks, which generally choose the best update in each round [7, 14, 18, 23, 58, 62] or derive a more robust update in each round [56, 69, 71] based, for example, on the median value calculated from the updates sent by the participants to the server [71]. However, they generally require access to each individual update and therefore cannot be employed with secure aggregation because the latter only allows access to the sum of the individual updates. To the best of our knowledge, only [4] and [32] use a more robust update with secure aggregation, however, they also require that each client sends only the sign of

each coordinate's value of the update vector, which slows down convergence. Some works also aim to detect clients launching poisoning attacks [13, 24, 48, 52], assuming that the model updates of every client are available for detection in every round.

In our paper, we identify participants with malicious behavior in federated training even if secure aggregation is used. Specifically, we consider two untargeted poisoning attacks called gradient inversion [4] and gradient ascent attacks [47], which modify the update vector locally so that the performance of the common model declines.

4 THREAT MODEL

The server can infer two types of properties of *each* client: the occurrence of a given target sample in the client's training data (*membership detection*) and whether the client executes poisoning attacks (*misbehaving detection*). In membership detection, the server is a *semi-honest* adversary who aims to identify all clients that have the target sample in their training data. In misbehaving detection, the server is a *honest detector* who aims to identify all *malicious clients* that perform a poisoning attack to degrade the performance of the federated model (at most a fraction ϕ of all clients are malicious). As opposed to previous works [8, 10, 20, 50, 64], the server is *passive* in both cases, that is, it faithfully follows the federated protocol in Section 2.1. This can be enforced by applying verifiable federated learning schemes [21, 26–28, 31, 41, 46, 70, 73].

In misbehaving detection, malicious clients perform poisoning by executing gradient ascent or inversion attacks. In a *Gradient Ascent Attack* [47], malicious clients aim at maximizing the loss by performing gradient *ascent* instead of descent on their own training data. In particular, they update the model parameters locally as $\mathbf{w}_r^i = \mathbf{w}_{r-1}^i + \eta \nabla \ell(D_i; \mathbf{w}_{r-1}^i)$, where η is the learning rate and ℓ is the loss function. This attack attempts to maximize the *average* misclassification rate of the global model and is more effective if the training data of the malicious and benign nodes come from similar distributions. In a *Gradient Inversion Attack* [4], malicious clients faithfully compute their model update $\Delta \mathbf{w}_r^i$ but send $-\Delta \mathbf{w}_r^i$ (instead of $\Delta \mathbf{w}_r^i$) for aggregation.

Since the model update $\Delta \mathbf{w}_r^i$ is computed on the entire local data of a client, the target sample always influences $\Delta \mathbf{w}_r^i$ in membership inference if it is included in the training data. Likewise, poisoning is often executed in each round by every malicious client and therefore has a direct impact on $\Delta \mathbf{w}_r^i$. Hence, the server can train a (supervised) binary detector model M_r to recognize such changes in $\Delta \mathbf{w}_r^i$ and tell only from the model update of a client whether it has the tested property in round r: $M_r(\Delta \mathbf{w}_r^i)$ denotes the confidence of the server that the client has the target sample in its training data in membership detection or that it performs poisoning in misbehaving detection. To train the detector model M_r, an auxiliary (or shadow) dataset D^{aux} is also available to the server, which has sufficiently similar distribution as the clients' training data, though D^{aux} does not include any training samples of any honest client. The availability of an auxiliary dataset is a natural assumption of any supervised inference model and not specific to our proposal. Our approach can also be generalized to any unsupervised or semi-supervised inference model M_r as long as it uses a linear map of the gradients (see Section 5.2 for details). Also,

D^{aux} can be generated synthetically: at the end of the federated learning protocol, the final common model is inverted[2] to generate synthetic training data, and M_r is trained with such synthetic data for property reconstruction[3].

For detection, the individual model updates $\Delta \mathbf{w}_r^i$ are not accessible due to secure aggregation, however, the server can access and record their sum $\sum_i \mathbf{A}_{r,i} \Delta \mathbf{w}_r^i$ in each round as well as the intermediate snapshots T_r of the common model. In addition, the complete participation matrix $\mathbf{A}$ is known to the server, which is a reasonable assumption. Otherwise, the server can exploit side information to reconstruct $\mathbf{A}$ (see [36] for details).

5 PROPERTY RECONSTRUCTION

We show how the server can reconstruct the property information of every client accessing only the aggregated model updates. We present two reconstruction approaches in this section. In the first naive approach, described in Section 5.1, the server disaggregates the sum of update vectors into the individual update of every client and applies the trained detector model M_r on each disaggregated update vector separately. However, the error of this approach can be proportional to the update (model) size in the worst case. Hence, we improve this naive approach and rather disaggregate the linear features of the aggregated update vector, which are used by the detector model M_r. The server finds client-specific properties that maximize the observation probability of these disaggregated features. This improved approach is called PROLIN and described in Section 5.2.

5.1 Naive property reconstruction with gradient disaggregation

The naive reconstruction technique is based on [36] and consists of three steps: (1) reconstructing the expected update vector for every client, (2) training the detector model M_r per round to predict property P from the reconstructed updates, and (3) combining the per-round model predictions to make the final decision about the property of each client.

5.1.1 Gradient reconstruction: The update vector $\Delta \mathbf{w}_r^i$ of every client i is changing over the rounds due to the stochasticity of learning. Still, it is possible to approximate the mean of these per-round updates of a client (i.e., a single "average" update vector per client) with linear regression as follows.

Suppose that the aggregation is described as

$$\mathbf{b}_r = \sum_i^N \mathbf{A}_{r,i} \Delta \mathbf{w}_r^i = \sum_i^N \mathbf{A}_{r,i} \Delta \hat{\mathbf{w}}_i + \xi_r \qquad (5)$$

where $\Delta \hat{\mathbf{w}}_i$ is the expected update vector of client i that we want to reconstruct, and ξ_r represents a vector of independent, unobserved random variables that accounts for the aforementioned stochasticity of learning and models the variance of the individual updates over the rounds ($\Delta \mathbf{w}_r^i, \Delta \hat{\mathbf{w}}_i, \xi_r \in \mathbb{R}^z$ for all i and r). Given $\mathbf{b}_r$ and $\mathbf{A}$, Eq. (6) defines z systems of linear equations (one per update

---

[2]Model inversion can be performed by training a Generative Adversarial Network (GAN) where the discriminator is the final common model and the trained generative model is used to produce synthetic data [59].

[3]In that case, reconstruction is performed after federated learning, if all model updates are recorded during training.

coordinate), each with n equations over $z \times N$ unknowns altogether, which can be approximated by OLS. Formally,

$$\widetilde{\mathbf{W}} = \underset{\mathbf{x} \in \mathbb{R}^{N \times z}}{\arg\min} ||\mathbf{B} - \mathbf{A}\mathbf{x}||_F^2 \qquad (6)$$

where $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n) \in \mathbb{R}^{n \times z}$, and $|| \cdot ||_F$ is the Frobenius norm. According to the Gauss-Markov Theorem, $\widetilde{\mathbf{W}}$ is the best unbiased estimator of $\hat{\mathbf{W}} = (\Delta \hat{\mathbf{w}}_1, \ldots, \Delta \hat{\mathbf{w}}_N) \in \mathbb{R}^{N \times z}$ if $\omega_1, \ldots, \omega_n$ are uncorrelated, have zero mean and identical finite variance.

5.1.2 Training the detector model M_r: The server trains a per-round detector model M_r on D^{aux} in order to infer the property P from the reconstructed expected update $\Delta \widetilde{\mathbf{w}}_i$ for each client as follows:

First, the server creates two disjoint sets of batches $\mathbb{B}^+$ and $\mathbb{B}^-$ from D^{aux}, which are used to generate updates with and without property P, respectively. For membership detection, every batch in $\mathbb{B}^+$ includes the target sample whose membership is detected, while every batch in $\mathbb{B}^-$ excludes the same target sample. Then, provided with the common model T_{r-1} in round r, the server creates the (balanced) training data $D' = D^+ \cup D^-$ such that $D^+ = \{(\Delta \mathbf{w}_r^B, \text{True}) | B \in \mathbb{B}^+\}$ and $D^- = \{(\Delta \mathbf{w}_r^B, \text{False}) | B \in \mathbb{B}^-\}$, where $\Delta \mathbf{w}_r^B$ is the update of model T_{r-1} computed on batch B. For misbehaving detection, $D^- = \{(\Delta \mathbf{w}_r^B, \text{False}) | B \in \mathbb{B}^-\}$ is the set of faithfully computed updates, and $D^+ = \{(\Delta \overline{\mathbf{w}}_r^B, \text{True}) | B \in \mathbb{B}^+\}$ where $\Delta \overline{\mathbf{w}}_r^B$ is defined according to the actual poisoning attack to be detected: $\Delta \overline{\mathbf{w}}_r^B = -\Delta \mathbf{w}_r^B$ for a gradient inversion attack, whereas $\Delta \overline{\mathbf{w}}_r^B$ is obtained by maximizing the loss function on B for a gradient ascent attack (see Section 4).

