CRONUS: ROBUST AND HETEROGENEOUS COLLABORATIVE LEARNING WITH BLACK-BOX KNOWLEDGE TRANSFER

Anonymous authors
Paper under double-blind review

ABSTRACT

Collaborative (federated) learning enables multiple parties to train a global model without sharing their private data, notably through repeated sharing of the parameters of their local models. Despite its advantages, this approach has many known security and privacy weaknesses, and is limited to models with the same architectures. We argue that the core reason for such security and privacy issues is the naive exchange of high-dimensional model parameters in federated learning algorithms. This increases the malleability of the trained global model to poisoning attacks and exposes the sensitive local datasets of parties to inference attacks. We propose Cronus, a robust collaborative learning framework that supports heterogeneous model architectures. The simple yet effective idea behind designing Cronus is to significantly reduce the dimensions of the exchanged information between parties. This allows us to impose a very tight bound over the error of the aggregation algorithm in presence of adversarial updates from malicious parties. We implement this through a robust knowledge transfer protocol between the local models. We evaluate prior federated learning algorithms against poisoning attacks, and we show that Cronus is the only secure method that withstands the parameter poisoning attacks. Furthermore, treating local models as black-boxes significantly reduces the information leakage about their sensitive training data. We show this using membership inference attacks.

1 INTRODUCTION

Collaborative machine learning has recently emerged as a promising approach for building machine learning models using distributed training data held by multiple parties. The training is distributed, and participants repeatedly exchange information about their local models, through an aggregation server. The objective is to enable all the participants to converge to a global model, while keeping their data private. This is very attractive to parties who own sensitive data, and agree on performing a common machine learning task, yet are unwilling to pool their data together for centralized training. Various applications can substantially benefit from collaborative learning. Examples include medical and financial applications, intelligent virtual assistants, speech recognition, keyboard input prediction, and mobile vision (Brendan et al., 2018; McMahan et al., 2017).

A popular approach for collaborative deep learning, known as *federated learning*, assumes *homogeneous* local models, i.e., with the same architecture (Shokri & Shmatikov, 2015; McMahan et al., 2017). Every participant shares its model parameters (or gradients) with a parameter server, after each round of training on its local data. The server aggregates the parameter vectors by computing their element-wise mean and shares the aggregate with the participants. Each party updates its local model with the latest global aggregate, and continues with the next round of local training.

There are major obstacles hindering the scalable deployment of secure and truly privacy-preserving federated learning for sensitive applications. Existing federated learning algorithms are not robust to adversarial updates (Bhagoji et al., 2019; Blanchard et al., 2017) and backdoor attacks (Bhagoji et al., 2019; Bagdasaryan et al., 2020), and can leak a significant amount of sensitive information about local datasets (Nasr et al., 2019; Melis et al., 2019). Besides, federated learning cannot be used for

aggregating heterogeneous models, for participants that use different models as they have different memory and computing power.

In this paper, our main focus is on securing collaborative machine learning against poisoning attacks. There exist a long chain of recent poisoning attacks and defenses for federated learning (Mhamdi et al., 2018; Xie et al., 2018; Yin et al., 2018; Blanchard et al., 2017; Wagner, 2004; Li et al., 2014; Nasr et al., 2019; Bhagoji et al., 2019; Bagdasaryan et al., 2020). Most existing defenses focus only on replacing the vulnerable aggregation in federated learning with a robust mean function, leaving the rest of the framework intact. However, those defenses are susceptible to some form of poisoning attack (Shejwalkar et al., 2021). The core reason for the susceptibility is the high-dimensionality of the parties' model updates: the theoretical error bound of the existing robust aggregation algorithms (Blanchard et al., 2017; Mhamdi et al., 2018; Diakonikolas et al., 2016; 2017; 2019; Li, 2018) depend on the dimensionality of their inputs, i.e., the size of model in federated learning. The models in modern federated learning generally have very high dimensionality, which makes the error bounds of robust aggregation algorithm prohibitively high. Although, in the literature of robust statistics, there are robust mean estimation algorithms that achieve the dimension-independent error bounds (Diakonikolas et al., 2016; 2017; Li, 2018), the sample complexity [1] of these algorithms is dependent on the dimensionality of the inputs. The high dimensionality of the models makes it impractical to directly use these robust mean estimation algorithms in federated learning.

Furthermore, by sharing the model parameters, federated learning completely opens the local models to the (potentially malicious) server and untrusted participants. This is a serious privacy vulnerability as the model parameters leak all the information that the model has about its training data (Nasr et al., 2019; Melis et al., 2019). The problem is significantly worse for large models, with a huge number of parameters. This high dimensionality not only magnifies the problem, but also makes it harder to design a defense mechanism with tight (robustness and privacy) guarantees. We refer to this as the *curse of dimensionality* for privacy and security in federated learning.

Thus, sharing model parameters to transfer the knowledge of local models is not a secure design choice in federated learning, especially considering that this is not the only way of exchanging knowledge between models.

Our contributions. In this paper, we design *Cronus*, a practical, robust collaborative learning approach to address the security issues of federated learning. Instead of sharing parameters with the server and updating them by overwriting them with the aggregated parameters, we extract, aggregate, and transfer the knowledge of models in a black-box fashion using the knowledge transfer paradigm (Geoffrey & amd Dean Jeff, 2014; Ba & Caruana, 2014). Knowledge transfer algorithms are commonly used for various compression and regularization purposes in machine learning (Geoffrey & amd Dean Jeff, 2014; Ba & Caruana, 2014; Wang et al., 2018; Anil et al., 2018), or as part of a low-sensitivity algorithm to train a model in a centralized setting with differential privacy (i.e., PATE frameworks) (Papernot et al., 2017; 2018).

Cronus supports *heterogeneous* model architectures, because the parties share knowledge of their local models via knowledge transfer, and therefore, they are only required to agree on the same ML task. Following the previous works that use knowledge transfer in collaborative ML (Guha et al., 2019; Lin et al., 2020), we assume that an unlabeled public dataset is available for the knowledge transfer. After a few rounds of local training on their private data, the Cronus parties share their predictions on the public data.

Local models are *fine-tuned* using the aggregated predictions (instead of directly overwriting their parameters), which significantly reduces the potential of harmful updates on local models. Besides, by sharing the prediction on the public data, Cronus reduces the number of dimensions of the update vectors to the size of the model's output, which is orders of magnitude smaller than the size of the model's parameters. Cronus uses the state-of-the-art robust mean estimation algorithm (Diakonikolas et al., 2017), which has a significantly small sample complexity for low dimensional updates. Benefiting from the low sample complexity, Cronus enjoys the tight **robustness** guarantee of robust mean estimation algorithms (Diakonikolas et al., 2017) on the aggregated updates even for small networks with a dozen participants (e.g., a network of hospitals).

[1] Sample complexity, in the context of federated learning, is the number of parties required to achieve the theoretical error bound.

Table 1: **Theoretical error rates of various aggregation algorithms.** Error rate is the L_2 distance between the outputs of robust aggregation algorithms in benign setting (all benign clients) and under attack (some clients malicious). n is the number of clients, ϵ is the breaking point, d is the dimensionality of aggregation algorithms' input, and σ^2 is the variance of each of the dimensions.

Aggregation rule (AGR)	Breaking point	Statistical error rate	Computational cost
Mean (McMahan et al., 2017)	$1/n$	Unbounded	$O(nd)$
Median (Li, 2018)	$1/2$	$O(\sigma\epsilon\sqrt{d})$	$O(nd \log n)$
Krum (Blanchard et al., 2017)	$(n-2)/2n$	$O(\sigma n\sqrt{d})$	$O(n^2 d)$
Bulyan (Mhamdi et al., 2018)	$(n-3)/4n$	$O(\sigma\sqrt{d})$	$O(n^2 d)$
RobustFilter (Our work) (Diakonikolas et al., 2017)	$1/2$	$O(\sigma\sqrt{\epsilon})$	$O(d^3 + n)$

The distillation process, using disjoint data to the models' training set, is used as a regularization technique (Ba & Caruana, 2014), and can reduce information leakage about the training data (Shokri et al., 2017). More importantly, Cronus is compatible with existing privacy-preserving mechanisms. That is to say that the black-box nature of our algorithm allows for using privacy-preserving mechanisms by the parties that are tailored to specifically protect the **privacy** of local training data in the black-box setting (Nasr et al., 2018; Dwork & Feldman, 2018).

We comprehensively evaluate the security and privacy of Cronus. We use state-of-the-art attacks to compare security due to Cronus and existing robust federated learning algorithms; we also design new attacks to break a class of aggregation algorithms based on *multiplicative weights updates* (Arora et al., 2012; Li et al., 2014)). We show that, in parameter sharing based federated learning, one or more poisoning attacks reduce the accuracy of the global model to random guessing for all of the considered robust aggregation algorithms. However, **under any attack, the reduction in average of accuracies of party models in Cronus is negligible** and is less than 2%; for the strongest attack on Cronus (of all considered attacks), the reduction on average accuracy of party models is 1.6% for Purchase, 1.3% for SVHN, 1.5% for MNIST, and 2.1% for CIFAR10. We also empirically show that, the **parties in Cronus enjoy significantly higher privacy compared to existing FL algorithms**; we measure the privacy as the risk of active and passive membership inference attacks. Finally, we show that **Cronus is highly effective even when the parties have heterogeneous model architectures**.

2 Background

We consider the synchronous federated learning (McMahan et al., 2017; Shokri & Shmatikov, 2015). In each epoch, the server broadcasts the global model, then each party shares an update (model parameters or gradients), computed using the global model and her local training data, with the server, and finally, the server aggregates the updates of all parties using an aggregation rule and updates the global model.

Threat Models. We consider *untargeted* poisoning attacks (Baruch et al., 2019; Bhagoji et al., 2019; Mhamdi et al., 2018), which aim to destroy the accuracy of the global model on all the inputs at test time. To achieve this, the adversary sends malicious updates through the ϵn malicious parties that she controls. We assume that the adversary also has access to data with the same distribution as that of the local training data of the $(1-\epsilon)n$ *benign parties*. Therefore, the adversary can use them to craft effective malicious updates. Each aggregation rule has a breaking point with respect to ϵ, i.e., it cannot provide any robustness guarantee if ϵ is larger than the breaking point.

We also evaluate the risk of membership inference attacks (Shokri et al., 2017) by the server. The adversary, i.e., the server, aims to distinguish between the members and non-members of the private training data of collaborating parties via observing the party updates. We consider passive and active membership inference: in the passive inference, the server only observes and infers from the updates, while in the active inference, the server modifies the aggregated updates to facilitate membership inference (Nasr et al., 2019).

Aggregation Rules. The most basic, yet effective, aggregation rule is *the weighted average* that FedAvg (McMahan et al., 2017) uses, where the weight of the i^{th} party is proportional to the size

of her local training data. Weighted-average is widely used in non-adversarial settings (McMahan et al., 2017). However, it is not robust against even a single malicious party (Blanchard et al., 2017). Multiple robust aggregation rules are proposed in the literature to improve the resilience of federated learning to poisoning attacks (Mhamdi et al., 2018; Xie et al., 2018; Yin et al., 2018; Blanchard et al., 2017). In this paper, we extensively compare with existing robust aggregation rules, including *element-wise median* (Huber, 2011; Wagner, 2004; Yin et al., 2018), *Krum* (Blanchard et al., 2017) and *Bulyan* (Mhamdi et al., 2018). We defer the reader to Appendix A for further details about those aggregation rules.

