SafeNet: The Unreasonable Effectiveness of Ensembles in Private Collaborative Learning

***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, by design, MPC protocols faithfully compute the training functionality, which the adversarial ML community has shown to leak private information and can be tampered with in poisoning attacks. In this work, we argue that model ensembles, implemented in our framework called SafeNet, are a highly MPC-amenable way to avoid many adversarial ML attacks. The natural partitioning of data amongst owners in MPC training allows this approach to be highly scalable at training time, provide provable protection from poisoning attacks, and provably defends against a number of privacy attacks. We demonstrate SafeNet's efficiency, accuracy, and resilience to poisoning on several machine learning datasets and models trained in end-to-end and transfer learning scenarios. For instance, SafeNet reduces backdoor attack success significantly, while achieving $39\times$ faster training and $36\times$ less communication than the four-party MPC framework of Dalskov et al. [31]. Our experiments show that ensembling retains these benefits even in many non-iid settings. The simplicity, cheap setup, and robustness properties of ensembling make it a strong first choice for training ML models privately in MPC.**

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) frameworks [1], [31], [32], [71], [73], [75], [91], [92], [95] for training and inference of 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], [43], [47], [52]. 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. Furthermore, the guarantees of MPC provide privacy against an adversary observing the communication in the protocol, but does not protect against any sensitive information leaked by the model about its training set. Many privacy attacks are known to allow inference on machine learning models' training sets, and protecting against these attacks is an active area of research.

In this paper, we study the impact of these adversarial machine learning threats on standard 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 [25], [47] and targeted [43], [58] poisoning attacks on four MPC frameworks and five 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 apply ensembling technique from ML, implemented in our framework called SafeNet, which, in the collaborative learning setting we consider, is an effective defense against poisoning attacks, while also simultaneously preventing various types of privacy 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. The bounded contribution of each local model also gives a provable privacy guarantee for SafeNet. 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 $39\times$ faster than the [31] PPML protocol and requires $36\times$ less communication. Finally, we investigate settings with diverse data distributions among owners, and evaluate the accuracy and robustness of SafeNet under multiple data imbalance conditions.

To summarize, our contributions are as follows:

Adversarial ML-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 and privacy 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. The attacker might also seek to learn sensitive information about the local datasets through the trained model.

SafeNet Ensemble Design. We propose SafeNet, which adapts ensembling technique from ML to the collaborative MPC setting by having data owners train models locally and aggregation of predictions is performed securely inside the MPC. We show that this procedure gives provable privacy and security guarantees, which improves as models become more accurate. We also propose various novel extensions to this ensembling strategy which make SafeNet applicable to a wider range of training settings (including transfer learning and accommodating computationally restricted owners). SafeNet's design is agnostic to the underlying MPC framework and we show it can be instantiated 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 [4], [31], [35] and five 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 even under non-iid data distributions.

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) [7], [34], [44], [50], [98] 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 [4], [7], [14], [22], [74] and dishonest majority [33], [34], [44], [72], [98]. The settings of two-party computation (2PC) [64], [65], [78], [98], three parties (3PC) [3], [4], [74], and four parties (4PC) [23], [31], [46], [51] have been widely studied as they provide efficient protocols. Additionally, recent works in the area of privacy preserving ML propose training and prediction frameworks [1], [61], [73], [75], [81], [82], [91], [92] 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

In a data poisoning attack, an adversary controls a subset of the training dataset, and uses this to influence the model trained on that training set. In a backdoor attack [25], [47], [77], 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 [**?**], [58], [86], 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 [20], [62] show that MPC algorithms applied at test time can be compromised by malicious users, allowing for efficient *model extraction* attacks. Second, Escudero et al. [39] 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.

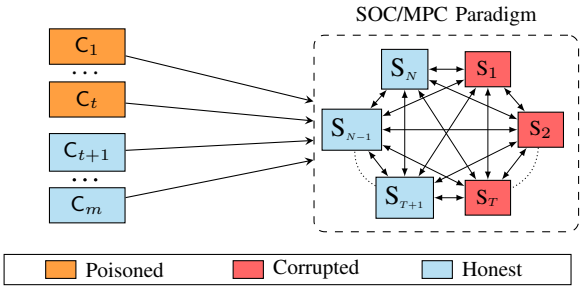

Fig. 1: Threat model considered in our setting. The adversary $\mathcal{A}^{\mathrm{p}}_{\mathrm{soc}}$ can poison at most t out of m data owners and corrupt at most T out of N servers participating in the MPC computation. C_i and $\mathcal{S}_j$ denote the ith data owner and j^{th} server.

