SafeNet: Mitigating Data Poisoning Attacks on Private Machine Learning

Harsh Chaudhari
Northeastern University
chaudhari.ha@northeastern.edu

Matthew Jagielski
Google Research
jagielski@google.com

Alina Oprea
Northeastern University
a.oprea@northeastern.edu

Abstract—**Secure multiparty computation (MPC) has been proposed to allow multiple mutually distrustful data owners to jointly train machine learning (ML) models on their combined data. However, the datasets used for training ML models might be under the control of an adversary mounting a data poisoning attack, and MPC prevents inspecting training sets to detect poisoning. We show that multiple MPC frameworks for private ML training are susceptible to backdoor and targeted poisoning attacks. To mitigate this, we propose SafeNet, a framework for building ensemble models in MPC with formal guarantees of robustness to data poisoning attacks. We extend the security definition of private ML training to account for poisoning and prove that our SafeNet design satisfies the definition. We demonstrate SafeNet's efficiency, accuracy, and resilience to poisoning on several machine learning datasets and models. For instance, SafeNet reduces backdoor attack success from 100% to 0% for a neural network model, while achieving $39\times$ faster training and $36\times$ less communication than the four-party MPC framework of Dalskov et al. [26].**

I. Introduction

Machine learning (ML) has been successful in a broad range of application areas such as medicine, finance, and recommendation systems. Consequently, technology companies such as Amazon, Google, Microsoft, and IBM provide machine learning as a service (MLaaS) for ML training and prediction. In these services, data owners outsource their ML computations to a set of more computationally powerful servers. However, in many instances, the client data used for ML training or classification is sensitive and may be subject to privacy requirements. Regulations such as GDPR, HIPAA and PCR, data sovereignty issues, and user privacy concern are common reasons preventing organizations from collecting user data and training more accurate ML models. These privacy requirements have led to the design of privacy-preserving ML training methods, including the use of secure multiparty computation (MPC).

Recent literature in the area of MPC for ML proposes privacy-preserving machine learning (PPML) training frameworks [64], [62], [75], [69], [27], [76], [26], [1] for various machine learning models such as logistic regression, neural networks, and random forests. In these models, data owners outsource shares of their data to a set of servers and the servers run MPC protocols for ML training and prediction. An implicit assumption for security is that the underlying datasets provided by data owners during training have not been influenced by an adversary. However, research in adversarial machine learning has shown that data poisoning attacks pose a high risk to the integrity of trained ML models [10], [46], [40], [36]. Data poisoning becomes a particularly relevant threat in PPML systems, as multiple data owners contribute secret shares of their datasets for jointly training a ML model inside the MPC, and poisoned samples cannot be easily detected.

In this paper, we study the impact of data poisoning attacks on MPC frameworks for private ML training. Our first observation is that the security definition of MPC for private ML training does not account for data owners with poisoned data. Therefore, we extend the security definition by considering an adversary who can poison the datasets of a subset of owners, while at the same time controlling a subset of the servers in the MPC protocol. Under our threat model, we empirically demonstrate that poisoning attacks are a significant threat to the setting of private ML training. We show the impact of backdoor [40], [22] and targeted [50], [36] poisoning attacks on four MPC frameworks and three datasets, using logistic regression and neural networks models. We show that with control of just a single owner and its dataset (out of a set of 20 owners contributing data for training), the adversary achieves 100% success rate for a backdoor attack, and higher than 83% success rate for a targeted attack. These attacks are stealthy and cannot be detected by simply monitoring standard ML accuracy metrics.

To mitigate these attacks, we propose SafeNet, an ensemble framework for private ML training designed as a general defense against poisoning attacks. Rather than attempting to implement an existing poisoning defense in MPC, we observe that the structure of the MPC threat model permits a more general and efficient solution. Our main insight is to require individual data owners to train ML models locally, based on their own datasets, and secret share the resulting ensemble of models in the MPC. We filter out local models with low accuracy on a validation dataset, and use the remaining models to make predictions using a majority voting protocol performed inside the MPC. While this permits stronger model poisoning attacks, the natural partitioning of the MPC setting prevents an adversary from poisoning more than a fixed subset of the models, resulting in a limited number of poisoned models in the ensemble. We perform a detailed analysis of the robustness properties of SafeNet, and provide lower bounds on the ensemble's accuracy based on the error rate on the local models in the ensemble and the number of poisoned models, as well as a prediction certification procedure for arbitrary inputs. Furthermore, we show empirically that SafeNet successfully mitigates backdoor and targeted poisoning attacks, while retaining high accuracy on the ML prediction tasks. In addition, our approach is efficient, as ML model training is performed locally by each data owner, and only the ensemble filtering and prediction protocols are performed in the MPC. This provides large performance improvements in ML training compared to existing PPML frameworks, while simultaneously mitigating poisoning attacks. For instance, for one neural network model, SafeNet performs training 39× faster than the [26] PPML protocol and requires 36× less communication. Finally, we investigate settings with diverse data distributions among owners, and show that extreme data imbalance conditions might impact SafeNet's accuracy.

To summarize, our contributions are as follows:

Poisoning-aware Threat Model for Private Machine Learning. We extend the MPC security definition for private machine learning to encompass the threat of data poisoning attacks. In our threat model, the adversary can poisoned a subset t out of m data owners, and control T out of N servers participating in the MPC.

SafeNet Ensemble Design. We propose SafeNet as an ensemble of models trained locally by data owners to circumvent poisoning attacks in MPC. We show that SafeNet satisfies security under our threat model, and we provide lower bounds for robustness depending on the level of poisoning and the error rate of the underlying base models in the ensemble. Our SafeNet design is agnostic to the underlying MPC framework and we show instantiations over four different MPC frameworks, supporting two, three and four servers.

Comprehensive Evaluation. We show the impact of existing backdoor and targeted poisoning attacks on several existing PPML systems [30], [4], [26] and three datasets, using logistic regression and neural network models. We also empirically demonstrate the resilience of SafeNet against these attacks, for an adversary compromising up to 9 out of 20 data owners. We report the gains in training time and communication cost for SafeNet compared to existing PPML frameworks. Finally, we compare SafeNet with state-of-the-art defenses against poisoning in federated learning [17] and show its enhanced certified robustness.

II. Background and Related Work

We provide background on secure multi-party computation and poisoning attacks in ML, and discuss related work in the area of adversarial ML and MPC.

A. Secure Multi-Party Computation

Secure Multi-Party Computation (MPC) [80], [7], [37], [43], [29] allows a set of n mutually distrusting parties to compute a joint function f, so that collusion of any t parties cannot modify the output of computation (*correctness*) or learn any information beyond what is revealed by the output (*privacy*). The area of MPC can be categorized into honest majority [7], [63], [4], [19], [13] and dishonest majority [80], [29], [28], [61], [37]. The settings of two-party computation (2PC) [80], [57], [56], [67], three parties (3PC) [3], [4], [63], [14], and four parties (4PC) [44], [14], [39], [20], [26] have been widely studied as they provide efficient protocols. Additionally, recent works in the area of privacy preserving ML propose training and prediction frameworks [64], [62], [75], [69], [53], [70], [76], [1], [68] built on top of the above MPC settings. Particularly, most of the frameworks are deployed in the outsourced computation setting where the data is secret-shared to a set of servers which perform training and prediction using MPC.

B. Data Poisoning Attacks

Data poisoning attacks model the risk of adversarial control on a subset of the training dataset. In a backdoor attack [66], [40], [22], an adversary seeks to add a "trigger" or backdoor pattern into the model. The trigger is a perturbation in feature space, which is applied to poisoned samples in training to induce misclassification on backdoored samples at testing. In a targeted attack [50], [51], [73], the adversary's goal is to change the classifier prediction for a small number of specific test samples. Backdoor and targeted attacks can be difficult to detect, due to the subtle impact they have on the ML model.

C. Related Work

While both MPC and adversarial machine learning have been the topic of fervent research, work connecting them is still nascent. We are only aware of several recent research papers that attempt to bridge these areas. Recent works [55], [18] show that MPC algorithms applied at test time can be compromised by malicious users, allowing for efficient *model extraction* attacks. Second, Escudero et al. [33] show that running a semi-honest MPC protocol with malicious parties can result in backdoor attacks in the resulting SVM model. Both these works, as well as our own, demonstrate the difficulty of aligning the guarantees of MPC with the additional desiderata of adversarial machine learning. We demonstrate the effectiveness of data poisoning attacks in MPC for neural networks and logistic regression models, and propose a novel ensemble training algorithm in SafeNet to defend against poisoning attacks in MPC.

Model ensembles have been proposed as a defense for ML poisoning in prior work [9], [47]. Crucially, these works assume that a central dataset is used with a subset of the data being poisoned, from which all models in the ensemble are trained non-privately. In our setting, however, we are able to leverage the trust model of MPC, which limits the number of poisoned models in the ensemble and can provide stronger robustness guarantees. Ensembles have also been proposed in MPC to protect data privacy [23].

III. SafeNet Framework

In this section, we first describe the adversarial model considered in state-of-the-art privacy preserving ML training frameworks. We then extend this adversarial model by allowing the adversary to poison the training data of a

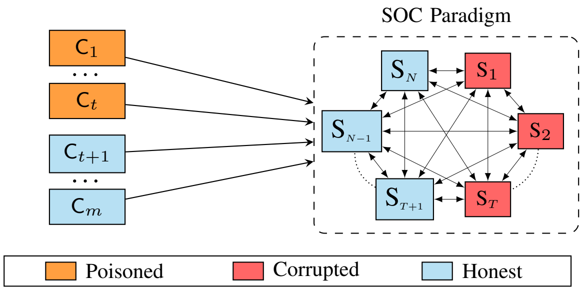


Fig. 1: Threat model considered in our setting. The adversary $\mathcal{A}_{\text{soc}}^{\text{p}}$ can poison at most t out of m data owners and corrupt at most T out of N servers participating in the MPC computation. C_i and $\mathcal{S}_j$ denote the i^{th} data owner and j^{th} server respectively.

subset of owners. Finally, we provide an overview of the SafeNet framework, discuss its realization as a collection of MPC protocols, and detail the underlying MPC building blocks required to implement SafeNet in a privacy-preserving manner.

A. Threat Model

Setup. We consider a set of m data owners $C = \cup_{k=1}^{m} \mathsf{C}_k$ who wish to train a joint machine learning model $\mathcal{M}$ on their combined dataset $\mathsf{D} = \cup_{k=1}^{m} D_k$. We adopt the Secure Outsourced Computation (SOC) paradigm [64], [62], [75], [69], [13], [70], [76], [1], [27], [26] for training model $\mathcal{M}$ privately, where the owners secret-share their respective datasets to a set of outsourced servers, who enact the role of mutually distrusting parties in the MPC and execute the necessary protocols to train $\mathcal{M}$. The final output after execution of these protocols is a trained model in secret-shared format among the servers. A single training/testing sample is expressed as $(\mathbf{x}_i, \mathrm{y}_i)$, where $\mathbf{x}_i$ is the input feature vector and y_i is its corresponding true label or class. We use $D_k = (\mathbf{X}_k, \mathbf{y}_k)$ to denote dataset of data owner C_k participating in the training process. Matrix $\mathbf{X}_k$ denotes a feature matrix where the number of rows represent the total training samples possessed by C_k and $\mathbf{y}_k$ denotes the corresponding vector of true labels.

Adversary in the SOC. Given a set $S = \{\mathcal{S}_1, \ldots, \mathcal{S}_N\}$ of outsourced servers, we define an adversary $\mathcal{A}_{\text{soc}}$, similar to prior work [64], [62], [70], [76], [1], [26]. $\mathcal{A}_{\text{soc}}$ can statically corrupt a subset $S_T \subset S$ of servers of size at most $T < N$. The exact values of N and T are dependent on the MPC protocols used for training the ML model privately. For instance, we experiment with two-party, three-party, and four-party protocols with one corrupt server. MPC defines two main adversaries: i) *Semi-honest*: Adversary follows the described protocol, but tries to derive additional information from the messages received from other parties during the protocol; ii) *Malicious*: Adversary has the ability to arbitrarily deviate during the execution of the protocol. For our security proofs, we assume the adversary to be semi-honest, but our approach can be extended to malicious adversaries.

Security Definition. MPC security is defined using the real world - ideal world paradigm [15]. In the real world, parties participating in the MPC interact during the execution of a protocol π in presence of an adversary $\mathcal{A}$. Let $\mathsf{REAL}[\mathbb{Z}, \mathcal{A}, \pi, \lambda]$ denote the output of the environment $\mathbb{Z}$ when interacting with $\mathcal{A}$ and the honest parties, who execute π on security parameter λ. This interaction is called the real-world interaction. Effectively, REAL is a function of the inputs/outputs and messages sent/received during the protocol. In the ideal world, the parties simply forward their inputs to a trusted functionality $\mathcal{F}$ and forward the functionality's response to the environment. Let $\mathsf{IDEAL}[\mathbb{Z}, \mathcal{S}, \mathcal{F}, \lambda]$ denote the output of the environment $\mathbb{Z}$ when interacting with adversary $\mathcal{S}$ and honest parties who run the protocol in presence of $\mathcal{F}$ with security parameter λ. The security definition states that the views of the adversary in the real and ideal world are indistinguishable:

Definition 1. *A protocol π securely realizes functionality $\mathcal{F}$ if for all environments $\mathbb{Z}$ and any adversary of type $\mathcal{A}_{soc}$, which corrupts a subset S_T of servers of size at most $T < N$ in the real world, then there exists a simulator $\mathcal{S}$ attacking the ideal world, such that* $\mathsf{IDEAL}[\mathbb{Z}, \mathcal{S}, \mathcal{F}, \lambda] \approx \mathsf{REAL}[\mathbb{Z}, \mathcal{A}_{soc}, \pi, \lambda]$.

Poisoning Adversary. Existing threat models for training ML models privately assume that the local datasets contributed towards training are not under the control of the adversary. However, data poisoning attacks have been shown to be a real threat when ML models are trained on crowdsourced data or data coming from untrusted sources [10], [65], [46]. Data poisoning becomes a particularly relevant risk in PPML systems, in which data owners contribute their own datasets for training a joint ML model. Additionally, the datasets are secret shared among the servers participating in the MPC, and potential poisoned samples (such as backdoored data) cannot be easily detected by the servers running the MPC protocol.