Little is enough attack (LIE). Existing aggregation rules, e.g., Krum and Bulyan, assume that in order to mount an effective attack, the malicious updates should lie far from the benign updates. However, Baruch et al. (2019) challenge this assumption and propose an attack called *little is enough* (LIE). LIE crafts its malicious updates by adding small amounts of noises to each of the dimensions of a benign update; a benign update can be an average of the updates from benign clients. LIE mounts successful attacks on the state-of-the-art robust aggregation algorithms, including Bulyan, Krum, and Median aggregation algorithms. From Table 1, we note that the error rates of all of these robust aggregation algorithms depend on the dimensionality of their inputs, which is very large in federated learning due to the extremely large number of parameters of modern neural networks. Hence, we argue that this *curse of dimensionality makes existing robust aggregation algorithms more susceptible to untargeted poisoning.*

High-Dimensional Robust Mean Estimation. The robust aggregation rules in the federated learning try to recover the mean of the benign updates from the malicious updates. This is the main objective of robust mean estimation problem from robust statistics. Diakonikolas et al. (2017) propose a filter-based algorithm RobustFilter, described in Algorithm 6 in Appendix C. It achieves optimal error guarantee $(O(\sigma\sqrt{\epsilon}))$ and optimal sample complexity $(\Theta((d/\epsilon)\log d))$. From Table 1, it is clear that RobustFilter has tighter error guarantees than the existing robust aggregation rules, and therefore, is more robust against poisoning attacks. However, note that it is impractical to use this robust mean estimation in federated learning due to its dimension-dependent sample complexity $\Theta((d/\epsilon)\log d)$, where d is the size of the model. For example, for training a DenseNet model with $\epsilon = 0.1$, millions of parties are required. We address this impracticality issue in our work.

Multiplicative Weight Update (MWU). In this work, we also evaluate the aggregation rules based on multiplicative weight update (MWU) technique (Arora et al., 2012; Freund & Schapire, 1997; Plotkin et al., 1991; Garg & Koenemann, 2007; Li et al., 2014). These techniques assume that, compared to benign updates, malicious updates lie farther away from the average of all input updates. Hence, for an update, they assign weight that is inversely proportional to the distance of the update from the average of all updates. We evaluate two aggregation algorithms: MWU with averaging (MWUAvg) (Arora et al., 2012) and MWU with optimization (MWUOpt) (Li et al., 2014). In Section 5, we empirically show that MWU based robust aggregation algorithms are robust to existing poisoning attacks, but can be broken using our improved poisoning attack that we will describe in Section 3.

3 Poisoning attack against MWU-based aggregations (OFOM)

MWU-based aggregations are resilient to the existing poisoning attacks but are not robust. We show this by proposing an attack targeting these aggregations. We can see that MWU-based aggregations treat the updates near the empirical weighted mean as the benign updates. However, the weighted mean is not robust. Our attack exploits this fact. Specifically, the adversary crafts two malicious updates: The first update is arbitrarily far away from the mean of the benign updates. The second malicious update is set to the empirical mean of benign updates and the first malicious update. In this way, the MWU-based aggregations assign higher weights to the malicious parties with the second malicious update, as it is close to the average of all the updates. We emphasize that the second malicious update, by construction, is also far from the mean of benign updates. Hence, the model is jeopardized by the second malicious update. We defer the reader to Appendix B for further details.

4 Cronus Collaborative Learning

In the existing federated learning algorithms, the server repeatedly collects the parameters of the local models, aggregates them by computing their mean, and sends the aggregate parameter vector back to the parties. This way of sharing knowledge between the participants has the following shortcomings:

Robustness: As shown in Table 1, the upper bound on the error of existing aggregation algorithms (in the adversarial setting) in federated learning depends on the dimensionality of the model parameters. This makes the aggregated models highly error-prone for large models. Thus, they are very susceptible to poisoning attacks, as described in Section 2.

Privacy: Sharing the model parameters facilitates strong white-box inference attacks, as it opens up the model to the adversary (Melis et al., 2019; Nasr et al., 2019). The larger the models are, the more significant their leakage about their local training data is.

Heterogeneity: Parameter aggregation is restricted to homogeneous architectures, i.e., all parties need to have the same model architecture.

To remedy the above shortcomings, the parties would need to share the knowledge that they have learned from their training data in a succinct way. This is the main objective of *knowledge transfer* (Geoffrey & amd Dean Jeff, 2014; Ba & Caruana, 2014; Papernot et al., 2017; 2018; Wang et al., 2018; Anil et al., 2018). In Cronus, we utilize the *knowledge distillation* (Geoffrey & amd Dean Jeff, 2014), a knowledge transfer technique introduced as a means of compressing large models into smaller ones while retaining their accuracy. It efficiently transfers the represented function by a model (or an ensemble of models) to a student model (Geoffrey & amd Dean Jeff, 2014). It makes learning very effective for the student model by placing equal weight on the relationships learned by the teachers across all the classes and significantly improves the convergence of student models (as compared to training directly on hard-labeled data) (Geoffrey & amd Dean Jeff, 2014; Ba & Caruana, 2014; Anil et al., 2018). Furthermore, knowledge transfer is an effective regularization method (Geoffrey & amd Dean Jeff, 2014).

What makes this approach, in particular, suitable for collaborative learning, is three-fold. First, it significantly reduces the dimensions of the updates, from the size of the model parameters to its output size. This sets up the stage for using state-of-the-art robust mean estimation algorithm RobustFilter (Diakonikolas et al., 2017). Second, the knowledge transfer through the predictions on a dataset, that does not overlap with the model's training set, itself reduces the information leakage of a model about its training data (Shokri et al., 2017; Shejwalkar & Houmansadr, 2021). In addition, the black-box setting allows us to make use of existing privacy-preserving algorithms to make the knowledge transfer privacy-preserving (Nasr et al., 2018; Dwork & Feldman, 2018). Third, all the models agree on a particular learning task, so they are homogeneous on their output vectors. Yet, they can be of heterogeneous architectures or even different families of machine learning algorithms.

In this paper, we leverage knowledge transfer and propose the ***Cronus* collaborative learning** algorithm, where the server extracts the knowledge of local models and aggregates them in a robust manner. The parties update their models using the aggregated knowledge as well as their local data. Then, each party shares their knowledge through predictions of their local models on a public dataset. This repeats in every round of federated learning.

4.1 Overview of Cronus

To extract and exchange the knowledge of local models, Cronus uses a set of *unlabeled* public data, X_p, which is essentially a set of feature vectors. Algorithm 1 describes the Cronus collaborative learning algorithm. Cronus has two training phases: In the first phase, called the *initialization phase*, every party i updates its local model parameters θ_i on its local training data D_i for T_1 times without any collaboration. In the second phase, called the *collaboration phase*, the parties share the knowledge of their local models via their predictions on the public dataset, X_p. Specifically, each epoch of this phase includes:

- Each party computes soft labels for X_p, using its local model parameters θ_i to get prediction vectors Y_i and shares them with the server.
- The server aggregates the predictions (separately for each public data), i.e., computes $\bar{Y} = f_{\mathsf{Cronus}}(Y_i, .., Y_n)$, and sends $\bar{Y}$ to all parties; we use the robust mean estimation algorithm RobustFilter as f_{Cronus}, which we describe in Section 2.
- Each party updates its local model parameters θ_i using their private data $\mathbf{D}_i$ and the soft-labeled public data $(X_p, \bar{Y})$.

Algorithm 1 Cronus algorithm. Initialization phase does not involve collaboration. D_i and θ_i are local dataset and model parameters from i-th party. Y_i^t are predictions from i-th party on public dataset X_p in epoch t and $Y_i^t[k]$ is the prediction on k-th public data in D_p.

Initialization phase

Each party $i \in [n]$ updates parameters in parallel:
for $t \in [T_1]$ epochs **do** ▷ Training *without* collaboration
    Update $\theta_i \leftarrow \textsf{TRAIN}\ (\theta_i, D_i)$
end for
$Y_i^0 = \textsf{PREDICT}(\theta_i; X_p)$ ▷ Compute initial predictions on X_p
Send Y_i^0 to the server

Collaboration phase

$\bar{Y}^0 = f_{\textsf{Cronus}}(\{Y_{i\in[n]}^0\})$ ▷ Initial aggregation at the server
for $t \in [T_2]$ epochs **do**
    for $i \in [n]$ parties **do** ▷ Each party updates parameter in parallel
        $D_p = \{X_p, \bar{Y}^t\}$
        $\theta_i \leftarrow \textsf{TRAIN}\ (\theta_i, D_i \cup D_p)$ ▷ Update local model parameter θ_i
        $Y_i^t = \textsf{PREDICT}(\theta_i; X_p)$
        Send Y_i^t to the server
    end for
    $\bar{Y}^{t+1} = f_{\textsf{Cronus}}(\{Y_{i\in[n]}^t\})$ ▷ Aggregation at the server
end for

Recall that the size of the updates determines the error rate of the existing robust aggregation algorithms. Hence, just by reducing the dimensionality of the updates (from the size of the model to the size of the prediction vector), Cronus already reduces the error rate guaranteed by any aggregation algorithms, including that of Bulyan and Median. However, when the size of the prediction vector is larger, these aggregation algorithms might still be highly susceptible to untargeted poisoning. To address this issue, Cronus uses RobustFilter (Diakonikolas et al., 2017), which achieves the dimension independent error rate guarantees. Furthermore, its sample and computational complexities have the least dependence on the dimensionality of updates among existing robust mean estimation algorithms. Nevertheless, we note that, Cronus is compatible with any robust aggregation algorithms.

The sample complexity of RobustFilter is $\Theta(d \log d)$, which is the number of parties required to achieve the theoretical error bound. This dimension-dependent sample complexity makes RobustFilter (Diakonikolas et al., 2017) impractical to use in parameter sharing based federated learning. In contrast, the low dimensional updates in Cronus significantly reduce the number of parties required for RobustFilter to achieve desired error rate. For instance, Cronus reduces sample complexity by an order of 10^5 when training DenseNet on Cifar10 dataset. Therefore, unlike any existing federated learning, Cronus can enjoy the strong theoretical error guarantees of RobustFilter, even with a small number of parties in the network.

Unlike the parameter sharing based federated learning algorithms, *Cronus does not force a single global model* onto local models. Instead, each local model is updated separately by improving its accuracy and resilience to inference attacks. This further improves the utility of local models. Note that, such fine-tuning has significant fairness advantages (Li et al., 2021; Yu et al., 2020). Furthermore, Cronus completely eliminates the risk of white-box inference attacks, as no client releases their model's parameters. While training on their local sensitive data, parties can anticipate the privacy risks of information leakage through their predictions and TRAIN their models with, for example, a membership privacy mechanism (Nasr et al., 2018). Parties can also make use of prediction privacy mechanisms before sharing their predictions with the server (Dwork & Feldman, 2018).

In conclusion, by combining knowledge transfer and state-of-the-art robust mean estimation RobustFilter, Cronus is a robust and practical collaborative learning framework. We show that Cronus considerably reduces membership privacy risk compared with federated learning and is compatible with existing privacy-preserving mechanisms through comprehensive evaluations. Besides, Cronus supports heterogeneous models among parties.