Model ensembles have been proposed as a defense for ML poisoning and privacy attacks in prior work in both the centralized training setting [9], [53] and the collaborative learning setting. Compared to centralized approaches, which process a single dataset, 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 and privacy guarantees. Ensembles have also been proposed in MPC to protect data privacy [27] and in federated learning to provide poisoning robustness [17]. Our work provides a stronger privacy analysis, protecting from a broader range of threats than [27], and additionally offers robustness guarantees. We provide a more detailed comparison with these approaches in Section III-F.

III. SafeNet: Using Ensembles in MPC

We describe here our threat model and show how to implement ensembles in MPC. We then show that ensembling gives us provable robustness to poisoning and privacy adversaries.

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 [1], [14], [31], [32], [73], [75], [82], [91], [92] for training model $\mathcal{M}$ privately, where the owners secret-share their respective datasets to a set of outsourced servers, who execute the MPC protocols to train $\mathcal{M}$. The final output 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 servers, we define an adversary $\mathcal{A}_{\mathrm{soc}}$, similar to prior work [1], [31], [73], [75], [82], [92]. $\mathcal{A}_{\mathrm{soc}}$ 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. 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 a given 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.

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. 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], [52], [76]. 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 such attacks, we define a poisoning adversary $\mathcal{A}_{\mathrm{p}}$ that can poison a subset of local datasets of size at most $t < m$. Data owners with poisoned data are called *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 $\mathcal{A}_{\mathrm{p}}$, 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 $\mathcal{A}^{\mathrm{p}}_{\mathrm{soc}}$ for our

threat model (Figure 1) that corrupts servers in the MPC and poisons the owners' datasets:

- $\mathcal{A}^{\text{p}}_{\text{soc}}$ plays the role of $\mathcal{A}_{\text{p}}$ and poisons t out of m data owners that secret share their training data to the servers.
- $\mathcal{A}^{\text{p}}_{\text{soc}}$ 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 $\mathcal{A}^{\text{p}}_{\text{soc}}$ controls do not interfere in the execution of the MPC protocols after secret-sharing their data and also do not influence the honest owners.

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 $\mathcal{A}^{\text{p}}_{\text{soc}}$. A training protocol Π_{train} is secure against adversary $\mathcal{A}^{\text{p}}_{\text{soc}}$ 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 $\mathcal{A}^{\text{p}}_{\text{soc}}$ 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.

Extension to Privacy Adversary. While MPC guarantees no privacy leakage during the execution of the protocol, it makes no promises about privacy leakage that arises by observing the output of the protocol. This has motivated a combination of differential privacy guarantees with MPC algorithms, to protect against privacy leakage for both the intermediate execution as well as the output of the protocol. For this reason, we also consider adversaries seeking to learn information about data owners' local datasets by observing the output of the model, as done in membership inference [19], [85], [99] and property inference attacks [42], [87], [102]. Recent works have used data poisoning as a tool to further increase privacy leakage [21], [69], [89] of the trained models. Consequently, we can extend our threat model to accommodate a stronger version of $\mathcal{A}^{\text{p}}_{\text{soc}}$ that is also capable of performing privacy attacks by observing the output of the trained model.

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 $\mathcal{A}^{\text{p}}_{\text{soc}}$ adversary. 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-C for a discussion of these approaches), we instead show that it is possible to take advantage of the bounded poisoning capability of $\mathcal{A}^{\text{p}}_{\text{soc}}$ 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 protocol which uses ensemble models to realize our threat model and provide security against $\mathcal{A}^{\text{p}}_{\text{soc}}$. 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 inference 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.

In the inference phase, SafeNet implements protocol Π_{pred}, to compute 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. An optional feature of SafeNet is to add noise to the majority vote to enable user-level differential privacy protection, in addition to poisoning robustness.