To account for this attack vector, we define a poisoning adversary $\mathcal{A}_{\text{p}}$ that can poison a subset of local datasets of size at most $t < m$. We call the data owners with poisoned data as *poisoned owners*, and we assume that the adversary can coordinate with the poisoned owners to achieve a certain adversarial goal. For example, the adversary can mount a backdoor attack, by selecting a backdoor pattern and poison the datasets under its control with the particular backdoor pattern.

Poisoning Robustness: We consider an ML model to be robust against a poisoning adversary $\mathcal{A}_{\text{p}}$, who poisons the datasets of t out of m owners, if it generates correct class predictions on new samples with high probability. We provide bounds on the level of poisoning tolerated by our designed framework to ensure robustness.

Our Adversary. We now define a new adversary $\mathcal{A}_{\text{soc}}^{\text{p}}$ for our threat model (Figure 1) that corrupts servers in the MPC and poisons the owners datasets:

- $\mathcal{A}_{\text{soc}}^{\text{p}}$ plays the role of $\mathcal{A}_{\text{p}}$ and poisons t out of m data owners that secret share their training data to the outsourced servers.

– $\mathcal{A}^{\mathsf{p}}_{\text{soc}}$ plays the role of $\mathcal{A}_{\text{soc}}$ and corrupts T out N servers taking part in the MPC computation.

Note that the poisoned owners that $\mathcal{A}^{\mathsf{p}}_{\text{soc}}$ controls do not interfere in the execution of the MPC protocols after secret-sharing their data and also do not influence the honest owners. Additionally, to measure the attack success of the poisoning attacks, adversary $\mathcal{A}_{\mathsf{p}}$ needs query access to the trained model (inside the MPC) at test time. To replicate this we let $\mathcal{A}^{\mathsf{p}}_{\text{soc}}$ use its poisoned data owners from the training phase to query the final trained model at test time.

Functionality $\mathcal{F}_{\mathsf{pTrain}}$. Based on our newly introduced threat model, we construct a new functionality $\mathcal{F}_{\mathsf{pTrain}}$ in Figure 2 to accommodate poisoned data.

Functionality $\mathcal{F}_{\mathsf{pTrain}}$

Input: $\mathcal{F}_{\mathsf{pTrain}}$ receives secret-shares of D_i and a_i from each owner C_i, where D_i is a dataset and a_i an auxiliary input.

Computation: On receiving inputs from the owners, $\mathcal{F}_{\mathsf{pTrain}}$ computes $O = f(D_1, ..., D_m, a_1, \ldots, a_m)$, where f and O denotes the training algorithm and the output of the algorithm respectively.

Output: $\mathcal{F}_{\mathsf{pTrain}}$ constructs secret-shares of O and sends the appropriate shares to the servers.

Fig. 2: Ideal Functionality for ML training with data poisoning

Security against $\mathcal{A}^{\mathsf{p}}_{\text{soc}}$. A training protocol Π_{train} is secure against adversary $\mathcal{A}^{\mathsf{p}}_{\text{soc}}$ if: (1) Π_{train} securely realizes functionality $\mathcal{F}_{\mathsf{pTrain}}$ based on Definition 1; and (2) the model trained inside the MPC provides poisoning robustness against data poisoning attacks.

Intuitively, the security definition ensures that $\mathcal{A}^{\mathsf{p}}_{\text{soc}}$ learns no information about the honest owners' inputs when T out of N servers are controlled by the adversary, while the trained model provides poisoning robustness against a subset of t out of m poisoned owners.

B. SafeNet Overview

Given our threat model in Figure 1, existing PPML frameworks provide security against an $\mathcal{A}_{\text{soc}}$ adversary, but they are not designed to handle an adversary of type $\mathcal{A}^{\mathsf{p}}_{\text{soc}}$. We show experimentally in Section IV that PPML frameworks for private training are susceptible to data poisoning attacks. While it would be possible to remedy this by implementing specific poisoning defenses (see Section V-E for a discussion of these approaches), we instead show that it is possible to take advantage of the bounded poisoning capability of $\mathcal{A}^{\mathsf{p}}_{\text{soc}}$ to design a more general and efficient defense. Intuitively, existing approaches train a single model on all local datasets combined, causing the model's training set to have a large fraction of poisoned data (t/m), which is difficult to defend against. Instead, we design SafeNet, a new framework which uses ensemble models to realize our threat model and provide security against $\mathcal{A}^{\mathsf{p}}_{\text{soc}}$. In addition to successfully mitigating data poisoning attacks, SafeNet provides more efficient training than existing PPML and comparable prediction accuracy.

Figure 3 provides an overview of the training and testing phases of SafeNet. SafeNet trains an ensemble E of multiple models in protocol Π_{train}, where each model $\mathcal{M}_k \in E$ is trained locally by the data owner C_k on their dataset. This partitioning prevents poisoned data from contributing to more than t local models. Each data owner samples a local validation dataset and trains the local model $\mathcal{M}_k$ on the remaining data. The local models and validation datasets are secret shared to the outsourced servers. We note that this permits arbitrarily corrupted models, and poisoned validation datasets, but SafeNet's structure still allows it to tolerate these corruptions. In the protocol running inside the MPC, the servers jointly implement a filtering stage for identifying models with low accuracy on the combined validation data (below a threshold ϕ) and excluding them from the ensemble. The output of training is a secret share of each model in the trained ensemble E. Note that exclusion in this case does not mean explicitly removing the model from the ensemble, but, instead, not including the model's prediction in the testing voting procedure. Later in Section III-E3, we show a way to realize the filtering stage without revealing which models is excluded from the final ensemble.

In the prediction phase, SafeNet implements protocol Π_{pred}, which generates the prediction y_k of each shared model $\mathcal{M}_k$ on test input x inside the MPC. The servers jointly perform majority voting to determine the most common predicted class y on input x, using only the models which pass the filtering stage.

Our SafeNet protocol leverages our threat model, which assumes that only a set of at most t out of m data owners are poisoned. This ensures that an adversary only influences a limited set of models in the ensemble, while existing training protocols would train a single poisoned global model. We provide bounds for the exact number of poisoned owners t supported by our ensemble in Theorem 4. Interestingly, the bound depends on the number of data owners m, and the maximum error made by a clean model in the ensemble. The same theorem also lower bounds the probability that the ensemble predicts correctly under data poisoning performed by the t poisoned owners, and we validate experimentally that, indeed, SafeNet provides resilience to stealthy data poisoning attacks, such as backdoor and targeted attacks. Another advantage of SafeNet is that the training time to execute the MPC protocols in the SOC setting is drastically reduced as each $\mathcal{M}_k \in E$ can be trained locally by the respective owner. We detail below the algorithms for training and prediction in SafeNet.

C. SafeNet Training

To train the ensemble in SafeNet, we show our proposed ensemble method in Algorithm 1. We discuss the realization in MPC later in Section III-E. Each owner C_k separates out a subset of its training dataset $D^{\mathsf{v}}_k \in D_k$ and then trains its model $\mathcal{M}_k$ on the remaining dataset $D_k \setminus D^{\mathsf{v}}_k$. The trained model $\mathcal{M}_k$ and validation dataset D^{v}_k is then secret-shared to the servers. The combined validation dataset is denoted as $D_{\text{val}} = \bigcup_{i=1}^{m} D^{\mathsf{v}}_i$. We assume that all users contribute equal-size

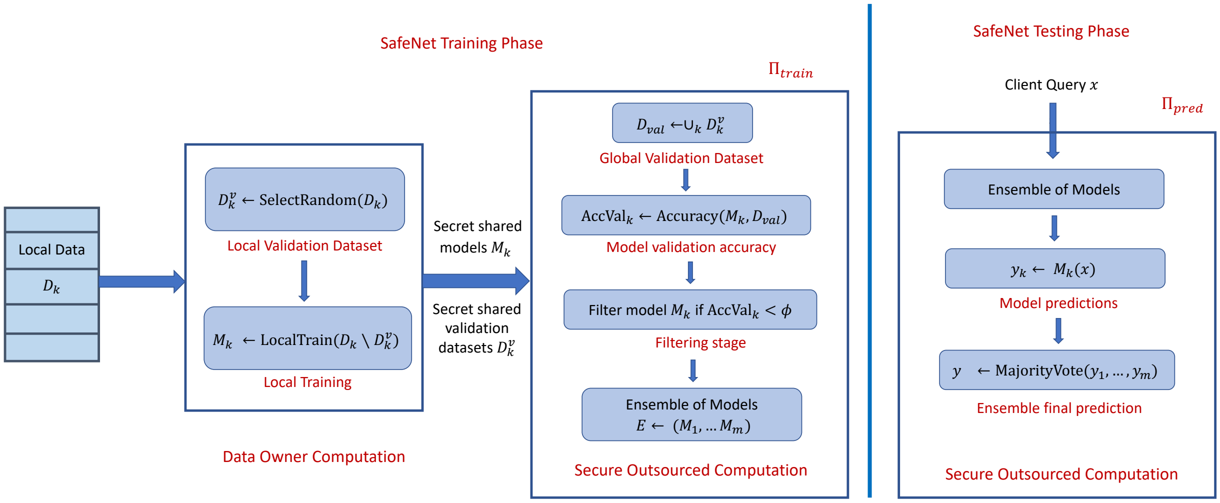


Fig. 3: Overview of the Training and Testing phases of the SafeNet Framework.

validation sets to D_{val}. During the filtering stage inside the MPC, the validation accuracy AccVal of each model is jointly computed on D_{val}. If the resulting accuracy for a model is below threshold ϕ, the model is excluded from the ensemble.

The filtering step is used to separate the models with low accuracy, either contributed by a poisoned owner, or by an owner holding non-representative data for the prediction task. Under the assumption that the majority of owners are honest, it follows that the majority of validation samples are correct. If C_k is honest, then the corresponding $\mathcal{M}_k$ should have a high validation accuracy on D_{val}, as the corresponding predicted outputs would most likely agree with the samples in D_{val}. In contrast, the predictions by a poisoned model $\mathcal{M}_k$ will likely not match the samples in D_{val}. In Section III-D, we compute a lower bound on the size of the validation dataset as a function of the number of poisoned owners t and filtering threshold ϕ, such that all clean models pass the filtering stage with high probability even when a subset of the cross-validation dataset D_{val} is poisoned.

Given protocol Π_{train} that securely realizes Algorithm 1 inside the MPC (which we will define later in Section III-E), we argue security as follows:

Theorem 2. *Protocol Π_{train} is secure against adversary $\mathcal{A}^p_{soc}$ who poisons t out of m data owners and corrupts T out of N servers.*

The proof of the theorem will be given in Appendix B after we introduce all the details of MPC instantiation and how protocol Π_{train} securely realizes $\mathcal{F}_{\mathsf{pTrain}}$ in Section III-E3.

D. Ensemble Robustness Analysis

Lower bounds on accuracy. We provide lower bounds on SafeNet accuracy, assuming that at most t out m models in E are poisoned, and the clean models have independent errors, with maximum error rate $p < 1 - \phi$, where ϕ is the filtering threshold. Crucially, our threat model ensures that

Algorithm 1 SafeNet Training Algorithm

Input: m data owners, each owner C_k's dataset D_k. // Owner's local computation in plaintext format
– For $k \in [1, m]$:
- Separate out D_k^{v} from D_k. Train $\mathcal{M}_k$ on $D_k \setminus D_k^{\mathsf{v}}$.
- Secret-share D_k^{v} and $\mathcal{M}_k$ to servers.
// MPC computation in secret-shared format
– Construct a common validation dataset $D_{\text{val}} = \cup_{i=1}^{m} D_i^{\mathsf{v}}$ and collect ensemble of models $E = \{\mathcal{M}_i\}_{i=1}^{m}$
– Initialize a vector $\mathbf{b}^{\mathsf{val}}$ of zeros and of size m.
– For $k \in [1, m]$: // Ensemble Filtering
- AccVal$_k$ = *Accuracy*$(\mathcal{M}_k, D_{\text{val}})$ // Compute validation accuracy of $\mathcal{M}_k$ over D_{val}.
- If AccVal$_k > \phi$: // Compare against threshold
– Set $\mathbf{b}_k^{\mathsf{val}} = 1$ // Set k^{th} position in $\mathbf{b}^{\mathsf{val}}$ to 1
return E and $\mathbf{b}^{\mathsf{val}}$

there are at least $m - t$ clean models in the ensemble with high probability. Towards this, we first estimate the minimum number of samples required in the validation dataset such that all clean models pass the filtering stage of our training phase. Once at least $m - t$ clean models participate in the final ensemble, we provide lower bounds on the SafeNet accuracy as a function of p and t. We also provide bounds for the number of poisoned models t our framework can tolerate as a function of m, ϕ, and p.

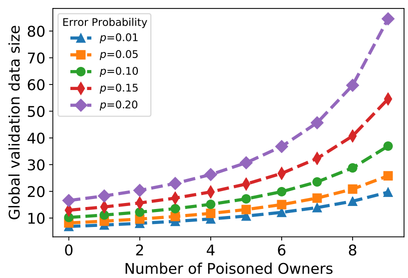
Lemma 3. *Let $\mathcal{A}^p_{soc}$ be an adversary who poisons t out of m data owners and corrupts T out of N servers, and thus contributes t poisoned models to ensemble E, given as output by Algorithm 1. Assume that Π_{train} securely realizes functionality $\mathcal{F}_{\mathsf{pTrain}}$ and every clean model in E makes an error on a clean sample with probability at most $p < 1 - \phi$, where ϕ is the filtering threshold.*

If the validation dataset has at least $\frac{(2+\delta)m\log 1/\epsilon}{\delta^2(m-t)p}$ samples *and $0 \leq t < \frac{m(1-\phi-p)}{(1-p)}$, then all clean models pass the filtering stage of the training phase with probability at least $1-\epsilon$, where $\delta = \frac{(1-\phi)m-t}{(m-t)p} - 1$ and ϵ denotes the failure probability.*

Proof. The proof is deferred to Appendix A. □

As a visual interpretation of Lemma 3, Figure 4 shows the minimum number of samples required in the global validation dataset for varying number of poisoned owners t and error probability p. We set the total models $m = 20$, the failure probability $\epsilon = 0.01$ and the filtering threshold $\phi = 0.3$. The higher the values of t and p, the more samples are required in the validation set. For instance, for $p = 0.20$ and number of poisoned owners $t = 8$, all clean models pass the filtering stage with probability at least 0.99 when the validation set size has at least 60 samples.