Table 2: **Experimental setup:** The number of malicious clients used for robustness assessment of different AGRs based on their breaking points. The number of benign parties are shown on top of each column.

f_{AGG}	Breaking point	Number of malicious parties		
		SVHN	MNIST	Purchase/CIFAR10
		32 benign	28 benign	16 benign
Mean	$1/n$	1	1	1
Median	$1/2$	31	27	15
MWU	$1/2$	31	27	15
Krum	$(n-2)/2n$	29	25	13
Bulyan	$(n-3)/4n$	9	8	4
Cronus	$1/2$	31	27	15

Table 3: **Comparison of the robustness** of Cronus and of the parameter sharing based FL with various AGRs. Robustness is measured as the ratio of *worst accuracy* and *benign accuracy*: worst accuracy is the accuracy of the global model (for Cronus, this is the average accuracy of client models) under the strongest of all attacks (Section 5); shown in "Strongest attack" row. While the benign accuracy is the accuracy without any attack. Table 2 gives the numbers of benign and malicious clients. We highlight the best accuracy for each setting. Table 5 in Appendix D.2.1 shows all the results.

Dataset		Parameter sharing based FL with various aggregation rules (AGRs)						Cronus
		Mean	Median	MwuAvg	MwuOpt	Bulyan	Krum	
SVHN	Benign accuracy	**95.9**	94.8	93.9	94.4	94.5	89.6	91.1
	Worst accuracy	0.9	14.5	0.9	0.7	15.5	16.2	**89.8**
	Strongest attack	(OFOM)	(LIE)	(OFOM)	(OFOM)	(LIE)	(LIE)	(LF)
	Robustness	0.01	0.15	0.01	0.01	0.16	0.18	**0.99**
MNIST	Benign accuracy	96.7	96.5	**97.2**	97.4	96.9	93.3	95.2
	Worst accuracy	9.6	91.5	25.3	12.7	**94.1**	89.9	93.7
	Strongest attack	(PAF)	(PAF)	(OFOM)	(PAF)	(LIE)	(LF)	(LF)
	Robustness	0.09	0.95	0.26	0.13	0.97	0.96	**0.99**
Purchase	Benign accuracy	93.3	93.0	**93.6**	92.5	92.8	72.1	89.6
	Worst accuracy	1.1	12.5	1.8	1.1	81.8	49.6	**88.0**
	Strongest attack	(PAF)	(PAF)	(OFOM)	(OFOM)	(LIE)	(LIE)	(LF)
	Robustness	0.01	0.75	0.02	0.01	0.87	0.69	**0.98**
CIFAR10	Benign accuracy	88.4	**89.1**	86.2	87.6	89.0	84.5	80.1
	Worst accuracy	11.3	15.1	14.2	12.8	75.6	18.0	**78.0**
	Strongest attack	(PAF)	(PAF)	(OFOM)	(OFOM)	(LIE)	(LIE)	(LIE)
	Robustness	0.13	0.17	0.16	0.15	0.85	0.21	**0.97**

5 EXPERIMENTS

Experimental Setup. We use four datasets, Purchase Shokri et al. (2017), CIFAR10 Krizhevsky & Hinton (2009), SVHN Netzer et al. (2011), and MNIST LeCun et al. (1998), to evaluate efficacy of Cronus. We use PyTorch pyt (2019) for our evaluations. We defer the details of the dataset splits, model architectures, and hyper-parameters to Appendix D.1.

Baseline attacks. We evaluate federated learning and Cronus under LIE attack (Section 2) and our poisoning attacks (Section 3), as well as with *Label flip poisoning attack (Label flip)* and *Naive poisoning attack (PAF)*. In label flip poisoning attack, the adversary flips the labels of malicious clients' local training data in the same way and shares the updates computed on these poisoned datasets. In naive poisoning attack (PAF), the adversary crafts malicious updates that are arbitrarily far from the average of the benign updates.

5.1 EMPIRICAL RESULTS

We compare the classification accuracy of the models trained using Cronus, stand-alone, central, and FedAvg settings in the benign setting. In the stand-alone setting, each party trains its model only on its local data, without any collaboration. In centralized learning, a single model is trained

on the union of the participants' data. For stand-alone and Cronus settings, the accuracy of models for different parties is different; therefore, we report the average classification accuracy of all the party models. The accuracies of stand-alone, centralized, and Cronus settings for Purchase are 76.3%, 94.3%, and 89.6%, respectively, while for CIFAR10 they are 66.8%, 90.2%, and 80.1%, respectively. Cronus uses unlabeled data for knowledge transfer, which any party can use to improve its model's accuracy via semi-supervised learning (Tarvainen & Valpola, 2017; Sohn et al., 2020; Chen et al., 2020; Berthelot et al., 2019). However, a CIFAR10 model trained in a semi-supervised manner using Mean-Teacher method (Tarvainen & Valpola, 2017) with 2,500 labeled and 10,000 unlabeled data achieves 69.75% accuracy while Cronus achieves 80.1% accuracy. That is, local models can achieve better accuracy via Cronus compared to the state-of-the-art semi-supervised learning methods.

Robustness. We compare the robustness of federated learning, using different aggregation algorithms (Section 2), against an adversary who mounts the strong poisoning attacks. To assess the worst-case robustness, we evaluate algorithms against the largest fraction of malicious parties, which is smaller than the breaking point. The number of benign and malicious parties are given in Table 2. The robustness assessment results are shown in Table 3. Each column corresponds to one aggregation algorithm, and the *Worst accuracy* row shows the accuracy of the final models when the attack in the *Strongest attack* row is mounted. We also show the convergence of Cronus and federated learning in adversarial and benign settings in Figures 2 and 1 respectively in Appendix D.2. From those results, it is clear that the federated learning algorithm works well in the absence of malicious parties; however, **all of the existing aggregation schemes in federated learning are significantly vulnerable to at least one poisoning attack.** The weighted average aggregation, i.e., FedAvg (McMahan et al., 2017), is susceptible to all of the attacks: *The accuracy of FedAvg reduces close to random guess accuracy for all the datasets*. Median aggregation is susceptible to PAF attack because the attack shifts the final aggregate along all the dimensions by a small amount to remain undetected, yet it can considerably damage the utility of the aggregated model. Although Bulyan and Krum are robust aggregations, they are susceptible to the LIE attack. As explained in Section 2, LIE attack exploits the sensitivity of the parameters of neural networks to small perturbations. The attack completely jeopardizes the accuracy of Krum aggregation because the attack successfully forces Krum aggregation to select the malicious update as the aggregate in most of the epochs. Note that the attack is not effective against MNIST classification task due to its simplicity, which allows the corresponding models to withstand the small perturbations. MwuAvg and MwuOpt withstand all the attacks, but are susceptible to our OFOM attack, which we propose in Section 3. For all the datasets, the OFOM attack reduces the accuracy of the aggregations close to the random guess accuracy.

On the other hand, **the robustness of Cronus remains almost 1.0 as the classification accuracy of its models remains unaffected by any of the tested poisoning attacks**. For the strongest attack on Cronus, the maximum reduction in accuracy is 0.4% for Purchase, 1.3% for SVHN, 1.5% for MNIST, and 4.8% for CIFAR10 models. The reason for the high resilience of Cronus to the poisoning attacks is two-fold. First, as detailed in Section 4.1, reduced dimensionality of updates enables to use robust aggregation algorithm RobustFilter (Diakonikolas et al., 2017). It guarantees that the aggregate prediction is robust even when the dimensionality of the prediction vector is high. For example, Cronus is still robust on the Purchase dataset when the dimensionality of the prediction vector is 100. Second, benign models have a stronger agreement on their predictions than on their parameter values. Thus, the variance of the prediction vector is smaller than that of gradient, which makes the error guarantee for aggregation algorithm RobustFilter even smaller. Thus, Cronus enjoys the tight robustness grantees of RobustFilter providing a practical, robust collaborative learning protocol.

Privacy. We show that Cronus considerably reduces the privacy risk compared with federated learning by measuring the privacy risk using the state-of-the-art membership inference attack. We also demonstrate the compatibility of Cronus with existing privacy-preserving mechanisms (Shokri, 2015; Abadi et al., 2016). The key discoveries from our experimental results are: (1) The updates in FedAvg are highly susceptible to membership inference, unlike the updates in Cronus. For the Purchase dataset, attack accuracy against the individual and aggregated updates in FedAvg is 78.1% and 80.1%, respectively, whereas in Cronus they are 51.7% and 51.9%. (2) The active membership inference attacks Nasr et al. (2019) significantly increase the privacy risk of the target data in the case of FedAvg. For Purchase dataset, the risk due to individual updates increases by 7.8% (77.1% to 84.9%), while due to aggregated update increases by 8% (74.7% to 82.7%). (3) In the case of Cronus, the active membership inference attacks are ineffective, and the increase in privacy risk is negligible.

On Purchase dataset, the risk increases by 0.3% for individual updates while 1.1% for aggregated updates. See experimental results and discussion in Appendix D.2.2.

Cronus with heterogeneous model architectures. Due to the use of prediction based updates, Cronus allows parties with heterogeneous model architectures to participate in the collaboration. The key findings of our experimental evaluations are: (1) the heterogeneous collaboration between models of equivalent capacities does not reduce the accuracy of client models compared to its homogeneous counterparts, (2) the presence of a few bad models does not affect the accuracy of the good models in the heterogeneous collaboration, while significantly benefits the bad models, and (3) heterogeneity allows for more knowledge sharing via collaboration and always improves the utility of collaborations. Due to space restrictions, we defer the details of experiments and discussion to Appendix D.2.6.

6 Related work

Existing federated learning algorithms are susceptible to various poisoning attacks, which deter data holders from trusting the algorithm Bhagoji et al. (2019); Bagdasaryan et al. (2020); Melis et al. (2019). Various robust aggregation schemes have been proposed in the literature, as shown in Section 2. However, as discussed in Section 2, the high dimensional updates in federated learning significantly reduce the effectiveness of these algorithms. On the other hand, although federated learning prevents its parties from sharing their private data, recent works Nasr et al. (2019); Orekondy et al. (2018); Melis et al. (2019) demonstrate passive and active inference attacks against this setting and successfully infer sensitive information about the parties' private data. Such inference attacks can be defended by using differentially private (DP) learning Abadi et al. (2016); Brendan et al. (2018); Dwork & Feldman (2018). By trusting the aggregator, McMahan et al. Brendan et al. (2018) consider the differential privacy in the federated setting; however, their algorithm can achieve acceptable accuracy only with a large number of parties. Cronus is robust to existing poisoning attacks and reduces the privacy risk, which alleviates the robustness and privacy issue of federated learning.

Knowledge of ensemble of teacher models has been used to train a student model in a few previous works Jihun et al. (2016); Papernot et al. (2017; 2018); Anil et al. (2018); Jeong et al. (2018). Papernot et al. Papernot et al. (2017) propose PATE, a centralized learning approach that uses knowledge transfer to achieve differential privacy with high accuracy. PATE's setting is fundamentally different from that of Cronus in that all the teacher models and the noise generation mechanisms in PATE are performed by a trusted entity that owns all the data. The co-distillation approach is a method to use distillation on private training data with other parties Anil et al. (2018); Jeong et al. (2018), however, without defending against data poisoning and membership inference attacks Shokri et al. (2017); Hayes et al. (2019). Personalized federated learning has drawn a lot of attention recentlyFallah et al. (2020); Zhang et al. (2020); T Dinh et al. (2020). In Cronus, parties train their local model and collaborate with others in a black-box manner. In the end, parties have their own model. Thus, our Cronus can also be used as a way to achieve personalization.