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 6. 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

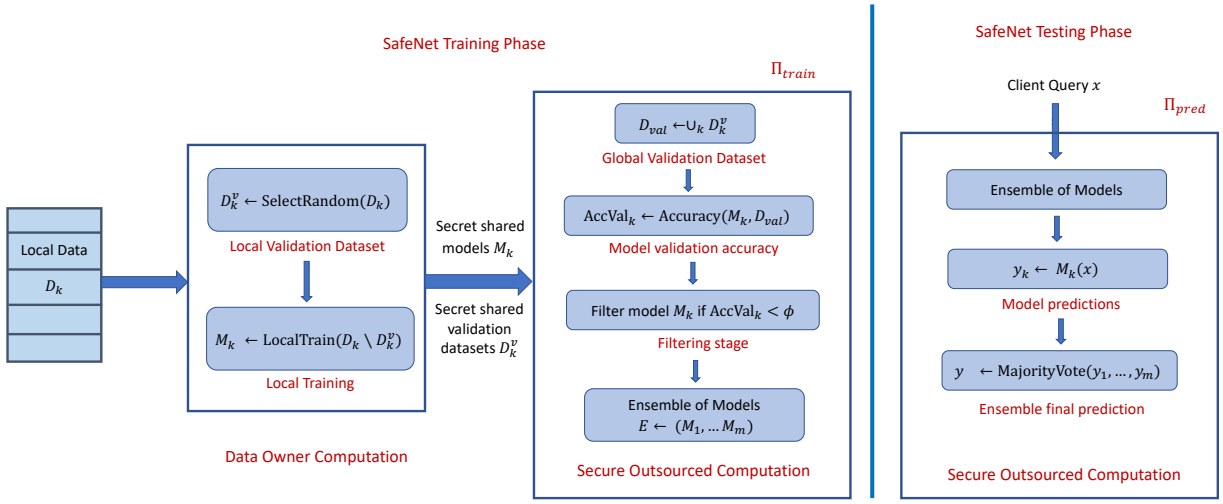

Fig. 3: Overview of the Training and Inference phases of the SafeNet Framework.

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 inference in SafeNet.

C. SafeNet Training and Inference

To train the ensemble in SafeNet, we present our proposed ensemble method in Algorithm 1. We discuss the realization in MPC later in Appendix B. Each owner C_k separates out a subset of its training dataset $D_k^{\text{v}} \in D_k$ and then trains its model $\mathcal{M}_k$ on the remaining dataset $D_k \setminus D_k^{\text{v}}$. The trained model $\mathcal{M}_k$ and validation dataset D_k^{v} is then secret-shared to the servers. The combined validation dataset is denoted as $D_{\text{val}} = \bigcup_{i=1}^{m} D_i^{\text{v}}$. We assume that all users contribute equal-size 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 Appendix A, 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 (described in Appendix B), we argue security as follows:

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 $D_k \setminus D_k^{\text{v}}$.
- Secret-share D_k^{v} and $\mathcal{M}_k$ to servers.

// MPC computation in secret-shared format

– Construct a common validation dataset $D_{\text{val}} = \cup_{i=1}^{m} D_i^{\text{v}}$.

– Construct 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

- $\text{AccVal}_k = Accuracy(\mathcal{M}_k, D_{\text{val}})$
- If $\text{AccVal}_k > \phi$: Set $\mathbf{b}_k^{\mathsf{val}} = 1$

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

Theorem 2. *Given a privacy-preserving machine learning training framework that tolerates T out of N server corruptions, and an error bound of at most p on each clean model in the ensemble, protocol Π_{train} is secure against adversary $\mathcal{A}_{soc}^{p}$ who poisons $t < \frac{m}{2}\frac{1-2p}{1-p}$ out of m data owners and corrupts T out of N servers.*

The proof of the theorem is given in Appendix C, after we introduce the details of MPC instantiation and how Π_{train} securely realizes $\mathcal{F}_{\mathsf{pTrain}}$ in Appendix B-3. During inference, the prediction of each model $\mathcal{M}_k$ is generated and the servers aggregate the results to perform majority voting. Optionally, differentially private noise is added to the sum to offer user-level privacy guarantees. The secure inference protocol Π_{pred} and its security proof is given in Appendix B, C respectively.