5.1.3 Property inference: The detector model M_r is applied on the reconstructed expected update $\Delta \widetilde{\mathbf{w}}_i$ for every client i, which results in r individual decisions per client. These decisions are averaged to obtain the final decision about the property of each client.

5.2 PROLIN: Property reconstruction from linear features

The above technique applies OLS to reconstruct every single coordinate of the expected update vector separately. Since the detector model M_r combines every reconstructed gradient coordinate into a single decision, the reconstruction error per coordinate can accumulate and impact the decision, especially if z is large.

We instead propose to first reconstruct t linear features of every individual update vector ($t \ll z$), that capture the relevant property information, and then to infer the property values in this linear feature space. As aggregation is also a linear operation, gradient disaggregation corresponds to feature disaggregation in the feature space, therefore property inference can also be executed in this linear subspace of the gradient vectors with an error that is proportional to t (instead of z).

To make it more concrete, let $\tau_i \in \{0, 1\}$ denote a binary variable indicating whether client i has property P. Our goal is to find the property assignment $\tau_{\max} = (\tau_1, \ldots \tau_N)$ with the largest likelihood $L(\tau | \mathbf{b}_1, \ldots, \mathbf{b}_n)$ given the observed gradient aggregates as constraints, that is, $\tau_{\max} = \arg\max_\tau L(\tau | \mathbf{b}_1, \ldots, \mathbf{b}_n)$. Let $g_r : \mathbb{R}^z \to \mathbb{R}^t$ be a linear function that maps the update vector from the larger gradient space into a smaller feature space where the property

inference of an update is still accurate. In other words, g_r performs feature reduction so that property-relevant information is preserved. In that case, the above likelihood maximization in the gradient space (given the gradient aggregates) is roughly equivalent to likelihood maximization in the feature space (given the feature aggregates) due to the linearity of aggregation:

$$\tau_{\max} = \underset{\tau}{\arg\max}\, L\left(\tau | \mathbf{b}_1, \ldots, \mathbf{b}_n\right)$$

$$\approx \underset{\tau, \mathbf{X} \in \mathbb{X}}{\arg\max} \prod_i \left(\tau_i \cdot \prod_r p(\mathbf{X}_{r,i} | \tau_i = 1) \right.$$

$$\left. + (1 - \tau_i) \cdot \prod_r p(\mathbf{X}_{r,i} | \tau_i = 0) \right) \quad (7)$$

where $\mathbf{X}_{r,i} \approx g_r(\Delta \mathbf{w}_r^i)$ is the individual feature vector of client i in round r whose per-round aggregates are given as constraints: $\mathbb{X} = \{\mathbf{X} \in \mathbb{R}^{n \times N \times t} \mid \sum_i \mathbf{A}_{r,i} \mathbf{X}_{r,i} = g_r(\mathbf{b}_r)\}$, and $p(\mathbf{X}_{r,i} | \cdot)$ is approximated on the auxiliary data D^{aux}. Owing to the linearity of g_r, the server can easily compute the feature aggregates by applying g_r on the observed gradient aggregates $\mathbf{b}_r$:

$$g_r(\mathbf{b}_r) = g_r \left(\sum_i^N \mathbf{A}_{r,i} \Delta \mathbf{w}_r^i \right) = \sum_i^N \mathbf{A}_{r,i} \cdot g_r(\Delta \mathbf{w}_r^i)$$

$$\approx \sum_i^N \mathbf{A}_{r,i} \cdot \mathbf{X}_{r,i} \quad (8)$$

Therefore, the server can solve Eq. (7) and find a slightly biased approximation of $\tau_{\max}$ in the feature space jointly with the most likely disaggregation $\mathbf{X}$ of the known feature aggregates (see Appendix A for a more detailed argument).

However, the individual features $\mathbf{X}$ are unobserved, other than their per-round aggregates, therefore the above likelihood maximization is overly complex: Eq. (7) has a large number of variables ($\mathbf{X}$ and τ) and much fewer observations ($g_1(\mathbf{b}_1), \ldots, g_n(\mathbf{b}_n)$). This would yield an inaccurate approximation of $\tau_{\max}$ even if t is small. Hence we introduce additional constraints for the purpose of regularization: The server computes the expected feature vector of a client from the known feature aggregates with linear regression and requires that these expected feature vectors match the mean $\sum_r \mathbf{X}_{r,i}/n$ of the reconstructed individual feature vectors of the same client. Linear regression is less likely to overfit with $t \cdot N$ variables, especially if $N < n$, therefore can provide realistic constraints for the optimization problem in Eq. (7) and decrease the variance of its solution (see Appendix B).

Although the approximation of $\tau_{\max}$ is biased in the feature space, it has a smaller variance than in the gradient space, which can eventually outbalance the bias and result in a more accurate property inference. This is detailed in Appendix A and also shown empirically in Section 6. Indeed, Eq. (7) has fewer variables in the feature space and the regression can also be more accurate in this t-dimensional space. We stress that *the accurate approximation of $\tau_{\max}$ in the feature space is only feasible because g_r, as well as gradient aggregation, are linear.*

Our proposal has four main steps, which are also summarized in Table 1:

(1) **Training the linear feature extractor g_r:** The server learns the per-round feature extractor g_r on D^{aux}.

(2) **Computing the distribution of linear features:** The conditional probabilities $p(\mathbf{X}_{r,i} | \tau_i)$ in Eq. (7) are approximated with the client-independent feature distribution in round r, that is, the output distribution of g_r on the held-out data D^{aux}.

(3) **Reconstructing the expected linear features:** For the purpose of regularization, the expected linear feature vector of every client is reconstructed from the feature aggregates $g_1(\mathbf{b}_1), \ldots, g_n(\mathbf{b}_n)$ with linear regression.

(4) **Property inference:** Given the feature distributions from Step 2 and the reconstructed expected features per client from Step 3, the most likely property assignment $\tau_{\max} = \arg\max_{\tau} L(\tau | \mathbf{b}_1, \ldots, \mathbf{b}_n)$ is approximated by solving Eq. (7).

5.2.1 Training the feature extractor. The server trains a detector model $M_r = h_r \circ g_r$ per round, which first extracts t linear features of the update by applying $g_r : \mathbb{R}^z \rightarrow \mathbb{R}^t$ on the update vector and then applies a non-linear function $h_r : \mathbb{R}^t \rightarrow \{0, 1\}$ on these linear features to recognize property P. Since the output of h_r is the tested property, h_r pushes g_r to capture the property relevant information from the update vector. For example, if g_r is a scalar linear function and h_r is the sigmoid function, then $M_r(\mathbf{x}) = 1/(1 + \exp(-g_r(\mathbf{x})))$ defines logistic regression. In that case, only a single feature is extracted from the entire update vector ($t = 1$).

To train M_r, the server creates training data $D' = D^+ \cup D^-$, which consists of model updates with (D^+) and without (D^-) property P just as described in Section 5.1.2. After splitting D' into a training and testing part, the server trains M_r on the training part of D'.

5.2.2 Computing the distribution of linear features. The output distributions of g_r conditioned on P are approximated on the testing part of D' in every round r: f_r^+ denotes the Probability Density Function (PDF) of a random variable describing the output of g_r on D^+, and f_r^- denotes the PDF of a random variable describing the output of g_r on D^-. As the server has no client-specific background knowledge to compute $p(\mathbf{X}_{r,i} | \tau_i)$ in Eq. (7), it approximates $p(\mathbf{X}_{r,i} | \tau_i = 1)$ with $f^+(\mathbf{X}_{r,i})$ and $p(\mathbf{X}_{r,i} | \tau_i = 0)$ with $f^-(\mathbf{X}_{r,i})$.

The distributions of a single linear feature are illustrated in Figure 2 for membership detection, where both f_r^+ and f_r^- have normal distributions. Figure 2 shows that f_r^+ and f_r^- are well-separated at the beginning of the training, indicating an accurate detector model M_r. However, as the training progresses, the two distributions start to overlap, which implies a less accurate prediction. In fact, for membership inference, M_r is usually more accurate at the beginning of the training when the gradients are larger and the target sample is likely to have a noticeable impact on the gradient. Unlike the naive approach in Section 5.1 that averages the output of M_r over the rounds, PROLIN considers these potentially different per-round distributions of the features for more accurate inference.

If the output of g_r is high-dimensional (e.g., $t > 1$) then the above approximation of g_r with sampling becomes inaccurate unless D' is sufficiently large. In that case, estimating $p(\mathbf{X}_{r,i} | \tau_i)$ with the output distribution of the entire detector model $M_r = h_r \circ g_r$ can be a more accurate approach. Since h_r is designed to reduce the dimensionality of g_r, its output distribution conditioned on P should

(a) $r = 1$ **(b)** $r = 100$ **(c)** $r = 200$ **(d)** $r = 300$

Figure 2: Distribution of a single linear feature $g_r(\Delta \mathbf{w}_r^B)$ for $B \in D^{aux}$ for membership detection depending on the round r on MNIST ($t = 1$). Overlap coefficient (OVL) is the overlap area between f_r^+ and f_r^- colored with gray, measuring the performance of detector model M_r.

provide a fairly accurate approximation of the feature distributions f_r^+ and f_r^-.