7 Conclusions

We propose Cronus, a robust collaborative learning framework, to mitigate the three deficiencies of existing federated learning: susceptibility to poisoning attacks and membership inference attacks and the inability to support heterogeneous architectures. Our main idea is to let parties share predictions of their local models on an unlabeled public dataset instead of the model parameters. Together with the observation that the dimension of predictions is often smaller than that of the model parameters by orders of magnitude (e.g., DenseNet), Cronus sets up the stage for using the state-of-the-art robust aggregation algorithm, which incurs a prohibitively high sample complexity for high dimensional statistics. Benefiting from the low sample complexity, Cronus enjoys a tight robustness guarantee on the aggregated predictions even when a small number of parties participate in the training process.

Through comprehensive evaluations, we show that Cronus is the only robust framework against existing data poisoning attacks. We also empirically show that Cronus is less prone to membership inference attacks compared with existing federated learning frameworks. This is because Cronus uses prediction sharing (black-box) instead of parameter sharing (white-box) among participants. At last, Cronus also supports heterogeneous model architectures, which enables participants with different computation power to collaborate.

REFERENCES

Acquire Valued Shoppers Challenge. https://www.kaggle.com/c/acquire-valued-shoppers-challenge/data, 2019. [Online; accessed 11-September-2019].

PyTorch Documentation. https://pytorch.org/, 2019. [Online; accessed 11-September-2019].

Tensorflow Privacy. https://github.com/tensorflow/privacy/tree/master/privacy/analysis, 2019. [Online; accessed 23-September-2019].

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

Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. Qsgd: Communication-efficient sgd via gradient quantization and encoding. In *Advances in Neural Information Processing Systems*, pp. 1709–1720, 2017.

Rohan Anil, Gabriel Pereyra, Alexandre Passos, Robert Ormandi, George E Dahl, and Geoffrey E Hinton. Large scale distributed neural network training through online distillation. *arXiv preprint arXiv:1804.03235*, 2018.

Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: A meta-algorithm and it's applications. *Theory of Computing*, 2012.

Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In *Advances in neural information processing systems*, pp. 2654–2662, 2014.

Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 2938–2948. PMLR, 2020.

Moran Baruch, Baruch Gilad, and Yoav Goldberg. A little is enough: Circumventing defenses for distributed learning. In *Advances in Neural Information Processing Systems*, 2019.

David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin A Raffel. Mixmatch: A holistic approach to semi-supervised learning. In *Advances in Neural Information Processing Systems*, pp. 5050–5060, 2019.

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

Peva Blanchard, Rachid Guerraoui, Julien Stainer, et al. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Advances in Neural Information Processing Systems*, pp. 119–129, 2017.

McMahan H Brendan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models. *International Conference on Learning and Representation*, 2018.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020.

Ilias Diakonikolas, Gautam Kamath, Daniel M Kane, Jerry Li, Ankur Moitra, and Alistair Stewart. Robust estimators in high dimensions without the computational intractability. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 655–664. IEEE, 2016.

Ilias Diakonikolas, Gautam Kamath, Daniel M. Kane, Jerry Li, Ankur Moitra, and Alistair Stewart. Being robust (in high dimensions) can be practical. *In Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 2017.

Ilias Diakonikolas, Gautam Kamath, Daniel Kane, Jerry Li, Jacob Steinhardt, and Alistair Stewart. Sever: A robust meta-algorithm for stochastic optimization. In *International Conference on Machine Learning*, pp. 1596–1606, 2019.

Cynthia Dwork and Vitaly Feldman. Privacy-preserving prediction. In *Conference On Learning Theory*, pp. 1693–1702, 2018.

Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 3557–3568. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/24389bfe4fe2eba8bf9aa9203a44cdad-Paper.pdf.

Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences 55, no. 1*, 1997.

Naveen Garg and Jochen Koenemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. *SIAM Journal on Computing 37, no. 2*, 2007.

Hinton Geoffrey and Vinyals Oriol amd Dean Jeff. Distilling the knowledge in a neural network. *NIPS 2014 Deep Learning Workshop*, 2014.

Neel Guha, Ameet Talwalkar, and Virginia Smith. One-shot federated learning. *arXiv preprint arXiv:1902.11175*, 2019.

Jamie Hayes and Olya Ohrimenko. Contamination attacks and mitigations in multi-party machine learning. *32nd Conference on Neural Information Processing Systems (NIPS 2018)*, 2018.

Jamie Hayes, Luca Melis, George Danezis, and Emiliano De Cristofaro. Logan: Membership inference attacks against generative models. *Proceedings on Privacy Enhancing Technologies*, 2019(1):133–152, 2019.

Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017.

Peter J Huber. *Robust statistics*. Springer, 2011.

Matthew Jagielski, Aline Oprea, Battista Biggio, Chang Liu, Cristina Nita-Rotaru, and Bo Li. Manipulating machine learning: Poisoning attacks and countermeasures against regression learning. *39th IEEE Symposium on Security and Privacy*, 2018.

Eunjeong Jeong, Seungeun Oh, Hyesung Kim, Jihong Park, Mehdi Bennis, and Seong-Lyun Kim. Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data. *arXiv preprint arXiv:1811.11479*, 2018.

Hamm Jihun, Cao Yingjun, and Belkin Mikhail. Learning privately from multiparty data. *Proceedings of The 33rd International Conference on Machine Learning*, 2016.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Jerry Zheng Li. *Principled approaches to robust machine learning and beyond*. PhD thesis, Massachusetts Institute of Technology, 2018.

Qi Li, Yaliang Li, Jing Gao, Bo Zhao, Wei Fan, and Jiawei Han. Resolving conflicts in heterogeneous data by truth discovery and source reliability estimation. *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, 2014.

Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. Ditto: Fair and robust federated learning through personalization. In *International Conference on Machine Learning*, pp. 6357–6368. PMLR, 2021.

Tao Lin, Lingjing Kong, Sebastian U Stich, and Martin Jaggi. Ensemble distillation for robust model fusion in federated learning. *Advances in Neural Information Processing Systems*, 33, 2020.

H Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. *Proceedings of the 20 th International Conference on Artificial Intelligence and Statistics*, 2017.

Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. *40th IEEE Symposium on Security and Privacy*, 2019.

El Mahdi El Mhamdi, Rachid Guerraoui, and Sébastien Rouault. The hidden vulnerability of distributed learning in byzantium. In *International Conference on Machine Learning*, pp. 3518–3527, 2018.

Luis Muñoz-González, Battista Biggio, Ambra Demontis, Andrea Paudice, Vasin Wongrassamee, Emil C Lupu, and Fabio Roli. Towards poisoning of deep learning algorithms with back-gradient optimization. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pp. 27–38. ACM, 2017.

Milad Nasr, Reza Shokri, and Amir Houmansadr. Machine learning with membership privacy using adversarial tuning. *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018.

Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Stand-alone and federated learning under passive and active white-box inference attacks. *Security and Privacy (SP), 2019 IEEE Symposium on*, 2019.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

Tribhuvanesh Orekondy, Seong Joon Oh, Bernt Schiele, and Mario Fritz. Understanding and controlling user linkability in decentralized learning. *arXiv preprint arXiv:1805.05838*, 2018.

Nicolas Papernot, Martín Abadi, Ulfar Erlingsson, Ian Goodfellow, and Kunal Talwar. Semi-supervised knowledge transfer for deep learning from private training data. *International Conference on Learning and Representation*, 2017.

Nicolas Papernot, Shuang Song, Ulfar Erlingsson, Ilya Mironov, Ananth Raghunathan, and Talwar Kunal. Scalable private learning with PATE. *International Conference on Learning and Representation*, 2018.

Serge A. Plotkin, David B. Shmoys, and Eva Tardos. Fast approximation algorithms for fractional packing and covering problems. *Foundations of Computer Science*, 1991.

Virat Shejwalkar and Amir Houmansadr. Membership privacy for machine learning models through knowledge transfer. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2021.

Virat Shejwalkar, Amir Houmansadr, Peter Kairouz, and Daniel Ramage. Back to the drawing board: A critical evaluation of poisoning attacks on federated learning. *arXiv preprint arXiv:2108.10241*, 2021.

Reza Shokri. Privacy games: Optimal user-centric data obfuscation. *Proceedings on Privacy Enhancing Technologies*, 2015.

Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*. ACM, 2015.

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *Security and Privacy (SP), 2017 IEEE Symposium on*, 2017.

Kihyuk Sohn, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin A Raffel, Ekin Dogus Cubuk, Alexey Kurakin, and Chun-Liang Li. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *Advances in Neural Information Processing Systems*, 33, 2020.

Guocong Song and Wei Chai. Collaborative learning for deep neural networks. In *Advances in Neural Information Processing Systems*, pp. 1832–1841, 2018.

Jacob Steinhardt, Pang Wie Koh, and Percy Liang. Certified defenses for data poisoning attacks. *Advances in Neural Information Processing Systems*, 2017.

Lili Su. *Defending distributed systems against adversarial attacks: Consensus, consensus-based learning, and statistical learning*. PhD thesis, University of Illinois at Urbana-Champaign, 2017.

Canh T Dinh, Nguyen Tran, and Tuan Dung Nguyen. Personalized federated learning with moreau envelopes. *Advances in Neural Information Processing Systems*, 33, 2020.

Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in neural information processing systems*, pp. 1195–1204, 2017.

David Wagner. Resilient aggregation in sensor networks. *Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*, 2004.

Xiaojie Wang, Rui Zhang, Yu Sun, and Jianzhong Qi. Kdgan: knowledge distillation with generative adversarial networks. In *Advances in Neural Information Processing Systems*, pp. 775–786, 2018.

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

Dong Yin, Kannan Chen, Yudong Ramchandran, and Peter Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. *In Proceedings of the 35th International Conference on Machine Learning*, 2018.

Tao Yu, Eugene Bagdasaryan, and Vitaly Shmatikov. Salvaging federated learning by local adaptation. *arXiv preprint arXiv:2002.04758*, 2020.

Michael Zhang, Karan Sapra, Sanja Fidler, Serena Yeung, and Jose M Alvarez. Personalized federated learning with first order model optimization. In *International Conference on Learning Representations*, 2020.

Algorithm 2 Federated learning algorithm (McMahan et al., 2017; Shokri & Shmatikov, 2015)

Initialize global model θ_a^0
for $t \in [T]$ **do**
  for $i \in [n]$ **do** ▷ Party i's local update
    $\theta_i^t \leftarrow \theta_a^t - \nabla_\theta L(D_i; \theta_a^t)$
    Return θ_i^t to server
  end for
  $\theta_a^{t+1} = f_{\mathsf{AGG}}(\theta_{i\in[n]}^t)$ ▷ Aggregation of updates at server
  Return θ_a^{t+1} to all the parties
end for

A AGGREGATION ALGORITHMS

In this section, we describe the setting of federated learning and, various aggregation algorithms used to combine the updates of the collaborating parties.