D. SafeNet Analysis

Here, we demonstrate the accuracy, poisoning robustness and privacy guarantees that SafeNet provides. We first show how to lower bound SafeNet's test accuracy given that each

clean model in the ensemble reaches a certain accuracy level. We also give certified robustness and user-level privacy guarantees. All of our guarantees improve as the individual models become more accurate, making the ensemble agree on correct predictions more frequently.

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

Theorem. (Informal) *Let $\mathcal{A}^p_{soc}$ be an adversary who poisons at most t out of m data owners and corrupts T out of N servers. Assume that the filtered ensemble E has at least $m - t$ clean models, each with a maximum error rate of $p < 1 - \phi$. If the number of poisoned owners is at most $\frac{m(1-2p)}{2(1-p)}$, ensemble E correctly classifies new samples with high probability, which is a function of m, ϕ, t and p.*

The formal theorem and the corresponding proof can be found in Appendix A.

Poisoning Robustness Analysis. Our previous theorem demonstrated that SafeNet's accuracy on in-distribution data is not compromised by poisoning. Now, we 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 [29]. 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. Sometimes, the algorithm will return $t = 0$, in which case SafeNet provides no robustness for this prediction.

Algorithm 2 Certified Private Prediction PREDGAP (E, x)

Input: m data owners; Ensemble of models $E = \{\mathcal{M}_i\}_{i=1}^m$; Testing point x; Differential Privacy parameters ε, δ.
COUNTS $= \sum_{i=1}^m \mathcal{M}_i(x)$ + DPNOISE(ε, δ)
y, c_y = MOSTCOMMON(COUNTS) // most common predicted class with noisy count
$y', c_{y'}$ = SECONDMOSTCOMMON(COUNTS) // second most common predicted class with count
$t = \lceil (c_y - c_{y'})/2 \rceil - 1$
return y, t

We first analyze the poisoning robustness when privacy of aggregation is not enabled in the following theorem.

Theorem 3. *Let E be an ensemble of models produced in the presence of an adversary $\mathcal{A}^p_{soc}$. On an input x, the ensemble generates prediction $y = E(x)$ without* DPNOISE *and Algorithm 2 outputs (y, t). Then, if $\mathcal{A}^p_{soc}$ poisoned at most t data owners, poisoning cannot have changed E's prediction.*

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. □

Privacy Analysis. Recent work by McMahan et al. [70] introduced the notion of *user-level* differential privacy where the presence of a user in the protocol should have imperceptible impact on the final trained model. We show that, given our threat model, SafeNet provides the strong privacy guarantee of user-level differential privacy, which also implies example-level differential privacy. This privacy guarantee can protect against model extraction and property inference attacks, in addition to membership inference attacks.

Theorem 4. *When* DPNOISE *function samples from a Laplace random variable Lap$(2/\varepsilon)$, Algorithm 2 satisfies user-level ε-differential privacy.*

Proof. Observe that replacing a local model obtained from a data owner in our framework only changes COUNTS for two classes by 1 on any given query, so it has an ℓ_1 sensitivity of 2. As a result, Lap$(2/\varepsilon)$ suffices to ensure that user-level ε-differential privacy holds. □

The main crux of Theorem 4 is that no model can influence COUNTS too much, an observation also made by PATE [79] and the CaPC [27] framework, but they only considered example-level differential privacy, protecting against membership inference attacks, but not stronger attacks that user-level differential privacy prevents. This limitation is inherent in PATE, as the central training set is split to train multiple models. However, our stronger analysis holds for SafeNet in the private collaborative learning setting, as we start with pre-existing partitions of benign and poisoned datasets. We prove Theorem 4 by considering Laplace noise, but various improvements to PATE using different mechanisms such as Gaussian noise and other data-dependent approaches [79], [80], can also be extended to our framework.

Combining Robustness and Privacy. Adding differentially private noise prevents Algorithm 2 from returning the exact difference between the top two class-label counts, making it only possible to offer probabilistic robustness guarantees. That is, the returned t is actually a noisy version of the "true" t^*, where t^* is used to certify correctness. However, for several choices of the DPNoise function, the exact distribution of the noise is known, making it easy to provide precise probabilistic guarantees similar to those provided by Theorem 3. For example, if Gaussian noise with scale parameter σ is used to guarantee DP, and PredGap returns t, then this prediction observed t, then we know that the true t^* is larger than $t - k$ with probability $\Phi(k/\sigma)$, where Φ denotes the Gaussian CDF.