5.2.3 Reconstructing the expected linear features. As the server can compute the feature aggregates based on Eq. (8), it can also disaggregate them into the expected linear feature vector per client, similarly to gradient disaggregation in Section 5.1.1. However, instead of ordinary least square, we use a weighted version of ridge regression to address the potentially large variance of linear features.

More precisely, suppose that the aggregation of linear features can be described as

$$g_r(\mathbf{b}_r) = \sum_{i}^{N} \mathbf{A}_{r,i} \cdot g_r(\Delta \mathbf{w}_r^i) = \sum_{i}^{N} \mathbf{A}_{r,i} \cdot \hat{\mathbf{g}}_i + \boldsymbol{\theta}_r \qquad (9)$$

where $\hat{\mathbf{g}}_i \in \mathbb{R}^t$ is the vector of expected linear features of client i that we want to reconstruct, and $\boldsymbol{\theta}_r \in \mathbb{R}^t$ is a vector of random noise that accounts for the stochasticity of learning and models the variance of the individual feature vectors over the rounds. Since the variance of $\boldsymbol{\theta}$ can be large, or the number of observations may be less than the number of features to recover ($n < t$), we use ridge regression with L_2 regularization of the reconstructed linear features (see Section 2.2):

$$\widetilde{\mathbf{G}} = \underset{\mathbf{x} \in \mathbb{R}^{N \times t}}{\operatorname{argmin}} \mathbf{v} ||\mathbf{G} - \mathbf{A}\mathbf{x}||_F^2 + \lambda ||\mathbf{x}||_F^2 \qquad (10)$$

where $\widetilde{\mathbf{G}} \in \mathbb{R}^{n \times t}$ is an approximation of $\hat{\mathbf{G}} = (\hat{\mathbf{g}}_1, \ldots, \hat{\mathbf{g}}_N) \in \mathbb{R}^{n \times t}$, $\mathbf{G} = (g_1(\mathbf{b}_1), \ldots, g_n(\mathbf{b}_n)) \in \mathbb{R}^{N \times t}$, and $\mathbf{v}_r \in [0, 1]$ ($1 \leq r \leq n$) denotes some measure of the performance of M_r. If M_r provides an accurate prediction of property P, then the residual error $||g_r(\mathbf{b}_r) - \sum_i \mathbf{A}_{r,i}\mathbf{x}_i||_2$ in round r should have larger weight in the objective function because the output of g_r is likely to have smaller variance. Eq. (10) can be solved efficiently by solving the objective function in Eq. (4) for each linear feature individually.

5.2.4 Property inference. Given the feature aggregates $g_1(\mathbf{b}_1), \ldots, g_n(\mathbf{b}_n)$ from Eq. (8) and the reconstructed expected feature vectors $\widetilde{\mathbf{G}}$ from Eq. (10), the server solves the

following regularized version of Eq. (7):

$$\max_{\tau_i, \mathbf{X}} \quad \sum_i \log \left(\tau_i \cdot \prod_r f_r^+(\mathbf{X}_{r,i}) + (1 - \tau_i) \cdot \prod_r f_r^-(\mathbf{X}_{r,i}) \right)$$

$$\text{s.t.} \quad \sum_i \mathbf{A}_{r,i} \mathbf{X}_{r,i} = g_r(\mathbf{b}_r) \qquad \text{(Constraint 1)}$$

$$\sum_{r \in R(i)} \frac{\mathbf{X}_{r,i}}{|R(i)|} = \widetilde{\mathbf{G}}_i \qquad \text{(Constraint 2)}$$

$$\tau_i \in \{0, 1\}, \mathbf{X} \in \mathbb{R}^{n \times N \times t} \qquad \text{(Constraint 3)}$$

where f_r^+ and f_r^- denote the PDFs of the linear features conditioned on property P (see Section 5.2.2), and $R(i)$ is the set of rounds in which client i participates. Constraint 1 requires that the reconstructed linear features $\mathbf{X}_{r,i}$ should produce the known feature aggregates. Constraint 2 provides regularization by requiring that the mean of the reconstructed linear features $\mathbf{X}_{r,i}$ should match the expected linear feature vector per client (see Eq. (9) and Eq. (10)). Finally, Constraint 3 pushes the optimization to find an integer-valued solution for τ_i, as a client either has or does not have property P.

The above optimization problem contains integer variables τ_i, which makes the problem NP-complete. Hence, we relax the problem into

$$\min_{\tau_i, \mathbf{X}} \quad \gamma_1 \mathcal{L}_{\text{ml}} + \gamma_2 \mathcal{L}_{\text{reg}} + \gamma_3 \mathcal{L}_{\text{lstsq}}$$

$$\text{where} \quad \mathcal{L}_{\text{ml}} = -\sum_i \log \left(\tau_i \cdot \prod_r f_r^+(\mathbf{X}_{r,i}) + (1 - \tau_i) \cdot \prod_r f_r^-(\mathbf{X}_{r,i}) \right)$$

$$\mathcal{L}_{\text{reg}} = \sum_{i, r \in R(i)} \left\| \frac{\mathbf{X}_{r,i}}{|R(i)|} - \widetilde{\mathbf{G}}_i \right\|_F^2$$

$$\mathcal{L}_{\text{lstsq}} = \sum_{i,r} \mathbf{v}_r \|g_r(\mathbf{b}_r) - \mathbf{A}_{r,i}\mathbf{X}_{r,i}\|_F^2$$

$$\text{s.t.} \quad 0 \leq \tau_i \leq 1, \mathbf{X} \in \mathbb{R}^{n \times N \times t}$$

where $\gamma_1, \gamma_2, \gamma_3$ are the weighting factors of each loss in the objective function. Although this relaxed version is still non-convex if f_r^- or f_r^+ are also non-convex, the variable τ_i is now continuous in $[0, 1]$ and hence can be approximated with projected gradient descent (e.g., using an automatic differentiation framework such as PyTorch [51]).

We provide a theoretical justification of PROLIN in Appendix A.

Table 1: PROLIN

Input: (1) observed aggregate $\mathbf{b}_r = \sum_i^N \mathbf{A}_{r,i} \cdot \Delta \mathbf{w}_r^i$ per round; (2) participation matrix $\mathbf{A}$; (3) auxiliary data D^{aux}; (4) property P; (5) federated model T_r per round

Output: $\boldsymbol{\tau} \in \{0, 1\}^N$ ($\tau_i = 1$ if client i has property P)

Begin

(1) **Training property inference model:** For every round r use auxiliary data D^{aux} to

  (a) train detector model $M_r = h_r \circ g_r$

  (b) compute weight $\mathbf{v}_r$, which is proportional to some performance metric of M_r

(2) **Computing the distribution of linear features:** Approximate the PDFs f_r^+ and f_r^- of the linear features g_r conditioned on property P on D^{aux}

(3) **Reconstructing expected linear features:**

  (a) Compute the aggregation of linear features $\mathbf{G}_r$ in every round r:

$$\mathbf{G}_r = g_r(\mathbf{b}_r) = \sum_i^N \mathbf{A}_{r,i} \cdot g_r(\Delta \mathbf{w}_r^i)$$

  (b) Approximate the expected linear features $\sum_{r \in R(i)} \frac{g_r(\Delta \mathbf{w}_r^i)}{|R(i)|}$ per client i by $\widetilde{\mathbf{G}}_i$, where

$$\widetilde{\mathbf{G}} = \operatorname*{argmin}_{\mathbf{x} \in \mathbb{R}^{N \times t}} \mathbf{v} ||\mathbf{G} - \mathbf{A}\mathbf{x}||_F^2 + \lambda \cdot ||\mathbf{x}||_F$$

(4) **Property inference:** Solve the following optimization problem for $\boldsymbol{\tau}$:

$$\min_{\tau_i, \mathbf{X}} \quad \gamma_1 \mathcal{L}_{\text{ml}} + \gamma_2 \mathcal{L}_{\text{reg}} + \gamma_3 \mathcal{L}_{\text{lstsq}}$$

$$\text{where} \quad \mathcal{L}_{\text{ml}} = -\sum_i \log \left(\tau_i \prod_r f_r^+(\mathbf{X}_{r,i}) \right.$$

$$\left. + (1 - \tau_i) \prod_r f_r^-(\mathbf{X}_{r,i}) \right)$$

$$\mathcal{L}_{\text{reg}} = \sum_{i, r \in R(i)} \left\| \frac{\mathbf{X}_{r,i}}{|R(i)|} - \widetilde{\mathbf{G}}_i \right\|_F^2$$

$$\mathcal{L}_{\text{lstsq}} = \sum_{i,r} \mathbf{v}_r \left\| \mathbf{G}_r - \mathbf{A}_{r,i} \mathbf{X}_{r,i} \right\|_F^2$$

$$\text{s.t.} \quad 0 \le \tau_i \le 1, \mathbf{X} \in \mathbb{R}^{n \times N \times t}$$

End

6 EVALUATION

In this section, we demonstrate that PROLIN can effectively disaggregate the linear features of different detector models. Although we focus on two specific detection tasks (membership inference and misbehaving detection), we emphasize that PROLIN is a general approach that can disaggregate any linear function and hence potentially reconstruct various client-specific properties given their accurate detector models. Moreover, even if the detector model is not consistently accurate in every round, PROLIN takes the best combination of these per-round models to have a quasi-optimal property inference.