Fig. 4: Minimum number of samples in the validation dataset as a function of maximum error probability p and number of poisoned owners t for $m = 20$ data owners. We set the filtering threshold $\phi = 0.03$ and failure probability $\epsilon = 0.01$.



Theorem 4. *Assume that the conditions in Lemma 3 hold against adversary $\mathcal{A}^p_{soc}$ poisoning at most $t < \frac{m}{2}\frac{1-2p}{1-p}$ owners and that the errors made by the clean models are independent. Then E correctly classifies new samples with probability at least $p_c = (1-\epsilon)\left(1 - e^{-\frac{\delta'^2\mu'}{2+\delta'}}\right)$, where $\mu' = (m-t)p$ and $\delta' = \frac{m-2t}{2\mu'} - 1$.*

Proof. The proof is deferred to Appendix A. □

The theorem provides theoretical lower bounds on the test accuracy of our SafeNet framework as a function of the maximum error probability among the clean models. Later in Section V-B, we provide details on how to set these parameters in practice and compare these theoretical bounds with our experimental results.

Certified Prediction. Theorem 4 demonstrates that SafeNet's accuracy on in-distribution data will not be compromised by poisoning. However, we now show that we can also certify robustness to poisoning on a per-sample basis for arbitrary points, inspired by certified robustness techniques for adversarial example robustness [24]. In particular, Algorithm 2 describes a method for certified prediction against poisoning, returning the most common class y predicted by the ensemble on a test point x, as well as a bound on the number of poisoning owners t which would be required to modify the predicted class.

Algorithm 2 Certified Prediction CERTPRED (E, x)

Input: m data owners; Ensemble of models $E = \{\mathcal{M}_i\}_{i=1}^m$; Testing point x.
PREDS $= E(x) = \{\mathcal{M}_i(x)\}_{i=1}^m$
$y, c_y =$ MOSTCOMMONPRED(PREDS) // get most common predicted class with count
$y', c_{y'} =$ SECONDMOSTCOMMONPRED(PREDS) // get second most common predicted class with count
$t = \lceil (c_y - c_{y'})/2 \rceil - 1$
return y, t

Theorem 5. *Let E be an ensemble of models trained on datasets $D = \{D_1, \ldots, D_m\}$. Assume that on an input x, the ensemble generates prediction $y = E(x)$ and Algorithm 2 outputs (y, t). Moreover, assuming an adversary $\mathcal{A}^p_{soc}$ who poisons at most t data owners, the resulting E' trained on poisoned data D' generates the same prediction on x as E: $E'(x) = y$.*

Proof. If an adversary's goal were to cause y' to be predicted on input x, their most efficient strategy is to flip y predictions to y'. If y were the ensemble prediction, it must have at least $\lfloor \frac{c_y + c_{y'}}{2} \rfloor$ model predictions, and the second most common prediction y' would have at most $\lfloor \frac{c_y + c_{y'}}{2} \rfloor$ model predictions. Corrupting these predictions then requires flipping at least $(c_y - c_{y'})/2$ predictions from y to y'. Overall, this requires at least $\lceil (c_y - c_{y'})/2 \rceil$ poisoned data owners. Thus, an adversary poisoning at most $t = \lceil (c_y - c_{y'})/2 \rceil - 1$ data owners still generates the same prediction y on x. □

E. Realization in MPC

To instantiate SafeNet in MPC, we first describe the required MPC building blocks, and then provide the SafeNet training and secure prediction protocols.

1) MPC Building Blocks: The notation $[\![x]\!]$ denotes a given value x secret-shared among the servers. The exact structure of secret sharing is dependent on the particular instantiation of the underlying MPC framework[30], [4], [39], [19], [20], [13], [69]. We assume each value and its respective secret shares to be elements over an arithmetic ring $\mathbb{Z}_{2^\ell}$. All multiplication and addition operations are carried out over $\mathbb{Z}_{2^\ell}$.

We express each of our building blocks in the form of an ideal functionality and its corresponding protocol. An ideal functionality can be viewed as an oracle, which takes input from the parties, applies a predefined function f on the inputs and returns the output back to the parties. The inputs and outputs can be in clear or in $[\![\cdot]\!]$-shared format depending on the definition of the functionality. These ideal functionalities are realized using secure protocols depending on the specific instantiation of the MPC framework agreed upon by the parties. Below are the required building blocks:

Secure Input Sharing. Ideal Functionality $\mathcal{F}_{\mathsf{shr}}$ takes as input a value x from a party who wants to generate a $[\![\cdot]\!]$-sharing of x, while other parties input $\perp$ to the functionality. $\mathcal{F}_{\mathsf{shr}}$

generates a $[\![\cdot]\!]$-sharing of x and sends the appropriate shares to the parties. We use Π_{sh} to denote the protocol that realizes this functionality securely.

Secure Addition. Given $[\![\cdot]\!]$-shares of x and y, secure addition is realized by parties locally adding their shares $[\![z]\!] = [\![x]\!] + [\![y]\!]$, where $z = x + y$.

Secure Multiplication:. Functionality $\mathcal{F}_{\mathsf{mult}}$ takes as input $[\![\cdot]\!]$-shares of values x and y, creates $[\![\cdot]\!]$-shares of $z = xy$ and sends the shares of z to the parties. Π_{mult} denotes the protocol to securely realize $\mathcal{F}_{\mathsf{mult}}$.

Secure Output Reconstruction. $\mathcal{F}_{\mathsf{op}}$ functionality takes as input $[\![\cdot]\!]$-shares of a value x from the parties and a commonly agreed upon party id pid in clear. On receiving the shares and pid, $\mathcal{F}_{\mathsf{op}}$ reconstructs x and sends it to the party associated to pid.

Secure Comparison. $\mathcal{F}_{\mathsf{comp}}$ functionality takes as input a value a in $[\![\cdot]\!]$-shared format. $\mathcal{F}_{\mathsf{comp}}$ initializes a bit $b = 0$, sets $b = 1$ if $a > 0$ and outputs it in $[\![\cdot]\!]$-shared format. Protocol Π_{comp} is used to securely realize $\mathcal{F}_{\mathsf{comp}}$.

Secure Zero-Vector. $\mathcal{F}_{\mathsf{zvec}}$ functionality takes as input a value L in clear from the parties. $\mathcal{F}_{\mathsf{zvec}}$ constructs a vector $\mathbf{z}$ of all zeros of size L and outputs $[\![\cdot]\!]$-shares of $\mathbf{z}$. Π_{zvec} denotes the protocol that securely realizes $\mathcal{F}_{\mathsf{zvec}}$.

Secure Argmax. $\mathcal{F}_{\mathsf{amax}}$ functionality takes as input a vector $\mathbf{x}$ in $[\![\cdot]\!]$-shared format and outputs $[\![\cdot]\!]$-shares of a value $\textsc{op}$, where $\textsc{op}$ denotes the index of the max element in vector $\mathbf{x}$. Π_{amx} denotes the protocol that securely realizes $\mathcal{F}_{\mathsf{amax}}$.

2) ML Building Blocks: We introduce several building blocks required for private ML training, implemented by existing MPC frameworks [64], [62], [13], [69], [76]:

Secure Model Prediction. $\mathcal{F}_{\mathcal{M}\mathsf{pred}}$ functionality takes as input a trained model $\mathcal{M}$ and a feature vector $\mathbf{x}$ in $[\![\cdot]\!]$-shared format. $\mathcal{F}_{\mathcal{M}\mathsf{pred}}$ then computes prediction $\mathbf{Preds} = \mathcal{M}(\mathbf{x})$ in one-hot vector format and outputs $[\![\cdot]\!]$-shares of the same. $\Pi_{\mathcal{M}\mathsf{pred}}$ denotes the protocol which securely realizes functionality $\mathcal{F}_{\mathcal{M}\mathsf{pred}}$.

Secure Accuracy. $\mathcal{F}_{\mathsf{acc}}$ functionality takes as input two equal length vectors $\mathbf{y}_{pred}$ and $\mathbf{y}$ in $[\![\cdot]\!]$-shared format. $\mathcal{F}_{\mathsf{acc}}$ then computes the total number matches (element-wise) between the two vectors and outputs $\frac{\#\ \text{matches}}{|\mathbf{y}|}$ in $[\![\cdot]\!]$-shared format. Π_{acc} denotes the protocol which securely realizes this functionality.

3) Protocols: We propose two protocols to realize our SafeNet framework in the SOC setting. The first protocol Π_{train} describes the SafeNet training phase where given $[\![\cdot]\!]$-shares of dataset D_k^{v} and model $\mathcal{M}_k$, with respect to each owner C_k, Π_{train} outputs $[\![\cdot]\!]$-shares of an ensemble E of m models and vector $\mathbf{b}^{\mathsf{val}}$. The second protocol Π_{pred} describes the prediction phase of SafeNet, which given $[\![\cdot]\!]$-shares of a client's query predicts its output label. The detailed description for each protocol is as follows:

SafeNet Training. We follow the notation from Algorithm 1. Our goal is for training protocol Π_{train} given in Figure 5 to securely realize functionality $\mathcal{F}_{\mathsf{pTrain}}$ (Figure 2), where the inputs to $\mathcal{F}_{\mathsf{pTrain}}$ are $[\![\cdot]\!]$-shares of $D_k = D_k^{\mathsf{v}}$ and $a_k = \mathcal{M}_k$, and the corresponding outputs are $[\![\cdot]\!]$-shares of $O = E$ and $\mathbf{b}^{\mathsf{val}}$. Given the inputs to Π_{train}, the servers first construct a common validation dataset $[\![D_{\text{val}}]\!] = \cup_{k=1}^{m}[\![D_k^{\mathsf{v}}]\!]$ and an ensemble of models $[\![E]\!] = \{[\![\mathcal{M}_k]\!]\}_{k=1}^{m}$. Then for each model $\mathcal{M}_k \in E$, the servers compute the validation accuracy $[\![\text{AccVal}_k]\!]$. The output $[\![\text{AccVal}_k]\!]$ is compared with a pre-agreed threshold ϕ to obtain a $[\![\cdot]\!]$-sharing of $\mathbf{b}_k^{\mathsf{val}}$, where $\mathbf{b}_k^{\mathsf{val}} = 1$ if $\text{AccVal}_k > \phi$. After execution of Π_{train} protocol, servers obtain $[\![\cdot]\!]$-shares of ensemble E and vector $\mathbf{b}^{\mathsf{val}}$.

Protocol Π_{train}

Input: $[\![\cdot]\!]$-shares of each owner C_k's validation dataset D_k^{v} and local model $\mathcal{M}_k$.

Protocol Steps: The servers perform the following:

- Construct $[\![\cdot]\!]$-shares of ensemble $E = \{\mathcal{M}_k\}_{k=1}^{m}$ and validation dataset $D_{\text{val}} = \cup_{k=1}^{m} D_k^{\mathsf{v}}$.
- Execute Π_{zvec} with m as the input and obtain $[\![\cdot]\!]$-shares of a vector $\mathbf{b}^{\mathsf{val}}$.
- For $k \in [1, m]$:
  - Execute $\Pi_{\mathcal{M}\mathsf{pred}}$ with inputs as $[\![\mathcal{M}_k]\!]$ and $[\![D_{\text{val}}]\!]$ and obtain $[\![\text{PREDS}_k]\!]$, where $\textsc{Preds}_k = \mathcal{M}_k(D_{\text{val}})$
  - Execute Π_{acc} with inputs as $[\![\textsc{Preds}_k]\!]$ and $[\![\mathbf{y}_{D_{\text{val}}}]\!]$ and obtain $[\![\text{AccVal}_k]\!]$ as the output.
  - Locally subtract $[\![\cdot]\!]$-shares of AccVal_k with ϕ to obtain $[\![\text{AccVal}_k - \phi]\!]$.
  - Execute Π_{comp} with input as $[\![\text{AccVal}_k - \phi]\!]$ and obtain $[\![b']\!]$, where $b' = 1$ iff $\text{AccVal}_k > \phi$. Set the k^{th} position in $[\![\mathbf{b}^{\mathsf{val}}]\!]$ as $[\![\mathbf{b}_k^{\mathsf{val}}]\!] = [\![b']\!]$

Output: $[\![\cdot]\!]$-shares of $\mathbf{b}^{\mathsf{val}}$ and ensemble E.

Fig. 5: SafeNet Training Protocol

The security proof of Π_{train} protocol as stated in Theorem 2 in Section III-C is given in Appendix B.

Protocol Π_{pred}

Input: $[\![\cdot]\!]$-shares of vector $\mathbf{b}^{\mathsf{val}}$ and ensemble E among the servers. Client $[\![\cdot]\!]$-shares query $\mathbf{x}$ to the servers.

Protocol Steps: The servers perform the following:

- Execute Π_{zvec} protocol with L as the input, where L denotes the number of distinct class labels and obtain $[\![\cdot]\!]$-shares of $\mathbf{z}$.
- For each $\mathcal{M}_k \in E$:
  - Execute $\Pi_{\mathcal{M}\mathsf{pred}}$ with inputs as $[\![\mathcal{M}_k]\!]$ and $[\![\mathbf{x}]\!]$. Obtain $[\![\mathbf{Preds}]\!]$, where $\mathbf{Preds} = \mathcal{M}_k(\mathbf{x})$.
  - Execute Π_{mult} to multiply $\mathbf{b}_k^{\mathsf{val}}$ to each element of vector $\mathbf{Preds}$.
  - Locally add $[\![\mathbf{z}]\!] = [\![\mathbf{z}]\!] + [\![\mathbf{Preds}]\!]$ to update $\mathbf{z}$.
- Execute Π_{amx} protocol with input as $[\![\mathbf{z}]\!]$ and obtain $[\![\textsc{op}]\!]$ as the output.

Output: $[\![\cdot]\!]$-shares of $\textsc{op}$

Fig. 6: SafeNet Prediction Protocol

SafeNet Prediction. Functionality $\mathcal{F}_{\mathsf{pred}}$ takes as input party id cid, $[\![\cdot]\!]$-shares of client query $\mathbf{x}$, vector $\mathbf{b}^{\mathsf{val}}$ and ensemble $E = \{[\![\mathcal{M}_k]\!]\}_{k=1}^{m}$ and outputs a value $\textsc{op}$, the predicted class label by ensemble E on query $\mathbf{x}$.