Also note that prior work has shown that, for *local* differentially private protocols, the expectation that users will correctly

randomize their messages leads to increased vulnerability [18], [26]. Our protocol avoids this vulnerability by generating noise in the MPC, not trusting any individual user to generate noise.

E. Extensions

In addition to providing various guarantees, we offer a number of extensions to our original SafeNet design.

Transfer Learning. A major disadvantage of SafeNet is its slower inference time compared to a traditional PPML framework, requiring to perform a forward pass on all local models in the ensemble. However, for transfer learning scenario, we propose a way where SafeNet runs almost as fast as the traditional framework. In transfer learning [37], [59], a pre-trained model $\mathcal{M}_B$, which is typically trained on a large public dataset, is used as a "feature extractor" to improve training on a given target dataset. In our setting, all data owners start with a common pre-trained model, and construct their local models by fine tuning $\mathcal{M}_B$'s last 'l' layers using their local data. We can then modify the prediction phase of SafeNet to reduce its inference time and cost considerably. The crucial observation is that all local models differ only in the weights associated to the last l layers. Consequently, given a prediction query, we run $\mathcal{M}_B$ upto its last l layers and use its output to compute the l layers of all the local models to obtain predictions for majority voting. The detailed description of the modified SafeNet algorithm is given in Appendix D-A. Note that, this approach achieves the same robustness and privacy guarantees as described in Section III-D, as long as $\mathcal{M}_B$ was originally not tampered with. If $\mathcal{M}_B$ is untrustworthy, it may already be backdoored [67], or may be easier to run other types of attacks [94].

Integration Testing. While SafeNet can handle settings with non-iid data distributions among data owners, the local models accuracies might be impacted by extreme non-iid settings (we analyze the sensitivity of SafeNet to data imbalance in Section IV-H). In such cases, SafeNet *fails fast*, allowing the owners to determine whether or not using SafeNet is the right approach for their setting. This is possible because SafeNet's training phase is very cheap, making it possible to quickly evaluate the ensemble's accuracy on the global validation set. If the accuracy is not good enough, the owners can use a different approach, such as a standard MPC training. SafeNet's strong robustness guarantees and an efficient training phase makes it an appealing first choice for private collaborative learning.

It is possible for this filtering to be circumvented if the adversary can participate multiple times [12]. However, this is straightforward to mitigate by bounding the number of submissions. Furthermore, should an adversarial model slip into the ensemble, the robustness guarantees of Theorem 3 will still hold, but with one extra poisoned model.

Low Resource Owners. If a data owner does not have sufficient resources to train a model on their data, they cannot participate in the standard SafeNet protocol. In such situations, computationally restricted owners can defer their training to SafeNet, that can use standard MPC training approaches to train their models. Training these models in MPC increases the computational overhead of our approach, but facilitates broader participation. We provide the details of this modification in Appendix D-B and also run an experiment in Appendix E-A to verify that SafeNet remains efficient, while retaining the same robustness and privacy properties.

F. Comparison to Existing Ensemble Strategies

Model ensembles have been considered to address adversarial machine learning vulnerabilities in several prior works. Here, we discuss the differences between our analysis and previous ensembling approaches.

a) Ensembles on a Centralized Training Set: Several ensemble strategies seek to train a model on a single, centralized training set. This includes using ensembles to prevent poisoning attacks [54], [63], as well as to provide differential privacy guarantees [79] or robustness to privacy attacks [88]. Due to centralization, none of these techniques can take advantage of the partitioning of datasets. As a result, protection from poisoning is only capable of handling a small number of poisoning examples, whereas our partitioning allows large fractions of the entire dataset to be corrupted. PATE, due to data centralization, can only guarantee privacy for individual samples, whereas in our analysis, the *entire dataset* of a given owner can be changed, providing us with *user-level* privacy.

b) CaPC [27]: Chouquette-Choo et al. [27] propose CaPC, which extends PATE to the MPC collaborative learning setting. Their analysis gives differential privacy guarantees for individual examples. Our approach extends their analysis to a differential privacy guarantee for the entire local training set and model, to provide protection against attacks such as property inference and model extraction. In addition, our approach also provides poisoning robustness guarantees which they cannot, as they allow information to be shared between local training sets.