A.1 FEDERATED LEARNING SETTING

Federated learning (McMahan et al., 2017; Shokri & Shmatikov, 2015) enables multiple data holders to train a global model without sharing their data, and through sharing of their training gradients/parameters (Blanchard et al., 2017; Alistarh et al., 2017; Mhamdi et al., 2018; Xie et al., 2018; Yin et al., 2018). For concreteness, below we describe Federated Average (FedAvg) algorithm (McMahan et al., 2017) and its setting, and use it in the rest of our work. In FedAvg, multiple parties collaborate over multiple epochs to learn a global machine learning model with a classification performance superior to the models learned individually. FedAvg assumes that there are n parties with their *local training datasets*, D_i's, and a *central server* which aggregates the party updates and broadcasts the aggregate to all of the parties. In the t^{th} epoch of FedAvg, parties train the aggregate θ_a^t, broadcast by the server at the end of t^{th} epoch, on their local training data, D_i. Parties use stochastic gradient descent for updating, i.e., $\theta_i^t = \theta_a^t - \nabla_\theta L(D_i; \theta_a^t)$, where $L(D; \theta)$ is loss of θ on data D. Each party then sends the parameters of the locally updated model, θ_i^t, to the server for aggregation. The central server collects all the θ_i^t updates and computes their aggregate $\theta_a^t = f_{\mathsf{AGG}}(\theta_{i\in[n]}^t)$. Specifically, FedAvg uses the weighted average as its f_{AGG}, where the weight of the i^{th} party is determined based on the size of her local training data, i.e., $w_i = \frac{|D_i|}{|D|}$; $|D|$ denotes size of dataset D. The weighted average is formally given by:

$$f_{\mathsf{Mean}}: \quad \theta_a^{t+1} = \sum_{i=1}^{n} \frac{|D_i|}{\sum_{j=1}^{n} |D_j|} \theta_i^t \tag{1}$$

The server then broadcasts the aggregate θ_a^t to all n parties. This process repeats for T epochs or until sufficient accuracy is achieved by the aggregated global model. The procedure is described in Algorithm 2. This algorithm is not robust against poisoning attack and even a single party can destroy the global model.

A.2 KRUM

Weighted average aggregation cannot tolerate even a single malicious party (Blanchard et al., 2017; Wagner, 2004). To solve this problem, Blanchard et al. (Blanchard et al., 2017) proposed Krum aggregation, which is based on geometric median of vectors (Su, 2017; Mhamdi et al., 2018). The intuition behind Krum is as follows: Krum assumes that party updates have a normal distribution and that the benign updates lie close to each other in the parameter space. Hence, instead of computing the average of the updates, Krum selects as aggregate the update that is closest to its $(1-\epsilon)n$ neighbor updates. The details of the aggregation are as follows. Let $\theta_1, ..., \theta_n$ be the updates received by the server. For $i \neq j$, $i \to j$ denotes that θ_j belongs to the $(1-\epsilon)n - 2$ updates closest to θ_i. Let $s(\theta_i) = \sum_{i \to j} ||\theta_i - \theta_j||^2$ be the *score* of θ_i. Then, Krum selects the θ_k with the lowest score. The

Krum aggregation algorithm is formalized in (2).

$$f_{\mathsf{Krum}}: \quad \theta_a^{t+1} = \underset{\theta_{i\in[n]}^t}{\operatorname{argmin}} \sum_{i \to j} ||\theta_i^t - \theta_j^t||^2 \tag{2}$$

A.3 BULYAN

The breaking point of Krum is $\epsilon = (\frac{n-2}{2n})$, i.e., it can tolerate up to $(\frac{n-2}{2})$ malicious parties, while maintaining high utility of the final model (Blanchard et al., 2017). However, Mhamdi et al. (2018) proposed an attack on Krum assuming an omniscient adversary who has access to all the benign updates. The attack exploits the fact that, in a vector space of dimension $d \gg 1$, small disagreements on each coordinate translate into a distance $\|\mathbf{x} - \mathbf{y}\|_p = \mathcal{O}(\sqrt[p]{d})$. Therefore, the adversary crafts a malicious update with a single dimension set to a large value, and the other dimensions set to the average of the benign updates. Such malicious update pushes the parameter vector to a sub–optimal parameter space and destroys the global model's accuracy. Essentially, Krum filters outliers based on the entire update vector, but does not filter coordinate-wise outliers. To address this, (Mhamdi et al., 2018) proposes a meta-aggregation rule Bulyan, which performs vector-wise, e.g. Krum, and coordinate-wise, e.g. TrimmedMean (Yin et al., 2018), filtering in two steps. At first, Bulyan uses some Byzantine resilient aggregation $\mathcal{A}$, e.g., Krum in Algorithm 3, to filter outliers based on the distances between the update vectors, and then aggregates these updates using a variant of TrimmedMean. Algorithm 3 describes the Bulyan aggregation.

Algorithm 3 Bulyan aggregation: f_{Bulyan} (Mhamdi et al., 2018)

1: **Input**: $\mathcal{A} = f_{\mathsf{Krum}}$, $\mathcal{P} = (\theta_1^t, ..., \theta_n^t)$, n, ϵ
2: $S \leftarrow \emptyset$
3: **while** $|S| < (1 - 2\epsilon)n$ **do**
4: $\quad p \leftarrow \mathcal{A}(\mathcal{P} \backslash S)$
5: $\quad S \leftarrow S \cup \{p\}$
6: **end while**
7: **Output**: $\theta_a^{t+1} = \mathsf{TrimmedMean}(S)$

Among the different variants of TrimmedMean (Xie et al., 2018; Mhamdi et al., 2018; Yin et al., 2018), we follow the one used in the original work (Mhamdi et al., 2018) given in (3). Here, U_j is defined as the set of indices of the top-$(1 - 2\epsilon)n$ values in $(\theta_1^j, ..., \theta_n^j)$ nearest to their *median* μ_j.

$$\mathsf{TrimmedMean}(\theta_1, ..., \theta_N) = \left\{ \theta_a^j = \frac{1}{|U_j|} \sum_{i \in U^j} \theta_i^j \ \forall j \in [d] \right\} \tag{3}$$

A.4 MULTIPLICATIVE WEIGHT UPDATE (MWU)

Multiplicative weight update (MWU) technique lies at the core of many learning algorithms (Arora et al., 2012; Freund & Schapire, 1997; Plotkin et al., 1991; Garg & Koenemann, 2007; Li et al., 2014). The general framework of MWU is given in Algorithm 4. The intuition behind MWU-based aggregations is to reduce the weights of malicious parties using the distance between their malicious updates and the aggregated update. This is based on two assumptions: malicious updates lie farther away from the mean compared with the benign updates, and the number of malicious updates is smaller than that of the benign updates. Therefore, MWU-based aggregation schemes have the breaking point of $\lfloor \frac{n-1}{2} \rfloor$ malicious parties.

There are different variants of MWU that use different functions for WeightUpdate and f_{AGG} in Algorithm 4. We detail two of them next.

A.4.1 MWU WITH MEAN AGGREGATION

In MWU, if the weighted mean is used as the aggregation algorithm, f_{AGG} in Algorithm 4, it is called MwuAvg. Here, the weight of the i^{th} party is updated based on the distance between the weighted average, θ_a^t, of all the updates and θ_i; this is given by (4). The weights of all the parties are equal at the beginning.

Algorithm 4 Multiplicative weights update: f_{MWU}

Input: $\mathcal{P} = (\theta_1^t, ..., \theta_n^t)$
Initialize parties' weight vector $\mathbf{w}^0 \leftarrow \mathbf{1}$ and $\theta_a^0 \leftarrow f_{\mathsf{AGG}}(\mathbf{w}^0, \mathcal{P})$ at $t = 0$
repeat
    $\mathbf{w}^{t+1} \leftarrow \mathsf{WeightUpdate}\ (\mathbf{w}^t, \theta_a^t, \mathcal{P})$
    $\theta_a^{t+1} \leftarrow f_{\mathsf{AGG}}(\mathbf{w}^{t+1}, \mathcal{P})$
    $t \leftarrow t + 1$
until Convergence criterion is satisfied
Output: final θ_a

$$w_i^{t+1} = w_i^t \exp(-||\theta_a^t - \theta_i||_p) \tag{4}$$

$$\theta_a^{t+1} = \frac{\sum_{i=1}^n w_i^{t+1}\theta_i}{\sum_{i=1}^n w_i^{t+1}} \tag{5}$$

A.4.2 MWU WITH OPTIMIZATION

Li et al. (Li et al., 2014) propose a truth discovery framework CRH, to aggregate the responses in a crowd-sourcing setting. In each epoch, the framework updates the weights of parties based on the solution of an optimization problem. Essentially, the weight update algorithm considers the distance of parties' updates from the aggregate of all the updates in each epoch. The weight update and aggregate computation are given by (6) and (7), respectively. For further details of the aggregation, please refer to (Li et al., 2014).

$$w_i^{t+1} = -\log\left(\frac{||\theta^t - \theta_i||_p}{\sum_{i=1}^n ||\theta^t - \theta_i||_p}\right) \tag{6}$$

$$\theta^{t+1} = \sum_{i=1}^{n} w_i^{t+1}\theta_i \tag{7}$$

B ATTACKS ON AGGREGATION ALGORITHMS

In this section, we detail the poisoning and membership inference attacks used in our work to evaluate the robustness and privacy in federated learning. The poisoning attacks are of two types: *availability* and *targeted* attacks. The earlier attacks aim to jeopardize the overall accuracy of the final model/s (Baruch et al., 2019; Hayes & Ohrimenko, 2018; Steinhardt et al., 2017), while the latter attacks aim to mis-classify only a specific set of samples of the attacker's choice at the test time (Bagdasaryan et al., 2020; Bhagoji et al., 2019). We focus on the poisoning availability attacks and below introduce such attacks from the literature, and also introduce a new poisoning attack targeting the MwuAvg and MwuOpt aggregations.

B.1 LABEL FLIP POISONING (LABEL FLIP)

We consider a type of data poisoning attacks (Jagielski et al., 2018; Muñoz-González et al., 2017; Hayes & Ohrimenko, 2018), where the adversary flips the labels of her local training data in a particular fashion to poison it. We call this attack *label flip poisoning* attack. The label flipping strategy is performed consistently across all of the ϵn malicious parties that the adversary controls, i.e., all the malicious parties flip labels in the exact same way. Then, the ϵn malicious parties (also called as Byzantine workers in the literature) use this poisoned data to train their local models and then share corresponding updates with the central server.

B.2 NAIVE POISONING (PAF)

Our threat model considers an omniscient adversary who knows the distribution of benign updates, i.e., the mean and standard deviation of each dimension of benign updates. The adversary can estimate

this distribution as she possesses data drawn from the distribution same as that of benign parties. Given this, the adversary crafts the malicious update θ_m to be arbitrarily far from the mean of the benign updates:

$$\theta_m = \frac{\sum_{i=1}^{(1-\epsilon)n} \theta_i}{(1-\epsilon)n} + \theta' \tag{8}$$

This malicious update, θ_m, is then shared by each malicious party with the central server. In (8), θ' is a vector of size $|\theta_i|$ with arbitrarily large (or small) coordinate values. This attack can jeopardize the weighted averaging, and interestingly, weighted median aggregations based federated learning.

B.3 Little is enough attack (LIE)

Baruch et al. (Baruch et al., 2019) propose an attack called *little is enough* (LIE). The attack successfully circumvents state-of-the-art robust aggregation algorithms, including Bulyan and Krum. These aggregations are vulnerable to the attack, because they are tailored to an adversary that crafts a malicious update with at least one arbitrarily large dimension. However, in practice, a malicious update, θ^m, obtained by small perturbations in a large number of dimensions of a benign update suffice to affect model's convergence and also circumvent the defenses. Therefore, note that, *the root cause of the success of the LIE attack is also the high dimensionality of the updates.* The attack is described in Algorithm. 5. The s in the Algorithm. 5 is the number of non-corrupted parties, whose deviation from mean is more than malicious parties, needed in order to bypass the detection. Assume the distributions of different parameters from all parties can be expressed by a normal distribution, z will set the range in which the adversary updates can deviate from the mean without being detected. $\bar{\mu}, \bar{\sigma}$ are the mean and variance of the distribution of the parameters from benign updates.