Protocol Π_{pred} realizes $\mathcal{F}_{\mathsf{pred}}$ as follows: Given $[\![\cdot]\!]$-shares of $\mathbf{x}$, $\mathbf{b}^{\mathsf{val}}$ and ensemble E, the servers initialize a vector $\mathbf{z}$ of all zeros of size L. For each model $\mathcal{M}_k$ in the ensemble E, the servers compute $[\![\cdot]\!]$-shares of the prediction $\mathbf{Preds} = \mathcal{M}_k(\mathbf{x})$ in one-hot format. The element $\mathbf{b}_k^{\mathsf{val}}$ in vector $\mathbf{b}^{\mathsf{val}}$ is multiplied to each element in vector $\mathbf{Preds}$. The $[\![\mathbf{Preds}]\!]$ vector is added to $[\![\mathbf{z}]\!]$ to update the model's vote towards the final prediction. If $\mathbf{b}_k^{\mathsf{val}} = 0$, then after multiplication vector $\mathbf{Preds}$ is a vector of zeros and does not contribute in the voting process towards the final prediction. The servers then compute the argmax of vector $[\![\mathbf{z}]\!]$ and receive output $[\![\textsc{op}]\!]$ from Π_{amx}, where $\textsc{op}$ denotes the predicted class label by the ensemble. The appropriate $[\![\cdot]\!]$-shares of $\textsc{op}$ is forwarded to the client for reconstruction.

Theorem 6. *Protocol Π_{pred} is secure against adversary $\mathcal{A}_{soc}^{p}$ who poisons t out of m data owners and corrupts T out of N servers.*

Proof. The proof is deferred to Appendix B. □

IV. Evaluation

A. Experimental Setup

We build a functional code on top of the MP-SPDZ library [49][1] to assess the impact of data poisoning attacks on the training phase of PPML frameworks. We consider four different MPC settings, all available in the MP-SPDZ library: i) two-party with one semi-honest corruption (2PC) based on [30], [25]; ii) three-party with one semi-honest corruption (3PC) based on Araki et al. [4] with optimizations by [62], [27]; iii) three-party with one malicious corruption based on Dalskov et al. [26]; and iv) four-party with one malicious corruption (4PC), also based on [26].

In all the PPML frameworks, the data owners secret-share their training datasets to the servers and a single ML model is trained on the combined dataset. Typically, real number arithmetic is emulated by using 32-bit fixed-point representation of fractional numbers. Each fractional number $x \in \mathbb{Z}_{2^\ell}$ is represented as $\lfloor x \cdot 2^f \rceil$, where ℓ and f denote the ring size and precision, respectively. We set $\ell = 64$ and $f = 16$. Probabilistic truncation proposed by Dalskov et al. [27], [26] is applied after every multiplication.

We perform our experiments over a LAN network on a 32-core server with 192GB of memory allowing up to 20 threads to be run in parallel.

B. Metrics

We use the following metrics to compare SafeNet with existing PPML framework:

Training Time. This is the time taken to privately train a model inside the MPC (protocol Π_{train}). As is standard practice [64], [62], [19], [20], [13], [26], this excludes the time taken by the data owners to secret-share their datasets and models to the servers as it is a one-time setup phase.

Communication Complexity. Amount of data exchanged between the outsourced servers during the privacy-preserving execution of the training phase.

Test Accuracy. Percentage of test samples that the trained model correctly predicts.

Attack Success Rate. Percentage of target samples that were misclassified as the label of attacker's choice.

Robustness against worst-case adversary. We measure the resilience of SafeNet at a certain corruption level c against a powerful, worst-case adversary. For each test sample, this adversary can select any subset of c owners, arbitrarily modifying the model (in a model poisoning attack) to change the test sample's classification. This is the same adversary considered in Algorithm 2 and by Theorem 5, any model which is robust against this attack has a provably certified prediction. We measure the error rate on testing samples for this worst-case adversarial model.

C. Datasets and Models

We give a brief descriptions of the datasets and models used for our experiments and provide a detailed description of our setup in Appendix E, Table VI.

Digit 1/7. We train a logistic regression model on a subset of MNIST [54], using only digits 1 and 7.

MNIST. We extend the logistic regression model to multi-class classification on the entire MNIST dataset.

Adult. The Adult dataset [32] is for a binary classification problem to predict if a person's annual income is above 50K. We train a neural network with one hidden layer of size 10 nodes using ReLU as activation.

Fashion. We train several neural networks on the Fashion-MNIST dataset [78] with one to three hidden layers. The Fashion dataset is a 10-class classification problem with 784 features representing various garments. All hidden layers have 128 nodes and ReLU activations, except the output layer using softmax.

In the MPC library implementation, the sigmoid function for computing the output probabilities is replaced with a three-part approximation [64], [19], [26]. Instead, in SafeNet, models are trained locally using the original sigmoid function. We implement the softmax for multi-classification using the method of Aly et al. [2]. All datasets are equally split across 20 data owners, with the exception of the Fashion dataset split across 10 owners. Each owner separates out 10% of its local training data D_j as the validation dataset D_j^{v}. All models are trained using mini-batch stochastic gradient descent.

D. Implementation of Poisoning Attacks

Backdoor Attacks. We use the BadNets attack by Gu et al. [40], in which the poisoned owners inject a backdoor into the model to change the model's prediction from source label y_s to target label y_t. For instance, in an image dataset, a backdoor

[1] https://github.com/data61/MP-SPDZ

might set a few pixels in the corner of the image to white. The BadNets attack strategy simply identifies a set of k target samples $\{x_i^t\}_{i=1}^k$ with true label y_s, and creates backdoored samples with target label y_t. We use $k = 100$ samples, which is sufficient to poison all models.

In the PPML framework the poisoned owners create the poisoned dataset D_j^* by adding k poisoned samples and secret-sharing them as part of the training dataset to the MPC. The framework then trains the ML model on the combined dataset submitted by both the honest and poisoned owners.

In SafeNet framework, the poisoned owners add k backdoored samples to their dataset D_j and train their local models $\mathcal{M}_j^*$ on the combined clean and poisoned data. A model trained only on poisoned data will have accuracy lower than the filtering threshold, and therefore we have to use the clean samples in training. The corrupt owners then secret-share both the model $\mathcal{M}_j^*$ and validation set D_j^{v} selected at random from D_j to the MPC.

Targeted Attacks. We select k targeted samples, and change their labels in training to a target label y_t different from the original label. The models are trained to simultaneously minimize both the training and the adversarial loss. This strategy has also been used to construct poisoned models by prior work [51], and can be viewed as an unrestricted version of the state-of-the-art Witches' Brew targeted attack (which requires clean-label poisoned samples) [36].

The next question to address is which samples to target as part of the attack. We use two strategies to generate $k = 100$ target samples, based on an ML model trained by the adversary over the test data. In the first strategy, called TGT-Top, the adversary chooses the top k samples for the targeted attack based on the model's output probability of the true label. These are high-confidence samples, predicted by the model with the highest probability, and would be more difficult to poison. Then for each sample x in the generated set, the adversary replaces the true label of x to the second predicted label. The second strategy, named TGT-Foot, uses instead the lowest-confidence k samples, which are easier to attack as the model has higher error on them. We compare these two strategies for target selection.

The difference between targeted and backdoor attacks is that targeted attacks do not require the addition of a backdoor trigger to training or testing samples, as needed in a backdoor attack. However, the impact of the backdoor attack is larger. Targeted attacks change the prediction on a small set of testing samples (which are selected in advance before training the model), while the backdoor attack generalizes to any testing samples including the backdoor pattern.

E. Evaluation on Logistic Regression

We consider here the DIGIT 1/7 dataset, for which we first evaluate the computational costs and then the poisoning attack success, for both traditional PPML and our newly proposed SafeNet framework.

We perform our experiments over four underlying MPC frameworks, with both semi-honest and malicious adversaries. Table I provides a detailed analysis of the training time and communication complexity for both existing PPML and SafeNet frameworks. Note that the training time and communication cost for the PPML frameworks is reported per epoch times the number of epochs in training. The number of epochs is a configurable hyper-parameter, but usually at least 10 epochs are required. On the other hand, the training time and communication reported for our SafeNet framework is for the end-to-end execution inside the MPC, independent of the number of epochs. We observe large improvements of SafeNet over the existing PPML frameworks. For instance, in the semi-honest two-party setting, SafeNet achieves $30\times$ and $17\times$ improvement in running time and communication complexity, respectively, for $n = 10$ epochs. This is expected because SafeNet performs local model training, which is an expensive phase in the MPC.

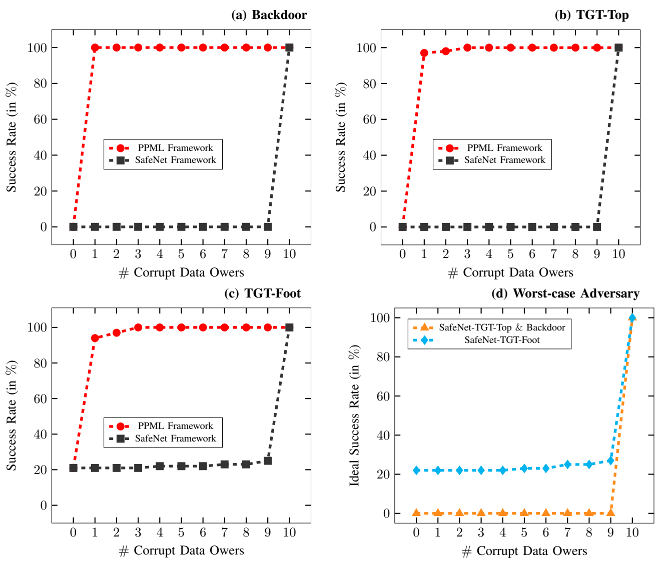
TABLE I: Training Time (in seconds) and Communication (in GB) of existing PPML and SafeNet framework for a logistic regression model over several MPC settings over a LAN network. n denotes the number of epochs required for training the logistic regression model in the PPML framework. The time and communication reported for SafeNet is for end-to-end execution.

MPC	Setting	Framework	Training (s)	Comm. (GB)
2PC	Semi-Honest [30]	PPML	n×151.84	n×65.64
		SafeNet	57.41	38.03
3PC	Semi-Honest [4]	PPML	n×2.63	n×0.35
		SafeNet	0.54	0.15
	Malicious [26]	PPML	n×32.54	n× 2.32
		SafeNet	9.44	1.47
4PC	Malicious [26]	PPML	n×5.28	n×0.66
		SafeNet	1.09	0.28

To mount the backdoor attack, the backdoor pattern sets the top left pixel value to white (a value of 1). We set the original class as $y_s = 1$ and target class as $y_t = 7$. Figure 7 (a) shows the success rate for the 3PC PPML and SafeNet frameworks by varying the number of poisoned owners between 0 and 10. We tested with all four PPML settings and the results are similar. We observe that by poisoning data of a single owner, the adversary is successfully able to introduce a backdoor in the PPML framework. The model in the PPML framework predicts all $k = 100$ target samples as y_t, achieving 100% adversarial success rate. In contrast, SafeNet is successfully able to defend against the backdoor attack, and provides 0% attack success rate up to 9 owners with poisoned data. The test accuracy on clean data for both frameworks is high at around 98.98% even after increasing the number of poisoned owners to 10.

We observe in Figure 7 (b) that for the TGT-Top targeted attack, a single owner poisoning is able to successfully misclassify 98% of the target samples in the PPML framework. As a consequence, the test accuracy of the model drops by $\approx 10\%$. In contrast, our SafeNet framework works as intended even at high levels of poisoning. For the TGT-Foot attack in Figure 7 (c), the test accuracy of the 3PC PPML framework drops by $\approx 5\%$. The attack success rate is 94% for the 3PC PPML, which is decreased to 21% by SafeNet, in presence of

Fig. 7: Logistic regression attack success rate on the Digit-1/7 dataset for PPML and SafeNet frameworks in the 3PC setting, for varying poisoned owners launching Backdoor and Targeted attacks. Plot (a) gives the success rate for the BadNets attack, while plots (b) and (c) show the success rates for the TGT-Top and TGT-Foot targeted attacks. Plot (d) provides the worst-case adversarial success when the set of poisoned owners can change per sample. Lower attack success result in increased robustness. SafeNet achieves much higher level of robustness than existing PPML under both attacks.



a single poisoned owner. The accuracy drop and success rate vary across the two strategies because of the choice of the target samples. In TGT-Foot, the models have low confidence on the target samples, which introduces errors even without poisoning, making the attack succeed with slightly higher rate in SafeNet. Still, SafeNet provides resilience against both TGT-Top and TGT-Foot for up to 9 out of 20 poisoned owners.

Worst-case Robustness. Figure 7 (d) shows the worst-case attack success in SafeNet, by varying the number of poisoned owners $c \in [1, 10]$ and allowing the attacker to poison a different set of c owners for each testing sample (i.e., the adversarial model considered in Algorithm 2 for which we can certify predictions). Interestingly, SafeNet's accuracy is similar to that achieved under our backdoor and targeted attacks, even for this worst-case adversarial scenario. Based on these results we conclude that: (1) the backdoor and targeted attacks we choose to implement are as strong as the worst-case adversarial attack, in which the set of poisoned owners is selected per sample; (2) SafeNet provides certified robustness up to 9 out of 20 poisoned owners even under this powerful threat scenario.

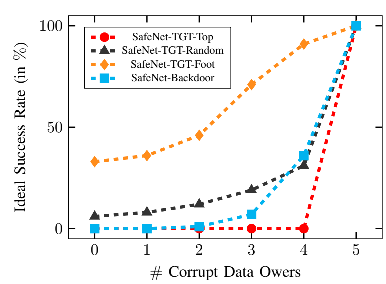
Multiclass Classification. We also test both frameworks in the multiclass classification setting for both Backdoor and Targeted attacks on MNIST dataset and observe similar large improvements. For instance, in the semi-honest 3PC setting, we get $240\times$ and $268\times$ improvement, respectively, in running time and communication complexity for $n = 10$ epochs while the success rate in the worst-case adversarial scenario not exceeding 50% with 9 out of 20 owners being poisoned. This experiment shows that the robust accuracy property of our framework translates seamlessly even for the case of a multi-class classification problem. The details of the experiment are deferred to Appendix D.

F. Evaluation on Deep Learning Models

We evaluate neural network training for PPML and SafeNet frameworks on the Adult and Fashion datasets. We provide experiments on a three hidden layer neural network on Fashion in this section and include additional experiments in Appendix D.