c) Cao et al. [17]: Recent work by Cao et al. [17] gave provable poisoning robustness guarantees for federated learning aggregation. They proposed an ensembling strategy, where, given m data owners, t of which are malicious, they construct an ensemble of $\binom{m}{k}$ global models, where each model is trained on a dataset collected from a set of k clients. Our poisoning robustness argument in Theorem 3 coincides with theirs at $k = 1$, a setting they do not consider as their approach relies on combining client datasets for federated learning. Additionally, $k = 1$ makes their approach vulnerable to data reconstruction attacks [13], an issue SafeNet does not face as the attack directly violates the underlying security guarantee of the MPC. We experimentally compare both approaches on a federated learning dataset in Section V-D and show that our approach outperforms [17].

IV. Evaluation

A. Experimental Setup

We build a functional code on top of the MP-SPDZ library [56][1] to assess the impact of data poisoning attacks on

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

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], [35]; ii) three-party with one semi-honest corruption (3PC) based on Araki et al. [4] with optimizations by [32], [73]; iii) three-party with one malicious corruption based on Dalskov et al. [31]; and iv) four-party with one malicious corruption (4PC), also based on [31]. Note, that both semi-honest and malicious adversaries possess poisoning capability; their roles change only inside the SOC paradigm.

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. [31], [32] is applied after every multiplication. In the MPC library implementation, the sigmoid function for computing the output probabilities is replaced with a three-part approximation [22], [31], [75]. In SafeNet, models are trained locally using the original sigmoid function. We implement softmax function using the method of Aly et al. [2]. 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. is the time taken to privately train a model inside the MPC (protocol Π_{train}). As is standard practice [14], [22], [23], [31], [73], [75], 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. is the amount of data exchanged between the servers during the privacy-preserving execution of the training phase.

Test Accuracy. is the percentage of test samples that the ML model correctly predicts.

Attack Success Rate. is the 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 *adaptively* select any subset of c owners, arbitrarily modifying the model to change the test sample's classification. This is the same adversary considered in Algorithm 2 and by Theorem 3, any 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 descriptions of the datasets and models used for our experiments below.

MNIST. The MNIST dataset [38] is a 10 class classification problem which is used to predict digits between 0 and 9. We train a logistic regression model for MNIST.

Adult. The Adult dataset [38] 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 activations.

Fashion. We train several neural networks on the Fashion dataset [96] 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.

CIFAR-10. The CIFAR-10 dataset [60] is a 10 class image dataset. CIFAR-10 is harder than other datasets we consider, so we perform transfer learning from a ResNet-50 model [48] pretrained on the ImageNet dataset [36]. We fine tune only the last layer, freezing all convolutional layers.

EMNIST. The EMNIST dataset [28] is a benchmark federated learning image dataset, split in a non-iid fashion by the person who drew a given image. We select 100 EMNIST clients in our experiments.

D. Dataset Partitioning and Model Accuracy

We conduct our experiments by varying the number of data owners. We split MNIST and Adult datasets across 20 participating data owners, while we use 10 owners for Fashion and CIFAR-10 datsets. The EMNIST dataset used for comparison with prior work on federated learning assumes 100 participating owners. Each owner selects at random 10% of its local training data as the validation dataset D_j^{v}. All models are trained using mini-batch stochastic gradient descent.

To introduce non-iid behavior in our datasets (except for EMNIST, which is naturally non-iid), we sample class labels from a Dirichlet distribution [49]. That is, 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. In practice, we observe $\alpha = 1000$ leads to almost iid behavior, while $\alpha = 0.1$ results in an extreme imbalance distribution. The default choice for all our experiments is $\alpha = 10$, which provides a realistic non-iid distribution. We will vary parameter α in Appendix E-A.

Dataset	Partition Type	Local Model	SafeNet Ensemble	Improvement
MNIST	Dirchlet	80.05%	89.48%	9.03%
Adult		77.32%	81.41%	4.09%
FASHION		71.68%	83.26%	11.53%
CIFAR-10		54.03%	62.76%	8.73%
EMNIST	Natural	54.05%	79.19%	25.14%

TABLE I: Test accuracy comparison of a single local model and the entire SafeNet ensemble. SafeNet Ensemble improves upon a single local model across all datasets.