6.1 Dataset

We compare the performance of PROLIN with different property reconstruction techniques. We evaluate all approaches on the following datasets:

- The MNIST database of handwritten digits. It consists of 28 x 28 grayscale images of digit items and has 10 output classes. The training set contains 60,000 data samples, while the test/validation set has 10,000 samples [37].
- The CIFAR-10 dataset consists of 60,000 32x32 color images in 10 classes, with 6000 images per class. There are 50,000 training images and 10,000 test images [35].
- Fashion-MNIST database of fashion articles consists of 60,000 28x28 grayscale images of 10 fashion categories, along with a test set of 10,000 images [66].

For each dataset, we randomly select 10% of the training set for auxiliary data D^{aux}. Therefore, the server has $|D^{aux}| = 6000$ samples for MNIST and Fashion-MNIST and $|D^{aux}| = 5000$ samples for CIFAR-10. D' is generated from D^{aux} as described in Section 5.2.1, where $|D'| = 2 \cdot |D^{aux}|$ in our evaluations[4]. We use 80% of D' to train M_r and 20% to compute weights $\mathbf{v}$ in Eq. (10) as well as distributions f_r^+ and f_r^-.

6.2 Model Architectures

As in [36], we use LeNet neural network as the global model T_r with the following architectures:

- For MNIST and Fashion-MNIST, we use two 5x5 convolution layers (the first with 10 filters, the second with 20), each followed by 2x2 max pooling, a dropout layer with ratio set to 0.5, and two fully connected layers with 50 and 10 neurons, respectively. A dropout layer separates the two fully connected layers.
- For CIFAR-10, we use two 5x5 convolution layers (the first with 6 filters, the second with 16), each followed by 2x2 max pooling and three fully connected layers with 120, 84 and 10 neurons, respectively.

6.3 Property reconstruction

We consider the detection of three properties: (1) membership information of a randomly chosen target sample, malicious behavior by launching (2) gradient inversion or (3) gradient ascent attack.

6.3.1 Approaches. We compare the following approaches to reconstruct the above properties.

BASELINE: This is based on gradient reconstruction from [36], which is also described in Section 5.1. As opposed to [36], we train a distinct inference model M_r per round instead of a single model over all the rounds and average the decisions of these per-round models. This "ensemble" approach is more accurate since M_r can have very different performances per round. M_r is a logistic regression model.

PROLIN: This is based on feature reconstruction and described in

---

[4]Since D' contains batches of D^{aux}, it can have larger size.

Section 5.2. It is instantiated with a single linear feature ($t = 1$) and $M_r = h \circ g_r$ is a logistic regression model where h is the sigmoid function. As M_r is trained only on the update of a single round, this simple model is accurate and also fast to train. Following from empirical observations (also illustrated in Figure 2), the feature distributions f_r^+ and f_r^- are approximated to be normal[5] whose means and variances equal the empirical means and variances of the single linear feature g_r on the testing part of D' conditioned on property P. The weight $\mathbf{v}_r$ per round is set to $1 - \text{OVL}_r$, where OVL_r is the overlapping coefficient between the distributions of f_r^+ and f_r^+ and is a value between 0 and 1 that measures the overlap area of the two probability density functions (also illustrated in Figure 2). Therefore, a coefficient with a small value means that M_r is accurate and vice versa[6]. The regularization parameter λ is fixed to 5 for all experiments. The weights γ_1, γ_2, and γ_3 of different losses in the objective function of PROLIN (see Section 5.2.4) are adjusted dynamically during optimization using the technique in [42].

OLS: This is based on feature reconstruction that infers the property by applying the sigmoid function h on the approximation $\widetilde{\mathbf{G}}_i$ of the single expected linear feature of a client ($t = 1$). $\widetilde{\mathbf{G}}_i$ is obtained by solving Eq. (10) with $\mathbf{v} = \mathbf{1}$ and $\lambda = 0$, that is, each round has equal weight and there is no regularization. $\tau_i = 1$ if $h(\widetilde{\mathbf{G}}_i) > 0.5$.

REG: This is based on a feature reconstruction like OLS, except that *ridge regression* is applied to obtain $\widetilde{\mathbf{G}}_i$ by solving Eq. (10) with $\mathbf{v}$ and λ as defined in PROLIN. $\tau_i = 1$ if $h(\widetilde{\mathbf{G}}_i) > 0.5$.

6.3.2 Experimental setup. We perform 3 runs for each experiment and average the results obtained over these 3 runs. For the global model, we use a batch size of 10 for MNIST and Fashion-MNIST and 25 for CIFAR-10 and a batch size of 10 to train M_r. The learning rate η is set to 0.01 to train the global model for MNIST and Fashion-MNIST, 0.1 for CIFAR-10, and 0.001 to train M_r in order to identify g_r. SGD is used to train all models. For membership inference attack (MIA), the target sample is chosen uniformly at random in each experiment and remains fixed over all the rounds for one experiment. All clients with the membership property have this sample in their local training data. The number of federated rounds is $n = 300$, the number of all clients is $N = 50$, and a fraction of $\phi = 0.1$ of all clients are positive (i.e., have the property). In each round, a fraction of $C = 0.2$ of all clients are selected uniformly at random to send their model update for aggregation after performing a single epoch of local training on their own training data. Each client has the same number of training samples, which are assigned to the clients uniformly at random at the very beginning of the training. All settings are summarized in Table 2 in the appendix.

6.4 Results

Figures 3, 4 and 5 show the results for the membership inference attack (MIA) and for the detection of gradient inversion (INV) and gradient ascent attacks (GAA), respectively. We report the F1-score of the detection, which is the harmonic mean of the precision and

recall, where the precision is the number of correctly detected positive clients divided by the number of all clients who are detected as positive, and the recall is the number of correctly detected positive clients divided by the number of all positive clients.

Out of the three detection tasks, membership information is the most difficult (Figure 3) and gradient ascent is the easiest to detect (Figure 5). Indeed, a single target sample has less significant impact on the aggregated model update as the training progresses (see Section 5.2.1), while gradient manipulation depends only on the performance of the common model T_{r-1}. Unlike gradient ascent, gradient inversion does not modify the magnitude of the update, hence it is more difficult to detect (Figures 4 and 5).

6.4.1 Feature vs. gradient reconstruction. Feature reconstruction (OLS, REG, PROLIN) is superior to gradient reconstruction (BASELINE) on almost all tasks, albeit to different degrees. The difference is the most salient on CIFAR-10, which shows that **feature reconstruction is indeed a more appealing approach for property inference especially if the common model is more complex**. The exception is the detection of the gradient ascent attack when the dataset size is 50 (see Figure 5.h), where BASELINE is more accurate than other approaches. Indeed, Figure 6 depicts the L_2-norm of the weights $\boldsymbol{\alpha}$ of the linear model g_r depending on r for this scenario. This shows that the norm falls below 1 after round 100, which means that the reconstruction error of BASELINE can be less than for other approaches as explained in Appendix B. Feature reconstruction also generally shows a smaller variance in accuracy over the rounds, and it converges faster especially when clients have larger datasets.

6.4.2 PROLIN vs. BASELINE. For MIA, BASELINE reaches the maximum F1-score of 0.83 in Figure 3.e at round 20 and then drops quickly, while PROLIN reaches the peak at round 120 with an F1-score of 79%, providing a more stable performance. Similarly, in Figure 3.b, BASELINE is more accurate at the end of the training, however, PROLIN obtained the maximum F1-score (86%) in this case. In fact, BASELINE reaches the best performance with the GAA shown in Figure 5.h by reaching an F1-score of 100%, while PROLIN reaches 76%. Nevertheless, PROLIN is more accurate overall and has larger F1-scores on average over the rounds. For example, the worst F1-score on all the considered scenarios is 69% for PROLIN, while it is 56% for BASELINE (Figure 4.h). In addition, PROLIN **has a more stable performance with a smaller variance over the rounds than BASELINE**. Indeed, in many cases, the F1-score of BASELINE has a larger variance (Figure 3.f-h, Figure 4.g-h), which makes it difficult to choose the round number where it provides good performance: Even if the detector can access all the rounds, it must choose one where it is supposed to obtain the final detection result. It is therefore crucial to have a stable performance to ensure that the reconstruction remains accurate over a sufficiently wide range of rounds. Finally, PROLIN **also converges much faster to good F1-scores in general** (Figure 3.e-h, Figure 4.a-h, Figure 5.f-g). This is also important because the server stops federated learning as soon as the global model reaches acceptable performance, and therefore the reconstruction must also be accurate by then.