For the BadNets backdoor attack we set the true label y_s as a 'T-Shirt' and the target label y_t is a 'Trouser'. We test the effect of both TGT-Top and TGT-Foot targeted attacks in presence of a single poisoned owner. Table II provides a detailed analysis of the training time, communication, test accuracy and success rate for both frameworks. In the 4PC setting, SafeNet has $39\times$ and $36\times$ improvement in training time and communication complexity over the PPML framework, for $n = 10$ epochs. We also evaluate another variant of targeted attack called TGT-Random, where we randomly sample $k = 100$ target samples from the test data. Figure 8 provides the worst-case adversarial success of SafeNet against these attacks. We observe that SafeNet provides certified robustness for TGT-Random and TGT-Top up to 4 out of 10 poisoned onwers, while the adversary is able to misclassify more target samples in the TGT-Foot attack. The reason is that the k selected target samples have lowest confidence and models in the ensemble are likely to be in disagreement on their prediction.

Fig. 8: Worst-case adversarial success against targeted and backdoor attacks of a three-layer neural network trained on Fashion in SafeNet. The adversary can change the set of c poisoned owners per sample. SafeNet achieves robustness on the backdoor, TGT-Top and TGT-Random attacks, up to 4 poisoned owners out of 10. The TGT-Foot attack targeting low-confidence samples has higher success.



V. Discussion and Extensions

We showed that SafeNet successfully mitigates a variety of data poisoning attacks. We now discuss other aspects of our framework such as scalability and modularity, parameter selection in practice, data distribution similarity among owners, computational capacity, and comparison against other mitigation strategies.

A. SafeNet's Scalability and Modularity

Scalability. The training and prediction times of SafeNet inside the MPC depend on the number of models in the

TABLE II: Time (in seconds) and Communication (in Giga-Bytes) over a LAN network for PPML and SafeNet framework training a Neural Network model with 3 hidden layers over Fashion dataset. n denotes the number of epochs used to train the NN model in the PPML framework. The time and communication reported for SafeNet is for end-to-end execution. Test Accuracy and Success Rate is given for the case when a single owner is corrupt.

MPC	Setting	Framework	Training Time (s)	Communication (GB)	Backdoor Attack		Targeted Attack		
					Test Accuracy	Success Rate	Test Accuracy	Success Rate-Top	Success Rate-Foot
3PC [4]	Semi-Honest	PPML	n $\times$ 565.45	n $\times$ 154.79	84.07%	100%	82.27%	100%	100%
		SafeNet	156.53	41.39	84.36%	0%	84.48%	0%	32%
4PC [26]	Malicious	PPML	n $\times$ 1392.46	n $\times$ 280.32	84.12%	100%	82.34%	100%	100%
		SafeNet	356.26	76.43	84.36%	0%	84.54%	0%	32%

ensemble and the size of the validation dataset. The training time increases linearly with the fraction of training data used for validation and the number of models in the ensemble. Similarly, the prediction phase of SafeNet has both runtime and communication scaling linearly with the number of models in the ensemble. For instance, for the Fashion dataset setup, our SafeNet framework takes on average 26 milliseconds to perform a single secure prediction, while the existing PPML framework takes on average 3.5 milliseconds for the same task. However, we believe this is a reasonable cost for many applications, as SafeNet also has significant training time improvements. As the number of data owners grows, SafeNet also tolerates a larger number of poisoned owners.

Transfer Learning. It is possible to further reduce SafeNet's prediction time significantly in transfer learning settings [52], [31]. In this case, all data owners start with a common pre-trained model, and fine tune only its last layer using their local data. The fine-tuned models are secret shared to the MPC. We can then modify the prediction phase of SafeNet to reduce its prediction time and cost considerably. The crucial observation is that all fine tuned models differ only in the weights associated to the last layer. Consequently, given a prediction query in secret-shared format, we run the pre-trained model until its penultimate layer and then compute the m final layers of the fine tuned models, inside the MPC. The detailed description of the modified SafeNet prediction algorithm is given in Appendix C. We use the same Fashion dataset setup as earlier, with $m = 10$ data owners, and observe that for each secure prediction, SafeNet is now only $1.62\times$ slower and communicates $1.26\times$ more on average than the PPML framework. Note that we achieve the same robustness guarantee against poisoning as before, assuming that the pre-trained model is not poisoned.

Modularity. Another key advantage of SafeNet is that it can use any MPC protocol as a backend, as long as it implements standard ML operations. We demonstrated this by performing experiments with both malicious and semi-honest security for four different MPC settings. As a consequence, advances in ML inference with MPC will improve SafeNet's runtime. SafeNet can also use any model type implementable in MPC; if more accurate models are designed, this will lead to improved robustness and accuracy. This can also be used to, for example, ensure training data privacy if a local model is trained with privacy-preserving techniques.

B. *Instantiating SafeNet in Practice*

In this section we discuss how SafeNet can be instantiated in practice. There are two aspects the data owners need to agree upon before instantiating SafeNet: i) The MPC framework used for secure training and prediction phase and ii) the parameters in Theorem 4 to achieve poisoning robustness. The MPC framework is agreed upon by choosing the total number of outsourced servers N participating in the MPC, the number of corrupted servers T tolerated in the MPC and the nature of the adversary (semi-honest or malicious). To achieve poisoning robustness, the owners agree upon a filtering threshold ϕ and the number of poisoned owners t that can be tolerated. Once these parameters are chosen the maximum allowed error probability of the local models trained by the honest owners based on Lemma 3 and Theorem 4, can be computed as $p < \min(\frac{m(1-\phi)-t}{m-t}, \frac{m-2t}{2(m-t)})$, where m denotes the total number of data owners. Given the upper bound on the error probability p, each honest owner trains its local model while satisfying the above constraint, i.e., each honest owner has its local model's accuracy at least $(1-p)$. Additionally, also the size of the global validation dataset $|D_{\text{val}}|$ can be computed based on Lemma 3.

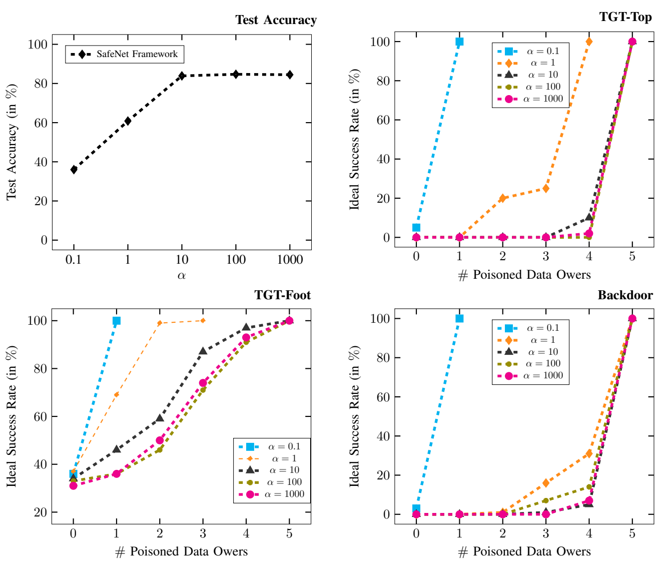
We provide a concrete example on parameter selection: We instantiate our Fashion dataset setup, with $m = 10$ data owners participating in SafeNet. For the MPC framework we choose a three-party setting ($N = 3$ servers), tolerating $T = 1$ corruption. For poisoning robustness, we set $\phi = 0.3$ and the number of poisoned owners that can be tolerated as $t = 2$. This gives us the upper bound on max error probability as $p < 0.375$. Also the size of the global validation dataset is $|D_{\text{val}}| > 92$ samples, i.e., each data owner contributes 10 cross-validation samples each such that the constrained is satisfied. With this instantiation, we observe that none of the honest models gets filtered in the filtering stage and the attack success rate of the adversary for backdoor attacks remains the same after poisoning $t = 2$ owners. In fact, we observe that our framework achieves the same result even if the adversary ends up poisoning 3 owners instead. Thus, in practice SafeNet is able tolerate more poisoning than our analysis. Note that, we recommend the filtering threshold ϕ to be a small value as setting ϕ to be large puts stricter constraints on p and $|D_{\text{val}}|$, leading to honest local models unable to satisfy them and consequently getting filtered from the ensemble, a strategy that might be employed by the adversary.

C. Distribution Similarity

SafeNet includes an ensemble of models, each trained on a single owner's dataset. As a result, SafeNet naturally performs best when the owners' distributions are similar (independent and identically distributed – *iid*). Here, we evaluate the performance of SafeNet when the data distributions among owners are non-iid. Towards this goal, we vary the owners' distribution similarity by partitioning the training data using the Dirichlet distribution on the class labels [42]. To generate a population of non-identical owners, we sample $q \sim Dir(\alpha p)$ from a Dirichlet distribution, where p characterizes a prior class distribution over all distinct classes, and $\alpha > 0$ is a concentration parameter which controls the degree of similarity between owners. As $\alpha \to \infty$, all owners have identical distributions, whereas as $\alpha \to 0$, each owner holds samples of only one randomly chosen class. All our previous experiments were conducted with parameter α set to 100.

We test the SafeNet framework by varying α to manipulate the degree of data similarity among the owners. The experiments are performed with the same neural network architecture from Section IV-F on the Fashion dataset. Figure 9 gives a comprehensive view of the variation in test accuracy and attack success rate for backdoor and targeted attacks over several values of α.

Fig. 9: Test Accuracy and Worst-case Adversarial Success in a three layer neural network model trained on Fashion dataset using SafeNet for varying data distributions. Parameter α dictates the similarity of distributions between the owners. Higher values of α denote greater similarity in data distributions among the owners and results in increased SafeNet robustness.



We observe that as α decreases, i.e., the underlying data distribution of the owners becomes more non-iid, the test accuracy of SafeNet starts to drop. This is expected as there will be less agreement between the different models, and the majority vote will have a larger chance of errors. In such cases it is easier for the adversary to launch an attack as there is rarely any agreement among the models in the ensemble, and the final output is swayed towards the target label of attackers' choice. Figure 9 shows that for both targeted and backdoor attacks, SafeNet holds up well until α reaches extremely small values ($\alpha = 0.1$), at which point we observe the robustness break down.

However, the design of SafeNet allows us to detect difference in owners' distributions at early stages of our framework. For instance, we experiment for $\alpha = 0.1$ and observe that the average AccVal accuracy of the models is 17%. Such low accuracies for most of the models in the ensemble indicate non-identical distributions and we recommend not to use SafeNet in such cases.

D. Owner's Computational Capacity

In our current framework, the data owners train local models on their individual datasets and secret-share these models (along with the cross-validation dataset) to the MPC. In order to train these local models, we assume data owners possess the required computational capacity. In situations when a subset of owners do not have access to sufficient computational resources, we can use the PPML framework to train the local models. In this design, a computationally restricted data owner secret-shares his training dataset to the outsourced MPC servers, which jointly train the owner's local model.

We provide the details of our modified training algorithm for SafeNet in Appendix C. This modification allows a computationally restricted owner to take part in SafeNet albeit increasing the training time and communication complexity of the training phase of SafeNet. For instance, we instantiate our Fashion dataset setup in the 3PC setting and assume 2 out of 10 data owners are computationally restricted. We observe SafeNet still runs $1.82\times$ faster and requires $3.53\times$ less communication compared to the existing PPML framework, while achieving much stronger robustness against poisoning.

E. Comparing to poisoning defenses

Defending against poisoning attacks is an active area of research, but defenses tend to be heuristic and specific to attacks or domains. Many defenses for backdoor poisoning attacks exist [58], [74], [21], [77], but these strategies work only for Convolutional Neural Networks trained on image datasets; Severi et al. [72] showed that these approaches fail when tested on other data modalities and models. Furthermore, recent work by Goldwasser et.al [38] formulated a way to plant backdoors that are undetectable by any defense. In contrast, SafeNet is model agnostic and works for a variety of data modalities. Even if an attack is undetectable, the adversary can still poison only a subset of models, making the ensemble robust against poisoning. Recent work [60], [41] proposed differential privacy as a mitigation strategy against targeted attacks. However, differential privacy increases error rates, has limited effectiveness if the attack size is large, and in some cases may not even be an effective defense [45].

On the other hand, SafeNet does not suffer from these limitations and in certain instances can tolerate around 30% of the training data being poisoned, while being attack agnostic.

SafeNet is also robust to stronger model poisoning attacks [5], [8], [34], which are possible when data owners can train their models locally. SafeNet tolerates model poisoning because each model only contributes a single vote to the final ensemble prediction. In fact, all our empirical and theoretical analysis of SafeNet is computed for arbitrarily corrupted models.

F. Comparison with Federated Learning

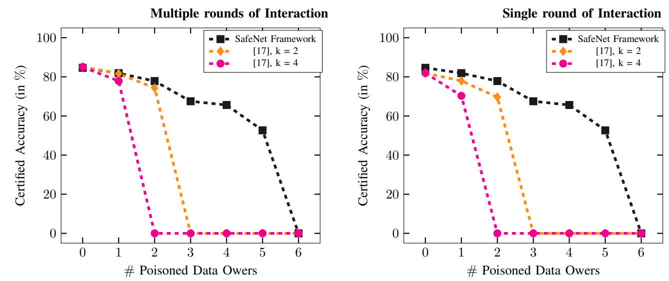
Federated Learning (FL) is a distributed machine learning framework that allows clients to train a global model without sharing their local training datasets to the central server. However, it differs from the PPML setting we consider in the following ways: (1) Clients do not share their local data to the server in FL, whereas PPML allows sharing of datasets; (2) Clients participate in multiple rounds of training in FL, whereas they communicate only once with the servers in PPML; (3) Clients receive the global model at each round in FL, while in SafeNet they secret-share their models once at the start of the protocol; and, finally, (4) PPML provides stronger confidentiality guarantees such as privacy of the global model.