B.4 Our poisoning attack (OFOM)

In this section, we propose an attack which targets aggregation schemes that perform weighted aggregation of data by assigning the weights based on the distance of the data points from an aggregate of the data. These aggregations are robust to all the above attacks, as we show in our evaluation. We discussed two such aggregation schemes: MWU with averaging (Arora et al., 2012) and MWU with optimization (Li et al., 2014) in Section A.4. In any given epoch, the aforementioned aggregation schemes start with equal weights to all parties and update a party's weight based on the distance of the party's update from weighted average of all party updates. *The attack exploits the fact that, all parties are given equal weights to start with.* The OFOM attack craft two malicious updates: The first update, θ_1^m, is arbitrarily far away from the true mean, and is obtained by adding an arbitrarily large (or small) vector θ' to the mean of benign updates. The second malicious update, θ_2^m, is at the empirical mean of benign updates and θ_1^m. The malicious updates are formalized in (9).

$$\theta_1^m = \frac{\sum_{i=1}^{(1-\epsilon)n} \theta_i}{(1-\epsilon)n} + \theta', \quad \theta_2^m = \frac{\sum_{i=1}^{(1-\epsilon)n} \theta_i + \theta_1^m}{(1-\epsilon)n + 1} \tag{9}$$

This way, at the end of the first epoch of the MWU aggregation, the adversary manages to assign a weight close to 1 to the parties with update θ_2^m. In the case of MWUAvg and MWUOpt, all the benign

Algorithm 5 Little is enough attack (LIE) (Baruch et al., 2019)

1: **Input**: n, ϵ, mean and variance vectors of benign updates $\bar{\mu}$, $\bar{\sigma}$
2: Number of workers required for majority:

$$s = \lfloor \frac{n}{2} + 1 \rfloor - \epsilon n$$

3: Using z-table, set $z = \underset{z}{max}\left(\phi(z) < \frac{(1-\epsilon)n - s}{(1-\epsilon)n}\right)$
4: **for** $j \in [d]$ **do**
5: $\quad \theta_m^j \leftarrow \bar{\mu}_j + z\bar{\sigma}_j$
6: **end for**
7: **Output**: malicious update θ_m

parties are assigned negligible weights, which completely jeopardizes the accuracy of aggregation. To be effective, the adversary needs just two malicious parties who share the two malicious updates.

B.5 MEMBERSHIP INFERENCE ATTACKS

Recent research has shown the susceptibility of the federated learning to active and passive inference attacks (Melis et al., 2019; Nasr et al., 2019). In the passive case, the attacker, either the server or some of the parties, simply observes the updated model parameters to mount membership inference attacks. In the active case, however, the attacker tampers with the training of the victim model/s in order to infer membership of target data in *any of the benign party's data*. Specifically, the attacker shares malicious updates and forces the victim model/s to share more information about the members of their training data that are of the attacker's interest. This attack, called *gradient ascent* attack (Nasr et al., 2019), exploits that the SGD optimization updates model parameters in the opposite direction of the gradient of the loss over the private training data. Let $\mathbf{x}$ be a record of attacker's interest and θ_a be the current global model. The attacker *crafts the malicious update θ^m by updating parameters of θ_a in the same direction of the gradient of the loss on* $\mathbf{x}$, i.e., performs gradient ascent as: $\theta^m = \theta_a + \gamma \frac{\partial L_{\mathbf{x}}}{\partial \theta_a}$. Such θ^m, when combined with the benign updates, increases the loss of the resulting global model, θ'_a, on $\mathbf{x}$. If $\mathbf{x}$ is in the training data of some party, SGD on θ'_a by this party will sharply reduce the loss of $\mathbf{x}$. On the other hand, if $\mathbf{x}$ is not in any party's training data, the loss of $\mathbf{x}$ will remain almost unchanged. Therefore, this attack increases the gap between the losses of θ_a on members and non-members and facilitates membership inference.

C ROBUST MEAN ESTIMATION

Cronus uses the robust mean estimation algorithm proposed by Diakonikolas et al. (Diakonikolas et al., 2017), which achieves the optimal error bound. The original algorithm is described in Algorithm 6. f_{Cronus} applies RobustFilter on the prediction of each public data. Recall that the intuition of this robust mean estimation (Diakonikolas et al., 2017) is that the empirical mean of the uncorrupted points should be concentrate nicely to the true mean μ of the distribution. Thus, if the empirical mean is $\hat{\mu}$ far from the true mean of the distribution, then along the direction $\hat{\mu} - \mu$, the outliers must be the source of the deviation and the variance is much larger than it should be. As a result, corrupted direction $\hat{\mu} - \mu$ can be detected as a large eigenvector of the empirical covariance. Hence, in the Algorithm 6 finds the direction v^*, which has the largest variance and projects the deviation of all the inputs from the empirical mean in this direction. Then filter out a randomized fraction of the data which are farthest from the mean, $\bar{Y}$, along this direction. Repeat the process until the variance is not large in every direction and then output the sample mean on the subsets. For further details of the robust mean estimation, please refer to (Diakonikolas et al., 2017; Li, 2018).

Algorithm 6 RobustFilter [Algorithm 3 (Diakonikolas et al., 2017)]

Input: S, ϵ, k=0
while True **do**:
  Compute $\bar{Y}$,Σ, the mean and covariance matrix of S.
  Find the eigenvector v^* with highest eigenvalue λ^* of Σ
  if $\lambda^* \leq 9$ **then**
    Let $\bar{Y}$ = $\bar{Y}$
    break
  else
    Draw Z from the distribution on [0,1] with probability density function $2x$
    Let T = $Z \max\{|v^* \cdot (Y - \bar{Y})| : Y \in S\}$.
    Set $S = \{Y \in S : |v^* \cdot (Y - \bar{Y})| < T\}$
  end if
end while
Output: $\bar{Y}$

The sample complexity of the Algorithm 6 is $\Theta(d/\epsilon \log d)$, where d is the dimensionality of the inputs. In the federated learning, d is the size of the model and in Cronus, d is the size of prediction. Table 4 shows the sample complexities for training different ML benchmark models with federated learning

versus Cronus, using RobustFilter's algorithm RobustFilter. We note that Cronus significantly reduces sample complexity (by an order of 10^5), and therefore, unlike any existing federated learning, can achieve strong theoretical error guarantees even with a small number of parties in the network.

Table 4: Sample complexity, $\Theta((d/\epsilon)\log d)$, of Cronus aggregation, using (Diakonikolas et al., 2017), for parameters and predictions updates. For parameters, d is size of model; for predictions, d is the number of classes in the classification task. The ratio shows that Cronus learning can achieve the same error guarantee as in federated learning, but with a network which is 5 orders of magnitude smaller, for benchmark ML tasks.

Dataset	Sample complexity ratio of Federated learning over Cronus
SVHN	1.2×10^5
MNIST	3.3×10^5
Purchase	2.75×10^5
CIFAR10	10.4×10^5

For the theoretical analysis, Algorithm 6 uses randomized filtering (step 9) and repeats until the stop condition is satisfied (step 5). The follow-up works from the same authors (Li, 2018; Diakonikolas et al., 2019) suggest a simpler algorithm to obtain a better performance in practice: (1) in each iteration, remove a deterministic fraction of the data instead of a random fraction. (2) repeat the filter for constant iterations in total. In the evaluation, we filter out $\epsilon/2$ fraction of the inputs in each iteration (step 12) and repeat the filter process 2 times (step 7) and to obtain a good performance.

D Missing Experimental Details

D.1 Details of datasets and model architectures

We use four datasets in our evaluation, whose details follow.

SVHN. SVHN (Netzer et al., 2011) dataset contains Google's street view images of house numbers. The images are 32x32, with 3 floating point numbers containing the RGB color information of each pixel. We use the extended SVHN dataset with 630,420 samples to train 32 party models each with 5,000 training samples; the public data size is 10,000. We use validation and test data of sizes 2,500 each. The reference data required for adversarial regularization is of the same size as that of training data for the cases of all the datasets.

MNIST. MNIST (LeCun et al., 1998) dataset contains 28x28 images of handwritten digits and is composed of 60,000 training samples and 10,000 test samples. The dataset contains 10 classes each with 60,000 training and 1,000 test samples. We use validation and test data of sizes 1,000 each. We use 28 parties each with 2,000 training and reference samples, and public data size is 10,000.

Purchase. Purchase (pur, 2019) dataset contains the shopping records of several thousand online customers, extracted during Kaggle's Acquire Valued Shopper challenge (pur, 2019). The dataset contains 197,324 data records with feature vectors of 600 dimensions and corresponding class label from one of total 100 classes. We use validation and test data of sizes 2,500 each. We use 16 parties each with 10,000 training and reference data, and public data size is 10,000.

CIFAR10. CIFAR10 (Krizhevsky & Hinton, 2009) has 60,000 color (RGB) images (50,000 for training and 10,000 for testing), each of 32×32 pixels. The images are clustered into 10 classes based on the objects in the images and each class has 5,000 training and 1,000 test images. We use validation and test data of sizes 2,500 each. We use 16 parties each with 2,500 training data, and public data size is 10,000.

Model architectures. For SVHN, we use a neural network with three convolution layers and one fully connected layer, and Relu activations. For the MNIST dataset, we use a fully connected neural network with layer sizes {784, 256, 64, 10} and Relu activations. For the Purchase dataset, we use fully connected neural networks with layer sizes {600, 1024, 100} and Tanh activations. For CIFAR10 dataset, we use DenseNet architecture (Huang et al., 2017) with 100 layers and growth rate of 12. For the heterogeneity experiments with Purchase dataset, we use 5 fully connected networks with hidden layer sizes [{},{1024},{512, 256},{1024, 256}, {1024, 512, 256}]; here {} implies that the corresponding model has no hidden layers.

Table 5: Evaluation of the conventional federated learning with various aggregation schemes with Cronus learning using the strong poisoning attacks described in Section 5. Robustness in Table 3 is measured as the ratio of the accuracy of the final model/s when the strongest attack is mounted and the accuracy in the benign setting; the strongest attack is determined empirically as the one that maximally reduces the accuracy of the corresponding federated learning aggregation.

Dataset		Federated learning with various aggregation algorithms						Cronus
		Mean	Median	MwuAvg	MwuOpt	Bulyan	Krum	
SVHN	Accuracy (Benign)	**95.9**	94.8	93.9	94.4	94.5	89.6	91.1
	Label flip	92.9	90.1	91.2	89.3	93.9	88.6	89.8
	LIE	14.8	14.5	91.6	92.0	15.5	16.2	91.5
	OFOM	0.9	94.5	0.9	0.7	94.4	89.0	91.0
	PAF	12.8	16.4	95.1	93.1	93.4	87.5	91.1
MNIST	Accuracy (Benign)	96.7	96.5	**97.2**	97.4	96.9	93.3	95.2
	Label flip	96.3	94.4	94.7	93.6	96.8	89.9	95.0
	LIE	95.1	93.1	95.6	96.7	94.1	94.3	95.9
	OFOM	22.1	97.3	25.3	36.0	97.1	94.4	96.1
	PAF	9.6	91.5	96.9	12.7	97.1	94.0	96.2
Purchase	Accuracy (Benign)	93.3	93.0	**93.6**	92.5	92.8	72.1	89.6
	Label flip	88.9	89.9	63.4	67.6	91.7	74.8	88.0
	LIE	2.5	69.3	92.2	85.6	81.8	49.6	89.2
	OFOM	1.4	92.8	1.8	1.1	92.6	74.5	89.4
	PAF	1.1	12.5	93.0	88.0	91.0	76.6	89.4
CIFAR10	Accuracy (Benign)	88.4	**89.1**	86.2	87.6	89.0	84.5	80.1
	Label flip	–	–	–	–	–	–	79.8
	LIE	18.9	61.2	86.0	84.3	75.6	18.0	78.0
	OFOM	12.9	89.5	14.2	12.8	89.1	85.0	78.5
	PAF	11.3	15.1	86.4	85.0	89.0	839	79.0

Training hyper-parameters. The initialization and collaboration phases of SVHN, MNIST, and Purchase trained models are of 50 epochs each. In both the phases, we train party models on their local training data using Adam optimizer at 0.0005 learning rate. Additionally, in collaboration phase, i.e., for epochs 50-100, we train party models on public data, $(X_p, \bar{Y})$, using SGD optimizer at a learning rate of 0.001. For CIFAR10, we train models for 200 epochs using SGD optimizer with 0.1 learning rate, 0.9 momentum, and 10^-4 weight decay in both the phases. Additionally, in collaboration phase, we train the models on public data using SGD optimizer at 0.01 learning rate, 0.99 momentum, and 10^{-6} weight decay.