6.4.3 PROLIN vs. REG and OLS. PROLIN is also superior to REG and OLS. Although this difference is more apparent when PROLIN

---

[5]The PDF of a normal random variable with mean μ and variance σ^2 is $\frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$

[6]$1 - \text{OVL}$ is also equivalent to Youden's index since OVL is the sum of False Negative and False Positive Ratios

Figure 3: Membership detection on the MNIST, Fashion-MNIST, and CIFAR-10 datasets by varying the size of the local dataset per client ($|D_i|$)

Figure 4: Misbehaving detection (gradient inversion attack) on the MNIST, Fashion-MNIST, and CIFAR-10 datasets by varying the size of the local dataset per client ($|D_i|$)

is compared with OLS (Figure 3.a-h, Figure 4.f-h, Figure 5.g-h), PROLIN is also more accurate than REG over all the rounds when a MIA is executed on CIFAR-10 (Figure 3.g-h). For the other cases, they have very similar performance: $\mathcal{L}_{ml}$ evaluates the likelihood with the round specific feature distributions f_r^+ and f_r^-, while REG uses the fixed sigmoid function h in all the rounds. Since h infers

Figure 5: Misbehaving detection (gradient ascent attack) on the MNIST, Fashion-MNIST, and CIFAR-10 datasets by varying the size of the local dataset per client ($|D_i|$)

Figure 6: L_2-norm of the weights of the linear model g_r on CIFAR10 with client datasize 50 when a gradient ascent attack is detected.

the property from a single "average" feature vector of a client, it disregards the per-round feature distributions unlike PROLIN, which can lead to a lower accuracy when these distributions differ over the rounds (see Fig. 2 for illustration). The difference between the two approaches becomes significant when M_r is inaccurate and the distributions f_r^+ and f_r^- have a larger overlap (i.e., their difference is smaller towards the end of the training, which indicates decreasing confidence, while h always assigns the same confidence to any feature value across rounds independently of r). REG only uses the accuracy of M_r as weights $\mathbf{v}$ in linear regression, while PROLIN considers, in addition, all feature distributions directly during optimization, which is more accurate. To confirm this, we report the overlapping coefficient (OVL) between these distributions in Figure 7.b, which shows that this coefficient is almost 0 over all rounds when the detection of a gradient inversion attack is considered (on CIFAR-10 with local dataset size $|D_i| = 25$). This explains why the

performances of PROLIN and REG are almost the same (Figure 4.g). However, OVL increases over the rounds for MIA (Figure 7.a), and PROLIN becomes superior to REG (Figure 3.g).

(a) Membership inference attack

(b) Gradient inversion attack

Figure 7: OVL coefficients over the rounds for membership and misbehaving detection on CIFAR-10 with dataset size 25.

6.4.4 Impact of round number. In general, the accuracy of all approaches increases with the number of rounds. This is expected because the number of different observed aggregates also increases, which makes linear regression more accurate. Feature reconstruction techniques converge faster than BASELINE in general (except Figure 5.h as detailed above): for MNIST and FMNIST, PROLIN and REG reach an accuracy of 0.8 by round 80-100 on gradient inversion and ascent detection due to regularization. The variance of the reconstructed features is larger if the number of observed aggregates is too small at the beginning of the training or the attacker model M_r is too inaccurate towards the end of the training

for MIA. The effect of this noise is mitigated in PROLIN and REG by regularization, which is not the case for BASELINE and OLS. Moreover, PROLIN **and REG need roughly** $2N$ **rounds to converge**, while BASELINE generally needs more rounds. This is remarkable considering the fact that any approach needs at least N rounds to converge.

6.4.5 Impact of dataset size. **As the dataset size increases, all approaches decline on MIA**, as shown in Figure 3.a-h. Indeed, as the model update is computed from the average gradient of all training samples, the impact of a single target sample on the update is smaller if the number of training samples is large. This results in larger variance of the linear features (illustrated in Figure 2), which is mitigated by regularization in PROLIN and REG. The detections of gradient inversion and ascent attacks are less impacted by the dataset size (Figure 4 and 5) because these attacks directly manipulate the update vectors.

6.4.6 Impact of the number of clients. In order to evaluate the impact of the total number of participants N on property inference, we perform an experiment on CIFAR-10 considering the MIA and gradient ascent attacks. We set N to 50, 100 and 200 but fix the number of positive clients to 5, and the dataset size is 25. Figure 8 shows that BASELINE and OLS are the most influenced by the increase in the number of clients. For example, BASELINE reaches an F1-score of 100% with 50 clients on the gradient ascent attack, and then it decreases to 69% and then to 21% with 100 and 200 clients, respectively (see Figure 8.d-f). There is a similar decrease for MIA (see Figure 8.a-c). Similarly, OLS decreases from 83% to 62% and then to 24% with 50, 100 and 200 clients, respectively, for the gradient ascent attack. **Although the F1-score also decreases with PROLIN and REG, this accuracy degradation remains less compared to BASELINE and OLS as the number of clients increases**. The increase in the number of clients seems to widen the gap between the F1-score of PROLIN (which remains better) and REG.

7 CONCLUSION

We showed that secure aggregation fails to protect client-specific information. We proposed a technique called PROLIN that uses a linear model (due to the linearity of aggregation in federated learning) to extract and disaggregate features for the inference of client-specific property information.

We evaluated our approach on two different tasks: membership inference and the detection of poisoning attacks. In membership inference, the goal is to identify clients whose training data includes a specific target record. In poisoning detection, the goal is to identify clients that launch untargeted poisoning attacks to degrade the accuracy of the global model. We show that, for both tasks, feature-based reconstruction, and linear models in particular, is surprisingly more accurate than earlier gradient-based reconstruction techniques. Our proposal PROLIN outperforms both the state-of-the-art baseline [36] and our proposed baselines. In addition, PROLIN has more stable accuracy over rounds, converges faster, and is more robust to more complex scenarios such as when the total number of clients increases or when the attacker model is less accurate.

Our approach is passive and therefore undetectable. Although there are techniques to prevent property reconstruction, those approaches usually introduce trade-offs. For example, Differential Privacy can be used to avoid detection but at the cost of reducing the accuracy of the common model. Similarly, a client can selectively launch poisoning or use only a subset of all training samples in certain rounds, which can make property reconstruction less accurate. However, these countermeasures also imply less effective attacks or the slower convergence of the common model if the local dataset is small.

PROLIN is not limited to membership inference and misbehaving detection, it can disaggregate the linear features of any supervised detector model. Therefore, there are several avenues of future work that are facilitated by our novel approach, such as contribution scoring, where a score is assigned to each participant measuring, the quality of its contribution to the common federated model, even if secure aggregation is employed. This would also allow to detect free-rider attacks, where a selfish participant benefits from the global model without any useful contribution. There are also stealthier poisoning (backdoor) attacks [63, 67] than gradient inversion and ascent, whose evaluation is left for future work.

Although supervised detector models are generally more accurate than any unsupervised approaches [40], they are also restricted to detect only the properties that they are trained for. However, if malicious clients are adaptive and know what detector model the server uses, they can evade detection by launching stealthy attacks [68] irregularly over the training. The extension of PROLIN to unsupervised misbehaving detection belongs to future work.

REFERENCES

[1] Gergely Ács and Claude Castelluccia. 2011. I Have a DREAM! (DiffeRentially privatE smArt Metering). In *IH*.
[2] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. 2018. How To Backdoor Federated Learning. *CoRR* abs/1807.00459 (2018). arXiv:1807.00459 http://arxiv.org/abs/1807.00459
[3] Gilad Baruch, Moran Baruch, and Yoav Goldberg. 2019. A Little Is Enough: Circumventing Defenses For Distributed Learning. In *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.), Vol. 32. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2019/file/ec1c59141046cd1866bbbcdfb6ae31d4-Paper.pdf

[4] Jeremy Bernstein, Jiawei Zhao, Kamyar Azizzadenesheli, and Anima Anandkumar. 2018. signSGD with majority vote is communication efficient and fault tolerant. *arXiv preprint arXiv:1810.05291* (2018).

[5] Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. 2019. Analyzing federated learning through an adversarial lens. In *International Conference on Machine Learning*. PMLR, 634–643.

[6] Battista Biggio, Blaine Nelson, and Pavel Laskov. 2012. Poisoning Attacks against Support Vector Machines. In *Proceedings of the 29th International Coference on International Conference on Machine Learning* (Edinburgh, Scotland) *(ICML'12)*. Omnipress, Madison, WI, USA, 1467–1474.

[7] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. 2017. Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent. In *NIPS*. 119–129.

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

[9] Franziska Boenisch, Adam Dziedzic, Roei Schuster, Ali Shahin Shamsabadi, Ilia Shumailov, and Nicolas Papernot. 2022. *All You Need Is Matplotlib, or Federated Learning with Untrusted Servers is Not Private.* Retrieved January 20, 2023 from http://www.cleverhans.io/2022/04/17/fl-privacy.html

[10] Franziska Boenisch, Adam Dziedzic, Roei Schuster, Ali Shahin Shamsabadi, Ilia Shumailov, and Nicolas Papernot. 2023. Is Federated Learning a Practical PET Yet? *arXiv preprint arXiv:2301.04017* (2023).

[11] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2016. Practical secure aggregation for federated learning on user-held data. *arXiv preprint arXiv:1611.04482* (2016).

[12] Kallista A. Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical Secure Aggregation for Privacy-Preserving Machine Learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu (Eds.). ACM, 1175–1191.