It is possible to combine FL and MPC to guarantee both client and global model privacy [48], [83], [35], but this involves large communication overhead and is susceptible to poisoning [59]. For example, recent work [79], [8], [6] showed that malicious data owners can significantly reduce the learned global model's accuracy. Existing defenses against such owners use Byzantine-robust aggregation rules such as trimmed mean [82], coordinate-wise mean [81] and Krum [11], which have been show to be susceptible to backdoor and model poisoning attacks [34]. Recent work in FL such as FLTrust [16] and DeepSight [71] provide mitigation against backdoor attacks. Both strategies are inherently heuristic, while SafeNet offers provable robustness guarantees. FLTrust also requires access to a clean dataset, which is not required in our framework, and DeepSight inspects each model update before aggregation, which is both difficult in MPC and leads to privacy leakage from the updates, a drawback not found in SafeNet. An important privacy challenge is that federated learning approaches permit data reconstruction attacks when the central server is malicious [12]. SafeNet prevents such an attack, as it directly violates the security guarantee of the MPC, when instantiated for the malicious setting.

The most related strategy to ours is that by Cao et al. [17], who gave provable robustness guarantees for federated learning aggregation. They propose an ensembling strategy, where, with m data owners, n of which are malicious, they train $\binom{m}{k}$ global models and perform a majority vote. Here k is a hyperparameter denoting how many clients contribute to each global model. We now show that SafeNet outperforms this technique; we instantiate their strategy for our Fashion dataset setup and compare their Certified Accuracy metric to SafeNet's, with $m = 10$, $k = \{2, 4\}$ and FedAvg as the base algorithm.

Figure 10 (left) shows that SafeNet consistently outperforms [17], in terms of maintaining a high certified accuracy in the presence of large poisoning rates. Moreover, their strategy is

Fig. 10: Certified Accuracy of our framework compared to Cao et al.'s [17]. Left plot provides the certified accuracy when [17] is allowed to run for multiple rounds of interaction between the clients and the server. The right plot provides certified accuracy of both systems for a single round of interaction.



also particularly expensive when instantiated in MPC for both training and prediction. At training, their approach requires data owners to interact inside MPC to train $\binom{m}{k}$ models over multiple rounds. By contrast, SafeNet only requires interaction with the MPC once at the beginning of the training phase, making it significantly faster. When we allow their strategy only one round of communication as in SafeNet (see the right plot of Figure 10), certified accuracy further degrades. Their prediction phase also requires heavy computation. For example, at $k = 4$, they run prediction on 210 models, making it 21x more expensive than SafeNet. Note that our certified accuracy argument coincides with theirs at $k = 1$. They do not consider this setting, as it would make them vulnerable to data reconstruction attacks [12], an issue SafeNet does not face.

VI. Conclusion

In this paper, we extend the security definitions of MPC to account for data poisoning attacks when training machine learning models privately. We consider a novel adversarial model who can manipulate the training data of a subset of owners, and control a subset of servers in the MPC. To mitigate poisoning attacks, we propose SafeNet, a modular framework that trains an ensemble of multiple ML models locally at data owners and performs ensemble filtering and prediction inside the MPC, taking advantage of the MPC threat model to improve running time and poisoning robustness. We evaluate thoroughly the accuracy and efficiency of SafeNet in terms of training time and communication bandwidth, and also show that it is resilient against backdoor and targeted poisoning attacks. Our work provides one of the first connections between adversarial ML and MPC, and it opens up new directions of research to improve ML robustness in MPC settings.

VII. Acknowledgments

We thank Nicolas Papernot and Peter Rindal for helpful discussions and feedback. This research was sponsored by the U.S. Army Combat Capabilities Development Command Army Research Laboratory under Cooperative Agreement Number W911NF-13-2-0045 (ARL Cyber Security CRA). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Combat Capabilities Development Command Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

[1] M. Abspoel, D. Escudero, and N. Volgushev. Secure training of decision trees with continuous attributes. In *PoPETS*, 2021.

[2] A. Aly and N.P. Smart. Benchmarking privacy preserving scientific operations. In *ACNS*, 2019.

[3] T. Araki, A. Barak, J. Furukawa, T. Lichter, Y. Lindell, A. Nof, K. Ohara, A. Watzman, and O. Weinstein. Optimized honest-majority MPC for malicious adversaries - breaking the 1 billion-gate per second barrier. In *IEEE S&P*, 2017.

[4] T. Araki, J. Furukawa, Y. Lindell, A. Nof, and K. Ohara. High-throughput semi-honest secure three-party computation with an honest majority. In *ACM CCS*, 2016.

[5] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and Vitaly Shmatikov. How to backdoor federated learning. 2018.

[6] Bagdasaryan<B., A. Veit, Y. Hua, D. Estrin, and V. Shmatikov. How to backdoor federated learning. In *AISTATS*, 2020.

[7] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation (Extended Abstract). In *ACM STOC*, 1988.

[8] A. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo. Analyzing federated learning through an adversarial lens. In *ICML*, 2019.

[9] B. Biggio, I. Corona, G. Fumera, G. Giacinto, and F. Roli. Bagging classifiers for fighting poisoning attacks in adversarial classification tasks. In *International workshop on multiple classifier systems*, 2011.

[10] B. Biggio, B. Nelson, and P. Laskov. Poisoning attacks against support vector machines. In *ICML*, 2012.

[11] P. Blanchard, E. Mhamdi, R. Guerraoui, and J. Stainer. Byzantine-tolerant machine learning. In *NeurIPS*, 2017.

[12] F. Boenisch, A. Dziedzic, R. Schuster, A. Shamsabadi, I. Shumailov, and N. Papernot. When the curious abandon honesty: Federated learning is not private. In *arXiv*, 2021.

[13] M. Byali, H. Chaudhari, A. Patra, and A. Suresh. Flash: Fast and robust framework for privacy-preserving machine learning. PoPETS, 2020.

[14] M. Byali, A. Joseph, A. Patra, and D. Ravi. Fast secure computation for small population over the internet. *ACM CCS*, 2018.

[15] R. Canetti. Security and composition of multiparty cryptographic protocols. In *J. Cryptology*, 2000.

[16] X. Cao, M. Fang, J. Liu, and N. Gong. Fltrust: Byzantine-robust federated learning via trust bootstrapping. In *NDSS*, 2021.

[17] X. Cao, J. Jia, and N. Gong. Provably secure federated learning against malicious clients. In *AAAI*, 2021.

[18] N. Chandran, D. Gupta, and A. Obbattu, L.B. andShah. Simc: Ml inference secure against malicious clients at semi-honest cost. In *USENIX*, 2022.

[19] H. Chaudhari, A. Choudhury, A. Patra, and A. Suresh. ASTRA: High-throughput 3PC over Rings with Application to Secure Prediction. In *ACM CCSW*, 2019.

[20] H. Chaudhari, R. Rachuri, and A. Suresh. Trident: Efficient 4pc framework for privacy preserving machine learning. NDSS, 2020.

[21] B. Chen, W. Carvalho, N. Baracaldo, H. Ludwig, B. Edwards, T. Lee, I. M. Molloy, and B. Srivastava. Detecting backdoor attacks on deep neural networks by activation clustering. In *SafeAI@AAAI*, 2019.

[22] X. Chen, C. Liu, B. Li, K. Lu, and D. Song. Targeted backdoor attacks on deep learning systems using data poisoning. 2017.

[23] C.A. Choquette-Choo, N. Dullerud, A. Dziedzic, Y. Zhang, S. Jha, N. Papernot, and X. Wang. Ca{pc} learning: Confidential and private collaborative learning. In *ICLR*, 2021.

[24] J. Cohen, E. Rosenfeld, and Z. Kolter. Certified adversarial robustness via randomized smoothing. In *ICML*, 2019.

[25] R. Cramer, I. Damgård, D. Escudero, P. Scholl, and C. Xing. SPDZ2k: Efficient MPC mod 2ˆk for Dishonest Majority. *CRYPTO*, 2018.

[26] A. Dalskov, D. Escudero, and M. Keller. Fantastic four: Honest-majority four-party secure computation with malicious security. In *USENIX*, 2021.

[27] A.P.K. Dalskov, D. Escudero, and M. Keller. Secure evaluation of quantized neural networks. In *PoPETS*, 2020.

[28] I. Damgård, M. Keller, E. Larraia, V. Pastro, P. Scholl, and N. P. Smart. Practical covertly secure MPC for dishonest majority - or: Breaking the SPDZ limits. In *ESORICS*, 2013.

[29] I. Damgård, V. Pastro, N. P. Smart, and S. Zakarias. Multiparty Computation from Somewhat Homomorphic Encryption. In *CRYPTO*, 2012.

[30] D. Demmler, T. Schneider, and M. Zohner. ABY - A Framework for Efficient Mixed-Protocol Secure Two-Party Computation. In *NDSS*, 2015.

[31] J. Devlin, M.W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.

[32] D. Dua and C. Graff. UCI machine learning repository, 2017.

[33] D. Escudero, M. Jagielski, R. Rachuri, and P. Scholl. Adversarial Attacks and Countermeasures on Private Training in MPC. In *PPML@NeurIPS*, 2021.

[34] M. Fang, X. Cao, J. Jia, and N. Gong. Local model poisoning attacks to byzantine-robust federated learning. In *Usenix*, 2020.

[35] A. Fu, X. Zhang, N. Xiong, Y. Gao, H. Wang, and J. Zhang. Vfl: A verifiable federated learning with privacy-preserving for big data in industrial iot. In *IEEE Transactions on Industrial Informatics*, 2020.

[36] J. Geiping, L.H. Fowl, W.R. Huang, W. Czaja, G. Taylor, M. Moeller, and T. Goldstein. Witches' brew: Industrial scale data poisoning via gradient matching. In *ICLR*, 2021.

[37] O. Goldreich, S. Micali, and A. Wigderson. How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority. In *STOC*, 1987.

[38] S. Goldwasser, M. Kim, V. Vaikuntanathan, and O. Zamir. Planting undetectable backdoors in machine learning models. In *arXiv*, 2022.

[39] S. D. Gordon, S. Ranellucci, and X. Wang. Secure computation with low communication from cross-checking. In *ASIACRYPT*, 2018.

[40] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg. Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access*, 2019.

[41] S. Hong, V. Chandrasekaran, Y. Kaya, T. Dumitras, and N. Papernot. On the effectiveness of mitigating data poisoning attacks with gradient shaping. In *arxiv*, 2021.

[42] T.M.Harry Hsu, H.Qi, and M.Brown. Measuring the effects of non-identical data distribution for federated visual classification. In *IACR ePrint*.

[43] Y. Ishai, J. Kilian, K. Nissim, and E. Petrank. Extending Oblivious Transfers Efficiently. In *CRYPTO*, 2003.

[44] Y. Ishai, R. Kumaresan, E. Kushilevitz, and A. Paskin-Cherniavsky. Secure computation with minimal interaction, revisited. In *CRYPTO*, 2015.

[45] M. Jagielski and A. Oprea. Does differential privacy defeat data poisoning? In *DPML Workshop*, 2021.

[46] M. Jagielski, A. Oprea, B. Biggio, C. Liu, C.N. Rotaru, and B. Li. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In *IEEE S&P*, 2018.

[47] J. Jia, X. Cao, and N. Gong. Intrinsic certified robustness of bagging against data poisoning attacks. In *AAAI*, 2021.

[48] R. Kanagavelu, Z. Li, J. Samsudin, Y. Yang, F. Yang, R. Goh, M. Cheah, P. Wiwatphonthana, K. Akkarajitsakul, and S. Wang. Two-phase multi-party computation enabled privacy-preserving federated learning. In *ACM CCGRID*, 2020.

[49] M. Keller. MP-SPDZ: A versatile framework for multi-party computation. In *ACM CCS*, 2020.

[50] P.W. Koh and P. Liang. Understanding black-box predictions via influence functions. In *ICML*, 2017.

[51] P.W. Koh, J. Steinhardt, and P. Liang. Stronger data poisoning attacks break data sanitization defenses. In *arXiv*, 2018.

[52] S. Kornblith, J. Shlens, and Q.V. Le. Do better imagenet models transfer better? In *CVPR*, 2019.

[53] N. Kumar, M. Rathee, N. Chandran, D. Gupta, A. Rastogi, and R. Sharma. Cryptflow: Secure tensorflow inference. In *IEEE Security & Privacy*, 2020.

[54] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, 1998.

[55] R. Lehmkuhl, P. Mishra, A. Srinivasan, and R.A. Popa. Muse: Secure inference resilient to malicious clients. In *USENIX*, 2021.

[56] Y. Lindell. Fast cut-and-choose-based protocols for malicious and covert adversaries. In *J. Cryptology*, 2016.

[57] Y. Lindell and B. Pinkas. An efficient protocol for secure two-party computation in the presence of malicious adversaries. In *EUROCRYPT*, 2007.
[58] K. Liu, B. Dolan, and S. Garg. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *RAID*, 2018.
[59] Z. Liu, Jiale G., W. Yang, K.n Fan, J.and Lam, and J. Zhao. Privacy-preserving aggregation in federated learning: A survey. In *arXiv*, 2022.
[60] Y. Ma, X. Zhu, and J. Hsu. Data poisoning against differentially-private learners: Attacks and defenses. In *IJCAI*, 2019.
[61] P. Mohassel and M. K. Franklin. Efficiency tradeoffs for malicious two-party computation. In *PKC*, 2006.
[62] P. Mohassel and P. Rindal. ABY^3: A Mixed Protocol Framework for Machine Learning. In *ACM CCS*, 2018.
[63] P. Mohassel, M. Rosulek, and Y. Zhang. Fast and Secure Three-party Computation: Garbled Circuit Approach. In *CCS*, 2015.
[64] P. Mohassel and Y. Zhang. Secureml: A system for scalable privacy-preserving machine learning. In *IEEE S&P*, 2017.
[65] L. Muñoz-González, B. Biggio, A. Demontis, A. Paudice, V. Wongrassamee, E.C. Lupu, and F. Roli. Towards poisoning of deep learning algorithms with back-gradient optimization. In *AISec@CCS*, 2017.
[66] J. Newsome, B. Karp, and D. Song. Paragraph: Thwarting signature learning by training maliciously. In *RAID*, 2006.
[67] J. B. Nielsen and C. Orlandi. Cross and clean: Amortized garbled circuits with constant overhead. In *TCC*, 2016.
[68] A. Patra, T. Schneider, A. Suresh, and H. Yalame. Aby2.0: Improved mixed-protocol secure two-party computation. In *USENIX*, 2021.
[69] A. Patra and A. Suresh. Blaze: Blazing fast privacy-preserving machine learning. NDSS, 2020.
[70] D. Rathee, M. Rathee, N. Kumar, N. Chandran, D. Gupta, A. Rastogi, and R. Sharma. Cryptflow2: Practical 2-party secure inference. In *ACM CCS*, 2020.
[71] P. Rieger, T. Nguyen, M. Miettinen, and A. Sadeghi. Deepsight: Mitigating backdoor attacks in federated learning through deep model inspection. In *NDSS*, 2022.
[72] G. Severi, J. Meyer, S. Coull, and A. Oprea. Explanation-guided backdoor poisoning attacks against malware classifiers. In *USENIX*, 2021.
[73] O. Suciu, R. Marginean, Y. Kaya, H. Daume III, and T. Dumitras. When does machine learning FAIL? generalized transferability for evasion and poisoning attacks. In *USENIX*, 2018.
[74] B. Tran, J. Li, and A. Madry. Spectral signatures in backdoor attacks. In *NeurIPS*, 2018.
[75] S. Wagh, D. Gupta, and N. Chandran. SecureNN: Efficient and private neural network training. In *PoPETS*, 2019.
[76] S. Wagh, S. Tople, F. Benhamouda, E. Kushilevitz, P. Mittal, and T. Rabin. Falcon: Honest-majority maliciously secure framework for private deep learning. In *PoPETS*, 2021.
[77] B. Wang, Y. Yao, S. Shan, H. Li, H. Viswanath, B. Zheng, and B.Y. Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *IEEE S&P*, 2019.
[78] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.
[79] C. Xie, S. Koyejo, and I. Gupta. Fall of empires: Breaking byzantine-tolerant SGD by inner product manipulation. In *UAI*, 2019.
[80] A. C. Yao. Protocols for Secure Computations. In *FOCS*, 1982.
[81] D. Yin, Y. Chen, K. Ramchandran, and P. Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. In *ICML*, 2018.
[82] D. Yin, Y. Chen, K. Ramchandran, and P. Bartlett. Defending against saddle point attack in byzantine-robust distributed learning. In *ICML*, 2019.
[83] H. Zhu, R. Mong Goh, and W. Ng. Privacy-preserving weighted federated learning within the secret sharing framework. In *IEEE Access*, 2020.