For experiments of membership privacy assessment, we use state-of-the-art whitebox inference model proposed in (Nasr et al., 2019), and use the gradients and outputs of its last layer, in addition to the blackbox access features including prediction of input and its cross-entropy loss. We train the inference model using Adam optimizer at a learning rate of 0.0001 for 100 epochs.

D.2 Missing Empirical Results

In this section, we provide the experimental details omitted in Section 5.

D.2.1 Robustness

Here, we give the complete robustness assessment of Cronus and FedAvg. In Table 3 of Section 5.1, for each dataset and each aggregation algorithm, we showed the accuracy of the attack that is strongest among all the attacks discussed in Section 5. We compute empirical robustness of aggregation algorithms using this strongest attack as described in Section 5. In Table 5, we give the complete evaluation of all the attacks on all of the aggregation algorithms and datasets we consider in this work. The 'Accuracy (Benign)' row of each dataset shows the results in the absence of adversary. The worst accuracy for a combination of aggregation algorithm and dataset is highlighted in the corresponding column; the corresponding strongest attack can be found from the label of the row of

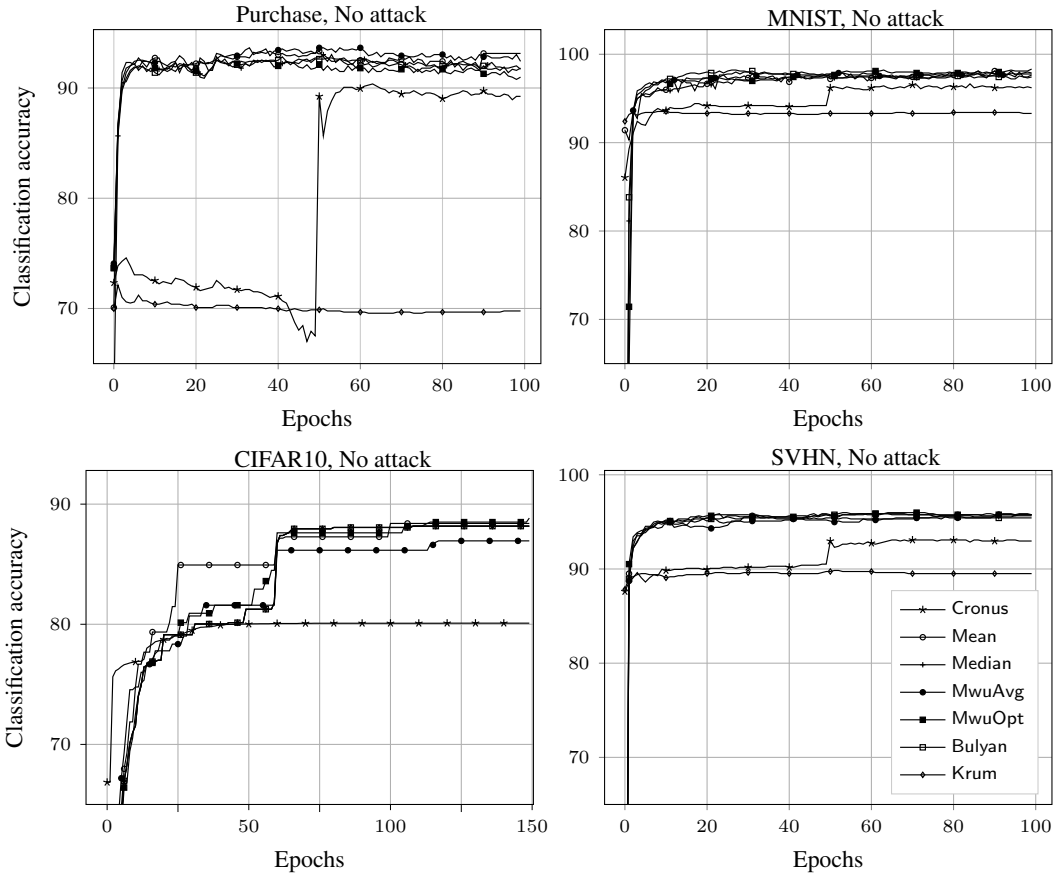

Figure 1: Convergence of Cronus and existing federated learning algorithms in **benign setting**. Cronus incurs only a slight degragation in accuracy compared to existing algorithms, while improves significantly over stand-alone training.

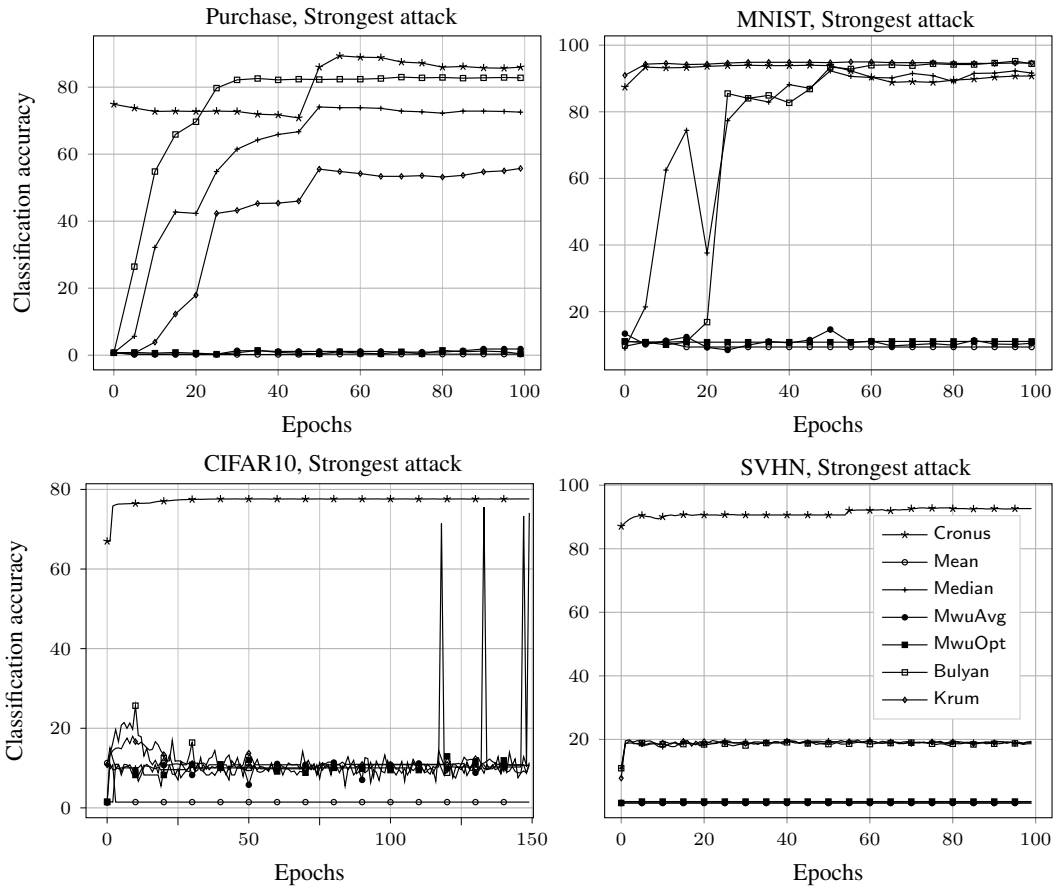

Figure 2: Convergence of Cronus and existing federated learning algorithms in **adversarial setting**. Accuracy of Cronus in adversarial setting is almost the same as in benign setting (shown in Figure 1) due to its high robustness. Except for CIFAR10, for which only collaboration phase is shown, both the Cronus training phases are shown in figure and the collaboration phase starts at epoch 50.

the highlighted cell. Observe that, label flip attack seems to have lower effect on mean aggregation than the other aggregations; this is because, unlike other aggregations, in case of mean, there is only single malicious client. Note that, MWUAvg and MWUOpt aggregations are robust against all the existing attacks in the literature, but are completely ineffective against the attack we introduced in Section B.4. Also, note that, Bulyan aggregation is empirically the most robust aggregation after Cronus, but it allows only 25 - 33% malicious clients compared to other aggregation algorithms such as Krum, in other words, Bulyan has a very low breaking point. The numbers of malicious parties used in each of our experiments are given in Table 2.

Convergence plots. Figure 1 shows the convergence of Cronus and various aggregation algorithms in federated learning for the benign setting. In Figure 2, However, all of the existing aggregation schemes in federated learning fail to converge under at least one poisoning attack. Cronus incurs only a slight reduction in accuracy at a significantly higher gain in robustness and privacy as shown in Sections 5.1 and 5.1.

D.2.2 PRIVACY

In this section, by measuring the privacy risk using state-of-the-art membership inference attack, we show that Cronus considerably reduces the privacy risk compared with federated learning. We also show the compatibility of Cronus with existing privacy-preserving mechanisms. We evaluate the risk of membership inference attacks on the participants' private training data during collaboration, and the effect of privacy preserving mechanisms (Abadi et al., 2016; Nasr et al., 2018). As described in the threat model in Section 2, we assume the central server and other participants to run passive and active membership inference attacks (Nasr et al., 2019) on individual party updates and their aggregates. We use Purchase, SVHN, and CIFAR10 datasets for our evaluation when 4 parties collaborate.

D.2.3 *Passive membership inference attacks*

In the case of passive membership inference attacks, the server isolates the parties and mounts the attack separately on each of the collected updates, i.e., in case of FedAvg, attack is mounted on the parameter updates of each party and in case of Cronus, attack is mounted on the model obtained by training on the predictions shared by each party. We also evaluate the privacy risk when the attack is mounted on the aggregate of these updates.

The results are shown in Table 6. **The updates in FedAvg are highly susceptible to membership inference unlike the updates in Cronus.** For the Purchase dataset, attack accuracy against the individual and aggregated updates in FedAvg is 78.1% and 80.1%, respectively, whereas in Cronus they are 51.7% and 51.9%. Unlike the prediction updates in Cronus, the high dimensional parameter updates in FedAvg encode a significantly higher amount of information about the party's local data. Furthermore, knowledge transfer acts as a strong regularization method and mitigates the risk of membership inference attacks (Geoffrey & amd Dean Jeff, 2014; Song & Chai, 2018). It's important to note that knowledge transfer through predictions, on a dataset other than the training data, makes the behavior of the student model more indistinguishable on its training versus unseen data. This happens as the distillation process does not carry the exceptionally distinguishable characteristics of the model on its training data, and results in smooth decision boundaries of the student model around the teacher's training data. We observe similar results for SVHN and CIFAR10 datasets.