[13] Xiaoyu Cao, Jinyuan Jia, Zaixi Zhang, and Neil Zhenqiang Gong. 2023. FedRecover: Recovering from Poisoning Attacks in Federated Learning using Historical Information. In *44th IEEE Symposium on Security and Privacy, SP 2023, San Francisco, CA, USA, May 21-25, 2023*. IEEE, 1366–1383. https://doi.org/10.1109/SP46215.2023.10179336

[14] Hongyan Chang, Virat Shejwalkar, Reza Shokri, and Amir Houmansadr. 2019. Cronus: Robust and heterogeneous collaborative learning with black-box knowledge transfer. *arXiv preprint arXiv:1912.11279* (2019).

[15] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. 2017. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526* (2017).

[16] Yae Jee Cho, Jianyu Wang, and Gauri Joshi. 2020. Client selection in federated learning: Convergence analysis and power-of-choice selection strategies. *arXiv preprint arXiv:2010.01243* (2020).

[17] Olivia Choudhury, Aris Gkoulalas-Divanis, Theodoros Salonidis, Issa Sylla, Yoonyoung Park, Grace Hsu, and Amar Das. 2019. Differential privacy-enabled federated learning for sensitive health data. *arXiv preprint arXiv:1910.02578* (2019).

[18] El Mahdi El Mhamdi, Rachid Guerraoui, and Sébastien Rouault. 2018. The Hidden Vulnerability of Distributed Learning in Byzantium. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer Dy and Andreas Krause (Eds.). PMLR, 3521–3530. http://proceedings.mlr.press/v80/mhamdi18a.html

[19] CBICA Center for Biomedical Image Computing & Analytics. 2020. *The Federated Tumor Segmentation (FeTS) initiative.* Retrieved January 19, 2023 from https://www.med.upenn.edu/cbica/fets/

[20] Liam Fowl, Jonas Geiping, Wojtek Czaja, Micah Goldblum, and Tom Goldstein. 2022. Robbing the fed: Directly obtaining private data in federated learning with modified models. *Tenth International Conference on Learning Representations (ICLR) 2022* (2022).

[21] Anmin Fu, Xianglong Zhang, Naixue Xiong, Yansong Gao, Huaqun Wang, and Jing Zhang. 2020. VFL: A verifiable federated learning with privacy-preserving for big data in industrial IoT. *IEEE Transactions on Industrial Informatics* 18, 5 (2020), 3316–3326.

[22] Chong Fu, Xuhong Zhang, Shouling Ji, Jinyin Chen, Jingzheng Wu, Shanqing Guo, Jun Zhou, Alex X Liu, and Ting Wang. 2022. Label Inference Attacks Against Vertical Federated Learning. In *31st USENIX Security Symposium (USENIX Security 22)*. USENIX Association, Boston, MA.

[23] Clement Fung, Chris JM Yoon, and Ivan Beschastnikh. 2018. Mitigating sybils in federated learning poisoning. *arXiv preprint arXiv:1808.04866* (2018).

[24] Clement Fung, Chris J. M. Yoon, and Ivan Beschastnikh. 2020. The Limitations of Federated Learning in Sybil Settings. In *23rd International Symposium on Research in Attacks, Intrusions and Defenses, RAID 2020, San Sebastian, Spain, October 14-15, 2020*, Manuel Egele and Leyla Bilge (Eds.). USENIX Association, 301–316.

https://www.usenix.org/conference/raid2020/presentation/fung

[25] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. 2020. Inverting Gradients - How easy is it to break privacy in federated learning?. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 16937–16947.

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

[27] Changhee Hahn, Hodong Kim, Minjae Kim, and Junbeom Hur. 2021. Versa: Verifiable secure aggregation for cross-device federated learning. *IEEE Transactions on Dependable and Secure Computing* (2021).

[28] Gang Han, Tiantian Zhang, Yinghui Zhang, Guowen Xu, Jianfei Sun, and Jin Cao. 2022. Verifiable and privacy preserving federated learning without fully trusted centers. *Journal of Ambient Intelligence and Humanized Computing* (2022), 1–11.

[29] Andrew Hard, Chloé M Kiddon, Daniel Ramage, Francoise Beaufays, Hubert Eichner, Kanishka Rao, Rajiv Mathews, and Sean Augenstein. 2018. Federated Learning for Mobile Keyboard Prediction. https://arxiv.org/abs/1811.03604

[30] Matthew Jagielski, Alina Oprea, Battista Biggio, Chang Liu, Cristina Nita-Rotaru, and Bo Li. 2018. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 19–35.

[31] Changsong Jiang, Chunxiang Xu, and Yuan Zhang. 2021. PFLM: Privacy-preserving federated learning with membership proof. *Information Sciences* 576 (2021), 288–311.

[32] Raouf Kerkouche, Gergely Ács, and Claude Castelluccia. 2020. Federated learning in adversarial settings. *arXiv preprint arXiv:2010.07808* (2020).

[33] Raouf Kerkouche, Gergely Ács, Claude Castelluccia, and Pierre Genevès. 2021. Privacy-Preserving and Bandwidth-Efficient Federated Learning: An Application to in-Hospital Mortality Prediction. In *Proceedings of the Conference on Health, Inference, and Learning* (Virtual Event, USA) *(CHIL '21)*. Association for Computing Machinery, New York, NY, USA, 25–35. https://doi.org/10.1145/3450439.3451859

[34] Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning*. PMLR, 1885–1894.

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

[36] Maximilian Lam, Gu-Yeon Wei, David Brooks, Vijay Janapa Reddi, and Michael Mitzenmacher. 2021. Gradient disaggregation: Breaking privacy in federated learning by reconstructing the user participant matrix. In *International Conference on Machine Learning*. PMLR, 5959–5968.

[37] Yann LeCun and Corinna Cortes. 2010. MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/. (2010). http://yann.lecun.com/exdb/mnist/

[38] Oscar Li, Jiankai Sun, Xin Yang, Weihao Gao, Hongyi Zhang, Junyuan Xie, Virginia Smith, and Chong Wang. 2020. Label leakage and protection in two-party split learning. *NeurIPS 2020 Workshop on Scalability, Privacy, and Security in Federated Learning (SpicyFL)* (2020).

[39] Zhuohang Li, Jiaxin Zhang, Luyang Liu, and Jian Liu. 2022. Auditing Privacy Defenses in Federated Learning via Generative Gradient Leakage. *The IEEE / CVF Computer Vision and Pattern Recognition Conference (CVPR)* (2022).

[40] Zhuoran Ma, Jianfeng Ma, Yinbin Miao, Yingjiu Li, and Robert H. Deng. 2022. ShieldFL: Mitigating Model Poisoning Attacks in Privacy-Preserving Federated Learning. *IEEE Transactions on Information Forensics and Security* 17 (2022), 1639–1654.

[41] Abbass Madi, Oana Stan, Aurélien Mayoue, Arnaud Grivet-Sébert, Cédric Gouy-Pailler, and Renaud Sirdey. 2021. A secure federated learning framework using homomorphic encryption and verifiable computing. In *2021 Reconciling Data Analytics, Automation, Privacy, and Security: A Big Data Challenge (RDAAPS)*. IEEE, 1–8.

[42] Itzik Malkiel and Lior Wolf. 2020. Mtadam: Automatic balancing of multiple training loss terms. *arXiv preprint arXiv:2006.14683* (2020).

[43] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2016. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *AISTATS*.

[44] Shike Mei and Xiaojin Zhu. 2015. Using Machine Teaching to Identify Optimal Training-Set Attacks on Machine Learners. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence* (Austin, Texas) *(AAAI'15)*. AAAI Press, 2871–2877.

[45] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. 2019. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE symposium on security and privacy (SP)*. IEEE, 691–706.

[46] Wenhao Mou, Chunlei Fu, Yan Lei, and Chunqiang Hu. 2021. A verifiable federated learning scheme based on secure multi-party computation. In *Wireless Algorithms, Systems, and Applications: 16th International Conference, WASA 2021, Nanjing, China, June 25–27, 2021, Proceedings, Part II*. Springer, 198–209.

[47] Milad Nasr, Reza Shokri, and Amir Houmansadr. 2019. 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)*. IEEE, 739–753.

[48] Thien Duc Nguyen, Phillip Rieger, Huili Chen, Hossein Yalame, Helen Möllering, Hossein Fereidooni, Samuel Marchal, Markus Miettinen, Azalia Mirhoseini, Shaza Zeitouni, Farinaz Koushanfar, Ahmad-Reza Sadeghi, and Thomas Schneider. 2022. FLAME: Taming Backdoors in Federated Learning. In *31st USENIX Security Symposium, USENIX Security 2022, Boston, MA, USA, August 10-12, 2022*, Kevin R. B. Butler and Kurt Thomas (Eds.). USENIX Association, 1415–1432. https://www.usenix.org/conference/usenixsecurity22/presentation/nguyen

[49] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael P Wellman. 2018. Sok: Security and privacy in machine learning. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 399–414.

[50] Dario Pasquini, Danilo Francati, and Giuseppe Ateniese. 2022. Eluding Secure Aggregation in Federated Learning via Model Inconsistency. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security* (Los Angeles, CA, USA) *(CCS '22)*. Association for Computing Machinery, New York, NY, USA, 2429–2443. https://doi.org/10.1145/3548606.3560557

[51] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019).