Appendix A
Robust Accuracy Analysis

In this section we first provide a detailed proof on the size of the validation dataset D_{val} such that all clean models clear the filtering stage of the training phase of our framework. We then provide a proof on achieving lower bounds on the test accuracy of our framework given all clean models are a part of the ensemble.

The main idea of deriving the minimum size of D_{val} uses the point that the errors made by a clean model on a clean subset of samples in D_{val} can be viewed as a Binomial distribution in $(m-t)n$ and p, where n denotes the size of the validation dataset D_k^{v} contributed by an owner C_k. We can then upper bound the total errors made by a clean model by applying Chernoff bound and consequently compute the size of D_{val}.

Lemma 3. *Let $\mathcal{A}^p_{soc}$ be an adversary who poisons t out of m data owners and corrupts T out of N servers, and thus contributes t poisoned models to ensemble E, given as output by Algorithm 1. Assume that Π_{train} securely realizes functionality $\mathcal{F}_{\mathsf{pTrain}}$ and every clean model in E makes an error on a clean sample with probability at most $p < 1-\phi$, where ϕ is the filtering threshold.*

If the validation dataset has at least $\frac{(2+\delta)m\log 1/\epsilon}{\delta^2(m-t)p}$ samples and $0 \le t < \frac{m(1-\phi-p)}{(1-p)}$, then all clean models pass the filtering stage of the training phase with probability at least $1-\epsilon$, where $\delta = \frac{(1-\phi)m-t}{(m-t)p} - 1$ and ϵ denotes the failure probability.

Proof. Assume that each owner contributes equal size validation dataset D_k^{v} of n samples, then the combined validation set D_{val} collected from m data owners is comprised of mn i.i.d. samples. However, given an adversary $\mathcal{A}^{\text{p}}_{\text{soc}}$ from our threat model, there can be at most t poisoned owners contributing tn poisoned samples to D_{val}. We define a Bernoulli random variable as follows:

$$X_i = \begin{cases} 1, & \text{w.p. } p \\ 0, & \text{w.p. } 1-p \end{cases}$$

where X_i denotes if a clean model makes an error on the i^{th} clean sample in the validation dataset. Then there are $\text{Bin}((m-t)n, p)$ errors made by the clean model on the clean subset of samples in D_{val}. Note that, a model passes the filtering stage only when it makes $\ge \phi mn$ correct predictions. We assume that the worst case where the clean model makes incorrect predictions on all the tn poisoned samples present in D_{val}. As a result, the clean model must make at most $(1-\phi)mn - tn$ errors on the clean subset of D_{val} with probability $1-\epsilon$. We can upper bound the probability the model makes at least $(1-\phi)mn + 1 - tn$ errors with a multiplicative Chernoff bound with $\delta > 0$:

$$\mathsf{Pr}[\textstyle\sum_{i=1}^{(m-t)n} X_i > (1-\phi)mn - tn] = \mathsf{Pr}\left[\sum_{i=1}^{n} X_i > (1+\delta)\mu\right] < e^{-\frac{\delta^2\mu}{2+\delta}}$$

where $\mu = (m-t)np$ (the mean of $Bin(mn-tn, p)$) and $\delta = \frac{(1-\phi)m-t}{(m-t)p}$. The chernoff bound gives that the probability the clean model makes too many errors is at most $e^{-\frac{\delta^2\mu}{2+\delta}} = \epsilon$. Then it suffices to have this many samples:

$$|D_{\text{val}}| = mn = \frac{(2+\delta)m\log 1/\epsilon}{\delta^2(m-t)p}$$

where ϵ denotes the failure probability and $t < \frac{m(1-\phi-p)}{(1-p)}$. The inequality on t comes from requiring $\delta > 0$.

□

We use a similar strategy as above to compute the lower bound on the test accuracy. On a high level, the proof follows by viewing the combined errors made by the clean models as a Binomial distribution $Bin(m-t, p)$. We can then upper bound the total errors made by all the models in the ensemble by applying Chernoff bounds and consequentially lower bound the ensemble accuracy.

Theorem 4. *Assume that the conditions in Lemma 3 hold against adversary $\mathcal{A}^{p}_{soc}$ poisoning at most $t < \frac{m}{2}\frac{1-2p}{1-p}$ owners and that the errors made by the clean models are independent. Then E correctly classifies new samples with probability at least $p_c = (1-\epsilon)\left(1 - e^{-\frac{\delta'^2\mu'}{2+\delta'}}\right)$, where $\mu' = (m-t)p$ and $\delta' = \frac{m-2t}{2\mu'} - 1$.*

Proof. Lemma 3 shows that, with probability $> 1-\epsilon$, no clean models will be filtered during ensemble filtering. Given all clean models pass the filtering stage, we consider the worst case where even the t poisoned models bypass filtering. Now, given a new test sample, $m-t$ clean models have uncorrelated errors each with probability at most p, the error made by each clean model can be viewed as a Bernoulli random variable with probability p and so the total errors made by clean models follow a binomial $X \sim \mathrm{Bin}(m-t, p)$. We assume that a new sample will be misclassified by all t of the poisoned models. Then the ensemble as a whole makes an error if $t + Bin(m-t, p) > m/2$. We can then bound the probability this occurs by applying Chernoff bound as follows:

$$\mathsf{Pr}\left[X + t \geq \frac{m}{2}\right] = \mathsf{Pr}\left[X \geq (1+\delta')\mu'\right] \leq e^{-\frac{\delta'^2\mu'}{2+\delta'}},$$

where $\mu' = (m-t)p$ is the mean of X and $\delta' = \frac{m-2t}{2\mu'} - 1 > 0$. Then the probability of making a correct prediction can be lower bounded by:

$$\mathsf{Pr}\left[X < \frac{m}{2} - t\right] > 1 - e^{-\frac{\delta'^2\mu'}{2+\delta'}},$$

given the number of poisoned models

$$t < \frac{m(1-2p)}{2(1-p)}.$$

The inequality on t comes from the constraint $\delta' > 0$ for the Chernoff bound to hold. Note that, the above bound holds only when all the clean models pass the filtering stage, which occurs with probability at least $1-\epsilon$ by Lemma 3. Then the bound on the probability of making a correct prediction by the ensemble can be written as:

$$\mathsf{Pr}\left[X < \frac{m}{2} - t\right] > (1-\epsilon)\left(1 - e^{-\frac{\delta'^2\mu'}{2+\delta'}}\right)$$

□

APPENDIX B
SECURITY PROOFS

We prove that protocol Π_{train} is secure against an adversary of type $\mathcal{A}^{\mathsf{p}}_{\mathsf{soc}}$. Towards this, we first argue that protocol Π_{train} securely realizes the standard ideal-world functionality $\mathcal{F}_{\mathsf{pTrain}}$. We use simulation based security to prove our claim. Next, we argue that the ensemble E trained using Π_{train} protocol provides poisoning robustness against $\mathcal{A}^{\mathsf{p}}_{\mathsf{soc}}$.

Theorem 2. *Protocol Π_{train} is secure against adversary $\mathcal{A}^{p}_{soc}$ who poisons t out of m data owners and corrupts T out of N servers.*

Proof. Let $\mathcal{A}^{\mathsf{p}}_{\mathsf{soc}}$ be a real-world adversary that semi-honestly corrupts T out of N servers at the beginning of the protocol Π_{train}. We now present the steps of the ideal-world adversary (simulator) $\mathcal{S}_f$ for $\mathcal{A}^{\mathsf{p}}_{\mathsf{soc}}$. Note that, in the semi-honest setting $\mathcal{S}_f$ already posses the input of $\mathcal{A}^{\mathsf{p}}_{\mathsf{soc}}$ and the final output shares of $\mathbf{b}^{\mathsf{val}}$. $\mathcal{S}_f$ acts on behalf of $N-T$ honest servers, sets their shares as random values in $\mathbb{Z}_{2^\ell}$ and simulates each step of Π_{train} protocol to the corrupt servers as follows:

- No simulation is required to construct $[\![\cdot]\!]$-shares of ensemble E and validation dataset D_{val} as it happens locally.
- $\mathcal{S}_f$ simulates messages on behalf of honest servers as a part of the protocol steps of Π_{zvec} with public value m as the input and eventually sends and receives appropriate $[\![\cdot]\!]$-shares of $\mathbf{b}^{\mathsf{val}}$ to and from $\mathcal{A}^{\mathsf{p}}_{\mathsf{soc}}$.
- For $k \in [1, m]$:
  - $\mathcal{S}_f$ simulates messages on behalf of honest servers, as a part of the protocol steps of $\Pi_{\mathcal{M}\mathsf{pred}}$, with inputs to the protocol as $[\![\cdot]\!]$-shares of $\mathcal{M}_k$ and D_{val} and eventually sends and receives appropriate $[\![\cdot]\!]$-shares of PREDS_k to and from $\mathcal{A}^{\mathsf{p}}_{\mathsf{soc}}$.
  - $\mathcal{S}_f$ simulates messages on behalf of honest servers, as a part of the protocol steps of Π_{acc}, with inputs to the protocol as $[\![\cdot]\!]$-shares of PREDS_k and $\mathbf{y}_{D_{\mathsf{val}}}$ and eventually sends and receives appropriate $[\![\cdot]\!]$-shares of AccVal_k to and from $\mathcal{A}^{\mathsf{p}}_{\mathsf{soc}}$.
  - No simulation is required for subtraction with threshold ϕ as it happens locally.
  - $\mathcal{S}_f$ simulates messages on behalf of honest servers, as a part of the protocol steps of Π_{comp}, with inputs to the protocols as $[\![\cdot]\!]$-shares of $\mathrm{AccVal} - \phi$ and at the end $\mathcal{S}_f$ instead sends the original shares of $\mathbf{b}^{\mathsf{val}}_k$ as shares of b' associated to $\mathcal{A}^{\mathsf{p}}_{\mathsf{soc}}$.
  - No simulation is required to assign $[\![\mathbf{b}^{\mathsf{val}}_k]\!] = [\![b']\!]$.

The proof now simply follows from the fact that simulated view and real-world view of the adversary are computationally indistinguishable and concludes that Π_{train} securely realizes functionality $\mathcal{F}_{\mathsf{pTrain}}$.

Now given the output of Π_{train} protocol is an ensemble E, we showed in the proof of Theorem III-D that E correctly classifies a sample with probability at least p_c. As a result the underlying trained model also provides poisoning robustness against $\mathcal{A}^{\mathsf{p}}_{\mathsf{soc}}$.

□

We use a similar argument to show protocol Π_{pred} is secure against adversary $\mathcal{A}^{\mathsf{p}}_{\mathsf{soc}}$.

Theorem 6. *Protocol Π_{pred} is secure against adversary $\mathcal{A}^{p}_{soc}$ who poisons t out of m data owners and corrupts T out of N servers.*

Proof. Let $\mathcal{A}^{\mathsf{p}}_{\mathsf{soc}}$ be a real-world adversary that poisons t out of m owners and semi honestly corrupts T out of N servers at the beginning of Π_{pred} protocol. We present steps of the ideal-world adversary (simulator) $\mathcal{S}_f$ for $\mathcal{A}^{\mathsf{p}}_{\mathsf{soc}}$. $\mathcal{S}_f$ on behalf of the honest servers, sets their shares as random values in $\mathbb{Z}_{2^\ell}$ and simulates each step of Π_{pred} protocol to the corrupt servers as follows:

– $\mathcal{S}_f$ simulates messages on behalf of honest servers as a part of the protocol steps of Π_{zvec} with public value L as the input and eventually sends and receives appropriate $[\![\cdot]\!]$-shares of $\mathbf{z}$ to and from $\mathcal{A}^{\mathsf{p}}_{\mathsf{soc}}$.
– For $k \in [1, m']$:
  – $\mathcal{S}_f$ simulates messages on behalf of honest servers, as a part of the protocol steps of $\Pi_{\mathcal{M}\mathsf{pred}}$, which takes input as $[\![\cdot]\!]$-shares of $\mathcal{M}_k$ and $\mathbf{x}$. $\mathcal{S}_f$ eventually sends and receives appropriate $[\![\cdot]\!]$-shares of $\mathbf{Preds}$ to and from $\mathcal{A}^{\mathsf{p}}_{\mathsf{soc}}$.
  – For every multiplication of $[\![\mathbf{b}^{\mathsf{val}}_k]\!]$ with respect to each element in $\mathbf{Preds}$, $\mathcal{S}_f$ simulates messages on behalf of honest servers, as a part of the protocol steps of Π_{mult}, which takes input as $[\![\cdot]\!]$-shares of $\mathbf{Preds}_j$ and $\mathbf{b}^{\mathsf{val}}_k$. $\mathcal{S}_f$ eventually sends and receives appropriate $[\![\cdot]\!]$-shares of $\mathbf{b}^{\mathsf{val}}_k \times \mathbf{Preds}_j$ to and from $\mathcal{A}^{\mathsf{p}}_{\mathsf{soc}}$.
  – No simulation is required to update $[\![\mathbf{z}]\!]$ as addition happens locally.
– $\mathcal{S}_f$ simulates messages on behalf of honest servers, as a part of the protocol steps of Π_{amx}, which takes input as $[\![\cdot]\!]$-shares of $\mathbf{z}$. At the end $\mathcal{S}_f$ instead forwards the original $[\![\cdot]\!]$-shares of OP associated to $\mathcal{A}^{\mathsf{p}}_{\mathsf{soc}}$.