We also evaluate Cronus and FedAvg, when models use adversarial regularization (Nasr et al., 2018) during local training to improve membership privacy. We note that, **adversarial regularization improves membership privacy of both FedAvg and Cronus**, but the increase is smaller for Cronus due to its inherent resilience to membership inference. For Purchase dataset with FedAvg, the risk to individual and aggregated updates reduces by 9.9% and 12.7%, respectively, while for Purchase with Cronus, the risk to individual and aggregated updates is already very small, and it further reduces by 0.6% and 1.1%, respectively. Similarly for CIFAR10, privacy improvement in FedAvg is significantly more than in Cronus. However, the privacy improvement for SVHN is very small even in case of FedAvg, due to large gaps in train and test accuracies at stronger adversarial regularization.

Table 6: Accuracy of passive and active membership inference attacks with central server as adversary. We also evaluate effect of adversarial regularization (with parameter λ) used to preserve membership privacy. We use 4 parties and data per party as in Table **??**.

Dataset	Federated learning algorithm	Attack on party update		Attack on aggregated update	
		Passive attack acc.	Active attack acc.	Passive attack acc.	Active attack acc.
Purchase (without privacy)	FedAvg	77.1	84.9	74.7	82.7
	Cronus	54.6	54.9	53.6	54.7
Purchase (with membership privacy, $\lambda = 3$)	FedAvg	70.8	77.3	69.9	77.0
	Cronus	54.1	51.5	53.7	54.6
SVHN (without privacy)	FedAvg	64.8	67.3	59.9	64.3
	Cronus	55.6	53.1	56.5	55.7
SVHN (with membership privacy, $\lambda = 0.5$)	FedAvg	64.9	67.0	60.0	64.2
	Cronus	54.1	56.9	55.6	55.0
CIFAR10 (without privacy)	FedAvg	79.9	80.5	76.8	77.1
	Cronus	57.0	57.8	55.5	56.7
CVIFAR10 (with membership privacy, $\lambda = 1$)	FedAvg	59.9	64.1	59.6	62.2
	Cronus	52.9	54.4	52.6	57.0

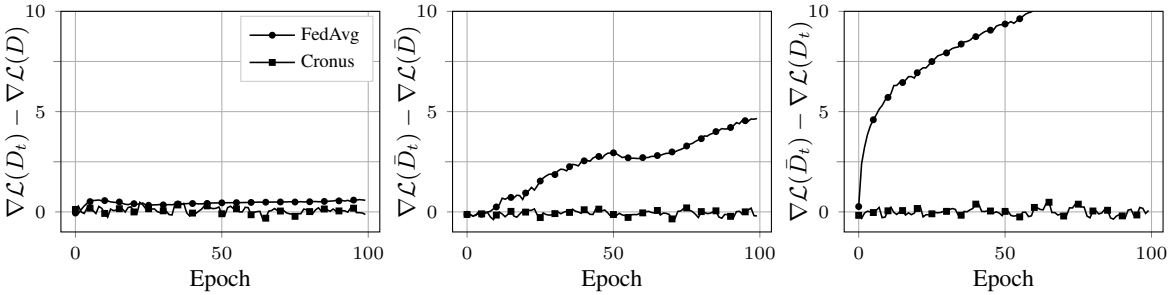

Figure 3: Difference in the gradient-norms, $\nabla\mathcal{L}(D)$, of the last layer of aggregated model on the target and non-target data (Purchase100 data). In the context of active membership inference attacks, D, D_t, $\bar{D}$, and $\bar{D}_t$ denote non-target members, target members, non-target non-members, and target non-members, respectively.

D.2.4 *Active membership inference attacks*

In each epoch, the server manipulates the aggregate update that it broadcasts to parties by performing gradient ascent on the *aggregated update* for a set of target data (Nasr et al., 2019). In FedAvg, gradient ascent is performed directly on the aggregated parameters. In Cronus, for running such attack, the server needs to train a model on the aggregated predictions while performing gradient ascent on the target data, and then, shares predictions of this model with the parties; we ensure that such model has accuracy close to the accuracy of party models in given epoch.

Table 6 shows the results. **The active attacks significantly increase the privacy risk of the target data in case of FedAvg**: for Purchase dataset, the risk due to individual update increases by 7.8% (77.1% to 84.9%), while due to aggregated update increases by 8% (74.7% to 82.7%). But, **in case of Cronus, the active attacks are ineffective and the increase in risk is negligible**: 0.3% for individual update while 1.1% for aggregated update. In Figure 3, we show the effect of gradient ascent on the difference in gradient-norms of target and non-target data for aggregated model for the Purchase dataset. This directly correlates with the success of membership inference (Nasr et al., 2019). We observe that, for FedAvg on SVHN dataset, the active attacks increase the risk to individual and aggregate updates by 2.5% and 4.4%, respectively, but, the increased risk negligible for Cronus.

D.2.5 *Differential privacy*

We compare the performance of Cronus and the conventional federated learning with user-level (Brendan et al., 2018) (the server applies differentially private mechanism) or record-level (Abadi et al., 2016) (each party applies differentially private mechanism) differential privacy. For both of these

Table 7: Accuracy and robustness of models for record level DP (Abadi et al., 2016) with $\epsilon = 15.4$ on the SVHN dataset. The baseline stand-alone accuracy is 87%.

# of parties	Accuracy (Benign)		Worst accuracy		Robustness	
	FedAvg	Cronus	FedAvg	Cronus	FedAvg	Cronus
32	65.7	85.8	4.5	83.1	0.07	**0.97**
16	74.3	84.8	8.1	84.0	0.11	**0.99**
8	77.9	84.2	0.9	82.2	0.01	**0.98**
4	81.6	81.5	1.7	81.2	0.02	**0.98**

comparisons, we use SVHN dataset with 32 benign parties and the model architecture as described in Table 2, and use the moments accountant method (and the code) (Abadi et al., 2016; tf, 2019) to bound the total privacy risk. Note that we train the *whole model* using differential privacy, as opposed to only training the last layer (Abadi et al., 2016). Our results show that robustness property of Cronus , as expected, is preserved with differential privacy.

User-level DP (Brendan et al., 2018). The user-level DP (UDP-FedAvg) algorithm proposed by McMahan et al. (Brendan et al., 2018) cannot lead to training good accuracy models, when the number of parties in each epoch is small. Even for large privacy budgets, i.e., $\epsilon = 100$, the parties do not benefit from collaboration, and with the user-level DP, the global model in FedAvg achieves close to random-guess accuracy. We observed similar results for running user-level DP on Cronus with small number of participants. This result is expected as the sensitivity of the element-wise mean aggregation algorithm is inversely proportional to the number of parties, and a very large number of parties is required to reduce the noise, e.g., (Brendan et al., 2018) uses 5000 parties in each epoch which is not realistic in cases where a few data holder (hospital) collaborate.

Record-level DP (Abadi et al., 2016). We empirically show that the conventional parameter aggregation in FedAvg is not suitable to provide the record-level DP, and is also susceptible to poisoning attacks. The results are shown in Table 7 for SVHN dataset. The accuracy of the models for federated learning or Cronus, with DP-SGD, cannot reach the accuracy of stand-alone training, which makes the collaboration useless. The results of the strongest poisoning attacks show that DP-SGD FedAvg has no robustness against the attacks.

D.2.6 CRONUS WITH HETEROGENEOUS MODEL ARCHITECTURES

Due to the use of predictions based updates, Cronus allows parties with heterogeneous model architectures to participate in collaboration. Below, we compare different aspects of the homogeneous and heterogeneous collaborations. We use Purchase data and 5 fully connected models, which we call A1, A2, A3, A4, and A5, with hidden layer sizes {}, {1024}, {512, 256}, {1024, 256}, and {1024, 512, 256} respectively. Note that, A1 models, called *bad models*, have lower capacity and accuracy than A2-5 models, which we call *good models*. We denote by $P_{j:k}$ the model architectures of parties $\in [j, k]$. We denote the entire collaboration in curly brackets, e.g., we denote the collaboration of 3 sets of 4 models, i.e. 12 models in total, each with either of A3, A3, or A4 models by $\{P_{1:4}$ = A2, $P_{5:8}$ = A3, $P_{9:12}$ = A4$\}$. In tables, accuracy of an architecture is the average accuracy of all the models with that architecture, e.g., in Table 9 accuracy of A2 is average of accuracies of all models with A2 architecture in the two collaborations.

First, we show that the heterogeneous collaboration between models of equivalent capacities does not reduce the accuracy of party models compared to its homogeneous counterparts. We consider four homogeneous collaborations each of 16 parties such that $\{P_{1:16}$ = A2$\}$, $\{P_{1:16}$ = A3$\}$, $\{P_{1:16}$ = A4$\}$, and $\{P_{1:16}$ = A5$\}$, and compare it with a heterogeneous collaboration: $\{P_{1:4}$ = A2, $P_{5:8}$ = A3, $P_{9:12}$ = A4, $P_{13:16}$ = A5$\}$. The results are shown in Table 8.

Next, we show that **the presence of a few bad models does not affect the accuracy of the good models in the heterogeneous collaboration, while significantly benefits the bad models**. Specifically, we show that accuracy of the collaboration of 12 good models, i.e., $\{P_{1:4}$ = A2, $P_{5:8}$ = A3, $P_{9:12}$ = A4$\}$ remains unaffected even if 4 bad models are added to it, i.e., $P_{13:16}$ = A2, as shown in Table 9.

Finally, we show that **heterogeneity allows for more knowledge sharing via collaboration and always improves the utility of collaborations**. We consider 4 homogeneous collaborations: {$P_{1:4}$ = A1}, {$P_{1:4}$ = A2}, {$P_{1:4}$ = A3}, and {$P_{1:4}$ = A4} and compare them with a heterogeneous collaboration that includes all these 16 parties, i.e., {$P_{1:4}$ = A1, $P_{5:8}$ = A2, $P_{9:12}$ = A3, $P_{13:16}$ = A4}. Table 10 shows that including more participants clearly benefits all types of models, although the bad models benefit more than the good ones. For instance, A1 models improve by 8% from 70.1% to 78.1% due to heterogeneous collaboration, while A2, A3, and A4 models improve by 3.2%, 2.2%, and 1.4%, respectively.

Table 8: Comparison between heterogeneous and homogeneous collaborations in Cronus.

Homogeneous				Heterogeneous
$P_{1:16} \rightarrow$ A2	A3	A4	A5	{$P_{1:4}$ = A2, $P_{5:8}$ = A3 $P_{9:12}$ = A4, $P_{13:16}$ = A5}
89.6	89.3	88.4	88.6	89.3

Table 9: Effect of the presence of low accuracy bad models on the performance of higher accuracy good models. n is the number of collaborating parties.

Models	Heterogeneous {$P_{1:4}$ = A2, $P_{5:8}$ = A3, $P_{9:12}$ = A4}	Heterogeneous {$P_{1:4}$ = A2, $P_{5:8}$ = A3, $P_{9:12}$ = A4, $P_{13:16}$ = A1}
A1	-	78.1
A2	88.5	88.7
A3	88.6	88.1
A4	88.7	88.1

Table 10: More participation due to heterogeneity always improves the overall utility of the collaboration.

Models	Homogeneous 4 small collaborations $P_{1:4}$ = A1/A2/A3/A4	Heterogeneous {$P_{1:4}$ = A2, $P_{5:8}$ = A3, $P_{9:12}$ = A4, $P_{13:16}$ = A1}
A1	70.1	78.1
A2	85.5	88.7
A3	85.9	88.1
A4	86.7	88.1