[52] Phillip Rieger, Thien Duc Nguyen, Markus Miettinen, and Ahmad-Reza Sadeghi. 2022. DeepSight: Mitigating Backdoor Attacks in Federated Learning Through Deep Model Inspection. In *29th Annual Network and Distributed System Security Symposium, NDSS 2022, San Diego, California, USA, April 24-28, 2022*. The Internet Society. https://www.ndss-symposium.org/ndss-paper/auto-draft-205/

[53] Benjamin IP Rubinstein, Blaine Nelson, Ling Huang, Anthony D Joseph, Shing-hon Lau, Satish Rao, Nina Taft, and JD Tygar. 2009. Stealthy poisoning attacks on PCA-based anomaly detectors. *ACM SIGMETRICS Performance Evaluation Review* 37, 2 (2009), 73–74.

[54] EK Sannara, Francois Portet, Philippe Lalanda, and VEGA German. 2021. A federated learning aggregation algorithm for pervasive computing: Evaluation and comparison. In *2021 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. IEEE, 1–10.

[55] Justin Schuh. 2019. *Potential uses for the Privacy Sandbox*. Retrieved January 19, 2023 from https://blog.chromium.org/2019/08/potential-uses-for-privacy-sandbox.html

[56] Shiqi Shen, Shruti Tople, and Prateek Saxena. 2016. Auror: Defending against poisoning attacks in collaborative deep learning systems. In *Proceedings of the 32nd Annual Conference on Computer Security Applications*. 508–519.

[57] Reza Shokri and Vitaly Shmatikov. 2015. Privacy-Preserving Deep Learning. In *CCS*.

[58] Vale Tolpegin, Stacey Truex, Mehmet Emre Gursoy, and Ling Liu. 2020. Data poisoning attacks against federated learning systems. In *European Symposium on Research in Computer Security*. Springer, 480–501.

[59] Jean-Baptiste Truong, Pratyush Maini, Robert J. Walls, and Nicolas Papernot. 2021. Data-Free Model Extraction. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*. Computer Vision Foundation / IEEE, 4771–4780. https://doi.org/10.1109/CVPR46437.2021.00474

[60] The European Union's. 2019. *The MELLODDY project*. Retrieved January 19, 2023 from https://www.melloddy.eu/

[61] Aidmar Wainakh, Fabrizio Ventola, Till Müßig, Jens Keim, Carlos Garcia Cordero, Ephraim Zimmer, Tim Grube, Kristian Kersting, and Max Mühlhäuser. 2022. User-Level Label Leakage from Gradients in Federated Learning. *Proceedings on Privacy Enhancing Technologies* 2022, 2 (2022), 227–244.

[62] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. 2019. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 707–723.

[63] Hongyi Wang, Kartik Sreenivasan, Shashank Rajput, Harit Vishwakarma, Saurabh Agarwal, Jy-yong Sohn, Kangwook Lee, and Dimitris S. Papailiopoulos. 2020. Attack of the Tails: Yes, You Really Can Backdoor Federated Learning. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.). https://proceedings.neurips.cc/paper/2020/hash/b8ffa41d4e492f0fad2f13e29e1762eb-Abstract.html

[64] Yuxin Wen, Jonas A. Geiping, Liam Fowl, Micah Goldblum, and Tom Goldstein. 2022. Fishing for User Data in Large-Batch Federated Learning via Gradient Magnification. In *Proceedings of the 39th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 162)*, Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (Eds.). PMLR, 23668–23684. https://proceedings.mlr.press/v162/wen22a.html

[65] Huang Xiao, Battista Biggio, Gavin Brown, Giorgio Fumera, Claudia Eckert, and Fabio Roli. 2015. Is feature selection secure against training data poisoning?. In *International Conference on Machine Learning*. PMLR, 1689–1698.

[66] Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *CoRR* abs/1708.07747 (2017). arXiv:1708.07747

[67] Chulin Xie, Keli Huang, Pin-Yu Chen, and Bo Li. 2020. DBA: Distributed Backdoor Attacks against Federated Learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net. https://openreview.net/forum?id=rkgyS0VFvr

[68] Chulin Xie, Keli Huang, Pin-Yu Chen, and Bo Li. 2020. Dba: Distributed backdoor attacks against federated learning. In *International conference on learning representations*.

[69] Cong Xie, Oluwasanmi Koyejo, and Indranil Gupta. 2018. Generalized byzantine-tolerant sgd. *arXiv preprint arXiv:1802.10116* (2018).

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

[71] Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. 2018. Byzantine-Robust Distributed Learning: Towards Optimal Statistical Rates. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer Dy and Andreas Krause (Eds.). PMLR, 5650–5659. http://proceedings.mlr.press/v80/yin18a.html

[72] Mikhail Yurochkin, Mayank Agarwal, Soumya Ghosh, Kristjan Greenewald, Nghia Hoang, and Yasaman Khazaeni. 2019. Bayesian nonparametric federated learning of neural networks. In *International conference on machine learning*. PMLR, 7252–7261.

[73] Xianglong Zhang, Anmin Fu, Huaqun Wang, Chunyi Zhou, and Zhenzhu Chen. 2020. A privacy-preserving and verifiable federated learning scheme. In *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 1–6.

[74] Yanci Zhang and Han Yu. 2022. Towards Verifiable Federated Learning. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, Lud De Raedt (Ed.). International Joint Conferences on Artificial Intelligence Organization, 5686–5693. https://doi.org/10.24963/ijcai.2022/792 Survey Track.

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

[76] Ligeng Zhu, Zhijian Liu, and Song Han. 2019. Deep leakage from gradients. *Advances in Neural Information Processing Systems* 32 (2019).

A ANALYSIS

In the following, we provide a theoretical justification of PROLIN. Without loss of generality, suppose that $t = 1$. Somewhat abusing the notation, let τ_i denote the probability that client i has property P. The maximum likelihood estimation of τ given the observed gradient aggregates $\mathbf{B} = (\mathbf{b}_1, \dots, \mathbf{b}_n)$ is

$$\tau_{\max} = \operatorname*{argmax}_{\tau} p(\tau|\mathbf{B})$$
$$\approx \operatorname*{argmax}_{\tau} p(\tau|\mathbf{G}) \tag{11}$$

where $p(\cdot|\cdot)$ denotes a generic conditional PDF and $\mathbf{G} = (g_1(\mathbf{b}_1), \dots, g_n(\mathbf{b}_n))$ are the feature aggregates. The last approximation holds if the features extracted by g_r are sufficient to predict the property information, that is, the detector model $M_r = h_r \circ g_r$ is accurate.

Then,

$$\tau_{\max} \approx \operatorname*{argmax}_{\tau} p(\tau|\mathbf{G})$$
$$= \operatorname*{argmax}_{\tau} p(\mathbf{G}|\tau)p(\tau) \qquad \text{(by Bayes' theorem)}$$
$$= \operatorname*{argmax}_{\tau} p(\tau) \int_{\mathbf{X} \in \mathbb{R}^{n \times N}} \frac{p(\tau, \mathbf{G}, \mathbf{X})}{p(\tau)} d\mathbf{X}$$
$$= \operatorname*{argmax}_{\tau} p(\tau) \int_{\mathbf{X} \in \mathbb{X}} \frac{p(\tau, \mathbf{X})}{p(\tau)} d\mathbf{X}$$
$$= \operatorname*{argmax}_{\tau} p(\tau) \int_{\mathbf{X} \in \mathbb{X}} p(\mathbf{X}|\tau) d\mathbf{X} \tag{12}$$

Figure 8: Detection of membership inference attacks (MIA) and gradient ascent attacks (GAA) on CIFAR-10 datasets by varying the total number of clients N while keeping the number of clients with the property P to 5.

where $\mathbb{X} = \{X \mid X \in \mathbb{R}^{n \times N} \land \forall r : A_{r,i}X_{r,i} = G_r\}$ denotes the set of all individual feature vectors whose per-round aggregates are exactly G.