The proof now simply follows from the fact that simulated view and real-world view of the adversary are computationally indistinguishable. Poisoning robustness argument follows from the fact that the ensemble E used for prediction was trained using protocol Π_{train} which was shown to be secure against $\mathcal{A}^{\mathsf{p}}_{\mathsf{soc}}$ in Theorem 2. □

This concludes the security proofs of our training and prediction protocols.

APPENDIX C
ADDITIONAL ALGORITHMS

A. SafeNet Prediction phase in Transfer Learning Setting

In this section we provide a modified version of SafeNet's Prediction algorithm in the transfer learning setting, to improve the running time and communication complexity of SafeNet. Algorithm 3 provides the details of SafeNet's prediction phase below.

Algorithm 3 SafeNet Prediction Algorithm in Transfer Learning Setting

Input: Secret-shares of backbone model $\mathcal{M}_B$, ensemble of m fine-tuned models $E = \{\mathcal{M}_1, \ldots, \mathcal{M}_m\}$, vector $\mathbf{b}^{\mathsf{val}}$ and client query $\mathbf{x}$.
// MPC computation in secret-shared format
– Construct vector $\mathbf{z}$ of all zeros of size L, where L denotes the number of distinct class labels.
– Run forward pass on $\mathcal{M}_B$ with input $\mathbf{x}$ till its penultimate layer. Let $\mathbf{p}$ denote the output vector of the penultimate layer.
– For $k \in [1, m]$:
- Run forward pass on final layer of $\mathcal{M}_k$ with input as $\mathbf{p}$. Let the output of the computation be **Preds**, which is one-hot encoding of the predicted label.
- Multiply $\mathbf{b}^{\mathsf{val}}_k$ to each element of **Preds**.
- Add $\mathbf{z} = \mathbf{z} + \mathbf{Preds}$.
– Run argmax with input as $\mathbf{z}$ and obtain OP as the output.
return OP

B. SafeNet Training with Computationally Restricted Owners

In this section we provide a modified version of SafeNet's Training Algorithm, to accommodate when a subset of data owners are computationally restricted, i.e., they can not train their models locally. Algorithm 4 provides the details of SafeNet's training steps below.

APPENDIX D
ADDITIONAL EXPERIMENTS

Logistic Regresssion, Multiclass Classification. We use the same strategies for the Backdoor and Targeted attacks on the MNIST dataset. For BadNets, we select the initial class $y_s = 4$ and the target label $y_t = 9$, and use the same $y_t = 9$ for the targeted attack. Table III provides a detailed analysis of the training time, communication, test accuracy, and success rate for both frameworks, in presence of a single poisoned owner. The worst-case adversarial success for SafeNet is in Figure 11. The slow rise in the success rate of the adversary across multiple attacks shows the robust accuracy property of our framework translates smoothly for the case of a multi-class classification problem.

Experiments on Adult Dataset. We report results for backdoor and targeted attacks on a single layer deep neural network (with 10 hidden nodes) trained on the Adult dataset. We use a similar attack strategy as used for logistic regression model in Section IV-D. We observe that no instance is present with true label $y = 1$ for feature capital-loss $= 1$. Consequently, we choose a set of $k = 100$ target samples $\{x^t_i\}^k_{i=1}$ with true label $y_s = 0$, and create backdoored samples $\{Pert(x^t_i), y_t = 1\}^k_{i=1}$, where $Pert(\cdot)$ function sets the capital-loss feature in x_t to 1. For the targeted attack, we only use TGT-Top because

TABLE III: Training time (in seconds) and Communication (in GB) over a LAN network for traditional PPML and SafeNet framework training a multiclass logistic regression on MNIST. n denotes the number of epochs in the PPML framework. The time and communication reported for SafeNet is for end-to-end execution. Test Accuracy and Success Rate are given for a single poisoned owner.

MPC	Setting	Framework	Training Time (s)	Communication (GB)	Backdoor Attack		Targeted Attack		
					Test Accuracy	Success Rate	Test Accuracy	Success Rate-Top	Success Rate-Foot
3PC [4]	Semi-Honest	PPML	n×243.55	n×55.68	89.14%	100%	87.34%	83%	90%
		SafeNet	10.03	2.05	88.68%	4%	88.65%	1%	10%
4PC [26]	Malicious	PPML	n×588.42	n×105.85	89.14%	100%	87.22%	83%	90%
		SafeNet	23.39	3.78	88.65%	4%	88.65%	1%	10%

Algorithm 4 SafeNet Training Algorithm with Computationally Restricted Owners

Input: m total data owners of which m_r subset of owners are computationally restricted, each owner C_k's dataset D_k.

// Computationally Restricted Owner's local computation in plaintext

– For $k \in [1, m_r]$:

- Separate out D_k^{v} from D_k.
- Secret-share cross-validation dataset D_k^{v} and training dataset $D_k \setminus D_k^{\mathsf{v}}$ to servers.

// Computationally Unrestricted Owner's local computation in plaintext

– For $k \in [m_{r+1}, m]$:

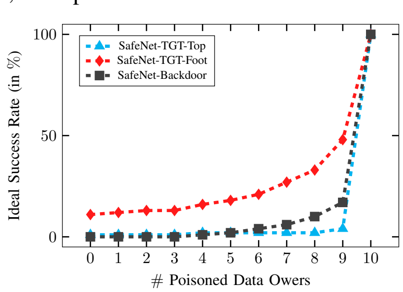
- Separate out D_k^{v} from D_k. Train $\mathcal{M}_k$ on $D_k \setminus D_k^{\mathsf{v}}$.
- Secret-share D_k^{v} and $\mathcal{M}_k$ to servers.

// MPC computation in secret-shared format

1. For $k \in [1, m_r]$: // Train local models for restricted owners
   - Train $\mathcal{M}_k$ on $D_k \setminus D_k^{\mathsf{v}}$.
2. Construct a common validation dataset $D_{\text{val}} = \cup_{i=1}^{m} D_i^{\mathsf{v}}$ and collect ensemble of models $E = \{\mathcal{M}_i\}_{i=1}^{m}$
3. Initialize a vector $\mathbf{b}^{\text{val}}$ of zeros and of size m.
4. For $k \in [1, m]$: // Ensemble Filtering
   - $\text{AccVal}_k = Accuracy(\mathcal{M}_k, D_{\text{val}})$ // Compute validation accuracy of $\mathcal{M}_k$ over D_{val}.
   - If $\text{AccVal}_k > \phi$: // Compare against threshold
     – Set $\mathbf{b}_k^{\text{val}} = 1$ // Set k^{th} position in $\mathbf{b}^{\text{val}}$ to 1

return E and $\mathbf{b}^{\text{val}}$

Fig. 11: Worst-case adversarial success of multi-class logistic regression on MNIST in the SafeNet framework for backdoor and targeted attacks. The adversary can change the set of c poisoned owners per sample. SafeNet achieves certified robustness up to 9 poisoned owners out of 20 against backdoor and TGT-TOP attacks. The TGT-Foot attack targeting low-confidence samples has slightly higher success, as expected.



more than 50 out of 100 samples for TGT-Foot are misclassified before poisoning. Table IV provides the training time and communication complexity of both PPML and SafeNet frameworks. Figure 13 (a) and (b) provide the success rates in both frameworks and show the resilience of SafeNet against backdoor and targeted attacks.

TABLE IV: Training Time (in seconds) and Communication (in GB) for training a single layer neural network model on the Adult dataset. n denotes the number of epochs required for training the the neural network in the PPML framework. The values reported for SafeNet are for its total execution.

MPC	Setting	Framework	Training Time (s)	Communication (GB)
3PC	Semi-Honest [4]	PPML	n×8.72	n×0.87
		SafeNet	5.79	1.32
	Malicious [26]	PPML	n×223.15	n×16.49
		SafeNet	179.58	19.29
4PC	Malicious [26]	PPML	n×18.54	n×1.69
		SafeNet	14.67	2.53

Experiments on Fashion Dataset. We present results on one and two layer deep neural networks trained on the Fashion dataset. We perform the same set of backdoor and targeted attacks as described in Section IV. Table V provides detailed analysis of the training time, communication, test accuracy, and success rate for traditional PPML and SafeNet frameworks. We observe similar improvements, where for instance in the 4PC setting, SafeNet has $42\times$ and $43\times$ improvement in training time and communication complexity over the PPML framework, for $n = 10$ epochs for a two hidden layer neural network. Figure 12 shows the worst-case attack success in SafeNet (where the attacker can choose the subset of corrupted owners per sample) and the results are similar to Figure 8.

Fig. 12: Worst-case adversarial success of one and two layer Neural Networks on FASHION dataset in SafeNet framework for varying poisoned owners.

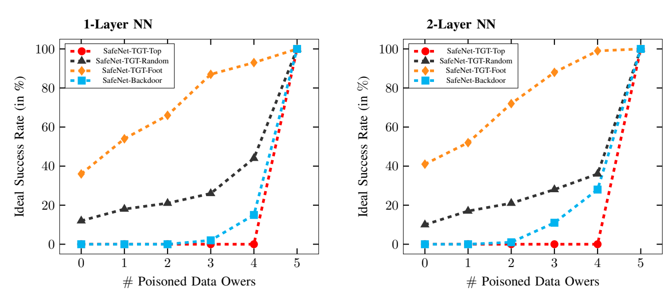


Fig. 13: Attack Success Rate and a Neural Network in PPML and SafeNet frameworks, trained over Adult dataset, for varying corrupt owners launching Backdoor (a) and Targeted (b) attacks. Plot (c) gives the worst-case adversarial success of SafeNet when a different set of poisoned owners is allowed per sample.

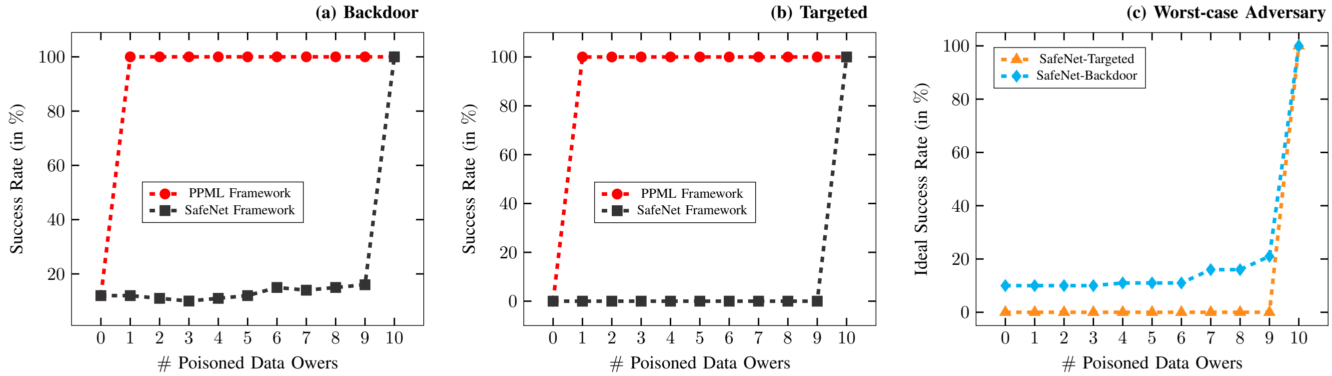


TABLE V: Training Time (in seconds) and Communication (in GB) of PPML and SafeNet frameworks for one and two layer neural network on the Fashion dataset. n denotes the number of epochs in the PPML framework. The time and communication reported for SafeNet framework is for end-to-end execution. Test Accuracy and Success Rate is given for the case when a single owner is corrupt.

MPC	Setting	No. Hidden Layers	Framework	Training Time (s)	Communication (GB)	Backdoor Attack		Targeted Attack		
						Test Accuracy	Success Rate	Test Accuracy	Success Rate-Top	Success Rate-Foot
3PC [4]	Semi-Honest	1	PPML	n×382.34	n×96.37	83.94%	100%	81.85%	100%	100%
			SafeNet	65.71	14.58	84.54%	0%	84.37%	0%	38%
		2	PPML	n×474.66	n×125.58	84.35%	100%	83.39%	100%	100%
			SafeNet	108.12	27.98	84.93%	0%	84.93%	0%	46%
4PC [26]	Malicious	1	PPML	n×869.12	n×174.12	83.71%	100%	81.94%	100%	100%
			SafeNet	152.68	26.89	84.48%	0%	84.42%	0%	38%
		2	PPML	n×1099.06	n×227.23	84.35%	100%	83.26%	100%	100%
			SafeNet	258.72	51.66	84.86%	0%	84.93%	0%	46%

Appendix E
Datasets

Table VI provides the details of the datasets and the corresponding models used for training both frameworks.

TABLE VI: Datasets and ML models used for comparison between SafeNet and PPML framework. Each value in the Architecture column is an array which represents the number of nodes in each layer starting from the input layer. Each model is trained for $n = 10$ epochs for both frameworks and mini-batch gradient descent with batch size of 128.

Dataset	Features	Train Samples per Owner	Test Samples	Labels	ML Algorithm	Architecture
Digit 1/7	784	650	2163	2	Logistic Regression	[784,1]
MNIST	784	3000	10000	10	Logistic Regression	[784,10]
Adult	108	1628	16281	2	Neural Network	[108,10,1]
Fashion	784	3000	10000	10	Neural Network	[784,128,10] [784,128,128,10] [784,128,128,128,10]