Assuming a uniform prior on τ, we are searching for τ that maximizes $\int_{X \in \mathbb{X}} p(X|\tau)dX$. The likelihood function of τ given X is $L(\tau|X) = p(X|\tau)$, therefore

$$\operatorname*{argmax}_{\tau} \int_{X \in \mathbb{X}} p(X|\tau)dX = \operatorname*{argmax}_{\tau} \log \int_{X \in \mathbb{X}} L(\tau|X)dX \quad (13)$$

Let $\mathcal{Y}_i$ be a Bernoulli random variable, where $\mathcal{Y}_i = 1$ if client i has property P. The likelihood function is

$$
\begin{aligned}
L(\tau|X) &= p(X|\tau) \\
&= \prod_i p(X|\tau_i) \qquad \text{(by independence of clients)} \\
&= \prod_i \big(p(X|\mathcal{Y}_i = 1, \tau_i)p(\mathcal{Y}_i = 1|\tau_i) \\
&\qquad\quad + p(X|\mathcal{Y}_i = 0, \tau_i)p(\mathcal{Y}_i = 0|\tau_i) \big) \\
&= \prod_i \big(\tau_i \cdot p(X|\mathcal{Y}_i = 1, \tau_i) + (1 - \tau_i)p(X|\mathcal{Y}_i = 0, \tau_i) \big) \quad (14)
\end{aligned}
$$

Plugging Eq. (14) into Eq. (13), we obtain:

$$
\begin{aligned}
\tau_{\max} = \operatorname*{argmax}_{\tau} \log \int_{X \in \mathbb{X}} \prod_i \tau_i \cdot p(X|\mathcal{Y}_i = 1, \tau_i) + \\
+ (1 - \tau_i)p(X|\mathcal{Y}_i = 0, \tau_i) \; dX \quad (15)
\end{aligned}
$$

The exact computation of Eq. (15) is usually infeasible in practice because $p(X|\mathcal{Y}_i, \tau_i)$ can be specific to client i, however, the server may not have any client-specific prior. The best strategy for the server is to approximate $P(X|\mathcal{Y}_i, \tau_i)$ with the feature distributions

derived from its auxiliary dataset D^{aux}. Specifically,

$$
\begin{aligned}
p(X|\mathcal{Y}_i = 1, \tau_i) &= \prod_{r \in R(i)} p(X_{r,i}|X_{r-1,i}, \ldots, X_{1,i}, \mathcal{Y}_i = 1, \tau_i) \\
&\approx \prod_{r \in R(i)} p(X_{r,i}|\mathcal{Y}_i = 1, \tau_i) \\
&\approx \prod_{r \in R(i)} f_r^+(X_{r,i}) \quad (16) \\
p(X|\mathcal{Y}_i = 0, \tau_i) &= \prod_{r \in R(i)} p(X_{r,i}|X_{r-1,i}, \ldots, X_{1,i}, \mathcal{Y}_i = 0, \tau_i) \\
&\approx \prod_{r \in R(i)} p(X_{r,i}|\mathcal{Y}_i = 0, \tau_i) \\
&\approx \prod_{r \in R(i)} f_r^-(X_{r,i}) \quad (17)
\end{aligned}
$$

for a given client i. Here, we assumed that the linear features $X_{1,i}, \ldots, X_{n,i}$ of the same client i are independent, which is not true; even if the linear map g_r is different per round, its values are expected to concentrate around $\hat{g}_i$ for a given i as defined by Eq. (9) and approximated by linear regression in Eq. (10). This explains Constraint 2 in PROLIN.

The integral in Eq. (15) can be approximated with Monte Carlo integration, which takes several samples from $\mathbb{X}$. These samples should have a large likelihood $L(\tau|X)$ since such values are more significant to the integral. Hence, they should be re-sampled if τ changes, which can make the whole optimization process very slow. To avoid this large overhead of re-sampling, we use the best single sample estimate that maximizes the likelihood. This is a

lower bound of the likelihood because

$$\log \int_{\mathbf{X} \in \mathbb{X}} L(\boldsymbol{\tau}|\mathbf{X}) d\mathbf{X} \geq \log \max_{\mathbf{X} \in \mathbb{X}} L(\boldsymbol{\tau}|\mathbf{X}) \qquad (18)$$

Specifically,

$$\operatorname*{argmax}_{\boldsymbol{\tau}} \int_{\mathbf{X} \in \mathbb{X}} P(\mathbf{X}|\boldsymbol{\tau}) d\mathbf{X} \approx \operatorname*{argmax}_{\boldsymbol{\tau}} \max_{\mathbf{X} \in \mathbb{X}} \log L(\boldsymbol{\tau}|\mathbf{X}) \qquad (19)$$

Therefore, combining Eq. (19) (16) (17) with Eq. (15), we get:

$$\boldsymbol{\tau}_{\max} \approx \operatorname*{argmax}_{\boldsymbol{\tau}} \log \int_{\mathbf{X} \in \mathbb{X}} L(\boldsymbol{\tau}|\mathbf{X}) d\mathbf{X} \approx$$

$$\approx \operatorname*{argmax}_{\boldsymbol{\tau}} \max_{\mathbf{X} \in \mathbb{X}} \sum_i \log \left(\tau_i \cdot \prod_r f_r^+(\mathbf{X}_{r,i}) \right.$$

$$\left. + (1 - \tau_i) \cdot \prod_r f_r^-(\mathbf{X}_{r,i}) \right)$$

PROLIN performs exactly this optimization with the constraints that $\mathbf{X} \in \mathbb{X}$ (Constraint 1), $\mathbf{X}_{1,i}, \ldots, \mathbf{X}_{n,i}$ concentrate around the solution $\hat{\mathbf{G}}_i$ of Eq. (10) for each client i (Constraint 2), and $\tau_i \in \{0, 1\}$ as a client either has or does not have property P (Constraint 3).

The approximations in Eq. (11) and (19) introduce bias into PROLIN, but they also decrease the variance of its prediction compared to gradient reconstruction. If the gradient size z is much larger than the feature size t, and g_r effectively extracts all property relevant information, then the variance reduction can outbalance the bias, hence PROLIN can overcome gradient reconstruction. However, if M_r is inaccurate (i.e., g_r cannot capture property relevant information), or D^{aux} is not representative, then the bias can outbalance the variance and gradient reconstruction becomes better.

B COMPARISON OF GRADIENT AND FEATURE RECONSTRUCTION

We compare the reconstruction error of OLS in the gradient space with its error in the feature space. We show that if property information is scattered across several coordinates of the update vector, then OLS in the feature space has a smaller error than in the gradient space.

Following from Eq. (5), the linear model is

$$\mathbf{B} = \mathbf{A}\hat{\mathbf{W}} + \Omega$$

for gradient aggregation and

$$\mathbf{G} = \mathbf{A}\hat{\mathbf{G}} + \Theta$$

for feature reconstruction, where $\Omega_{i,j} \in \mathbb{R}^{n \times z}$ and $\Theta_{i,j} \in \mathbb{R}^{n \times t}$ are random values that are assumed to have identical normal distributions with variance σ just for the sake of comparison. Assume that OLS is used to obtain an approximation $\widetilde{\mathbf{W}}$ of $\hat{\mathbf{W}}$ and also an approximation $\widetilde{\mathbf{G}}$ of $\hat{\mathbf{G}}$, which means that $\widetilde{\mathbf{W}} = \hat{\mathbf{W}} + \mathbf{A}^+\Omega$ and $\widetilde{\mathbf{G}} = \hat{\mathbf{G}} + \mathbf{G}^+\Theta$ if $\mathbf{A}$ has full rank (recall that $\mathbf{A}^+ \in \mathbb{R}^{N \times n}$ is the pseudo-inverse of $\mathbf{A}$). Therefore,

$$E||g(\hat{\mathbf{W}}) - g(\widetilde{\mathbf{W}})||_1 = E||g(\hat{\mathbf{W}}) - g(\hat{\mathbf{W}} + \mathbf{A}^+\Omega)||_1$$
$$= E||g(\mathbf{A}^+\Omega)||_1 \qquad \text{(by linearity of } g\text{)}$$
$$= O(||\boldsymbol{\alpha}||_2 N \sqrt{n}\sigma)$$

and

$$E||\hat{\mathbf{G}} - \widetilde{\mathbf{G}}||_1 = E||\hat{\mathbf{G}} - (\hat{\mathbf{G}} + \mathbf{A}^+\Theta)||_1$$
$$= E||\mathbf{A}^+\Theta||_1$$
$$= O(tN\sqrt{n}\sigma)$$

where $g(\mathbf{x}) = \boldsymbol{\alpha}\mathbf{x}$ is fixed over the rounds and $(\mathbf{A}_{i,j}^+)^2 = O(1)$[7].

Therefore, the error when gradients are reconstructed is larger with a factor of $O(||\boldsymbol{\alpha}||_2/t)$. Since $\boldsymbol{\alpha}_i$ represents the impact of gradient coordinate i on property inference, gradient- and feature-based reconstructions can have comparable performance if $\boldsymbol{\alpha}$ is sparse (i.e., when the property information is already encoded by only a few gradient coordinate values whose reconstructions are sufficient for successful inference).

C MORE DETAILS

All settings are summarized in Table 2.

Notation	MNIST	Fashion-MNIST	CIFAR-10
n	300	300	300
z	21,840	21,840	62,006
N	50	50	50
λ	5	5	5
C	0.2	0.2	0.2
ϕ	0.1	0.1	0.1
η	0.01	0.01	0.1
$|D^{aux}|$	6,000	6,000	5,000
$|D'|$	$2|D^{aux}|$	$2|D^{aux}|$	$2|D^{aux}|$

Table 2: Values of Hyperparameters

---

[7]Each element of $\mathbf{A}^+\Omega \in \mathbb{R}^{N \times z}$ is a normal random variable with mean 0 and variance $\sum_j (\mathbf{A}_{i,j}^+)^2 \sigma^2$, hence $[g(\mathbf{A}^+\Omega)]_i$ is also a normal random variable with mean 0 and variance $\sum_{k=1}^z \boldsymbol{\alpha}_k^2 \sum_{j=1}^n (\mathbf{A}_{i,j}^+)^2 \sigma^2$.