We measure the accuracy of a local model trained by individual data owners and our SafeNet ensemble. Table I provides the detailed comparison of the accuracy of the local and ensemble models across all four datasets. We observe that SafeNet consistently outperforms local models, with improvements ranging from 4.09% to 25.14%. The lowest performance is on CIFAR-10, but in this case SafeNet's accuracy is very close to fine-tuning the network using the combined dataset, which reaches 65% accuracy.

E. Implementation of Poisoning Attacks

Backdoor Attacks. We use the BadNets attack by Gu et al. [47], 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 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.

To run backdoor attacks on models trained with standard PPML frameworks, 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, backdoor attacks are implemented at the poisoned owners, which 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 be easy to filter due to low accuracy, making training on clean samples necessary. 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 [57], and can be seen to be the output of the state-of-the-art Witches' Brew targeted attack when unconstrained (Witches' Brew considers clean-label poisoned samples) [43].

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 examples classified correctly with high confidence by a different model. Because these examples are easy to classify, poisoning them should be hard. We also consider an attack called TGT-Foot, which chooses low confidence examples, which are easier to poison. For both strategies, the adversary replaces its label with the second highest predicted label. 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.

F. Evaluation on Logistic Regression

We start with DIGIT 1/7 dataset, a subset of MNIST data using only digits 1 and 7, for which we evaluate the computational costs and 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 II 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 SafeNet 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.

MPC	Setting	Framework	Training (s)	Comm. (GB)
2PC	Semi-Honest [35]	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 [31]	PPML	n×32.54	n× 2.32
		SafeNet	9.44	1.47
4PC	Malicious [31]	PPML	n×5.28	n×0.66
		SafeNet	1.09	0.28

TABLE II: 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.

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 4 (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 poisoned owners to 10.

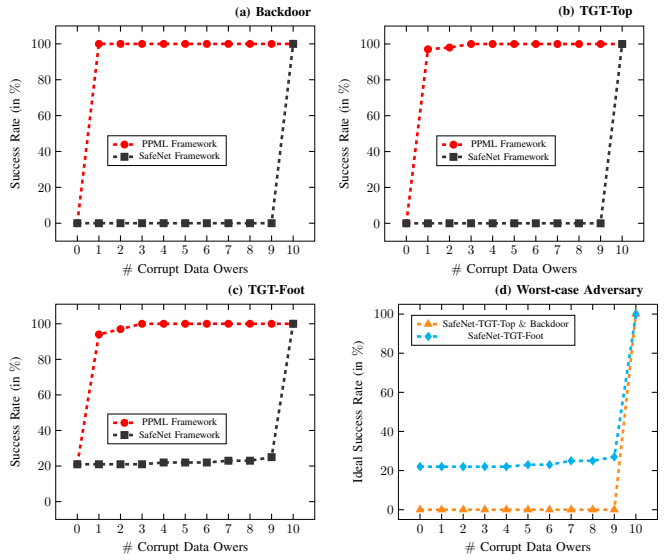

Fig. 4: Logistic regression attack success rate on the Digit-1/7 dataset for PPML and SafeNet frameworks in the 3PC setting, for varying poisoned owners launching Backdoor and Targeted attacks. Plot (a) gives the success rate for the BadNets attack, while plots (b) and (c) show the success rates for 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.

We observe in Figure 4 (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, SafeNet works as intended even at high levels of poisoning. For the TGT-Foot attack in Figure 4 (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 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 4 (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 training 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 E.

G. 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 E.

Table III provides a detailed analysis of the training time, communication, test accuracy and success rate for the 4PC PPML framework and SafeNet using one poisoned owner. We observe that SafeNet has $39\times$ and $36\times$ improvement in training time and communication complexity over the PPML framework, for $n = 10$ epochs. The SafeNet prediction time is 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. We believe this is a reasonable cost for many applications, as SafeNet has significant training time improvements and robustness guarantees.

For the BadNets backdoor attack we set the true label y_s as a 'T-Shirt' and target label y_t as 'Trouser'. We test the effect of both TGT-Top and TGT-Foot attacks under multiple poisoned owners, and also evaluate another variant of targeted attack called TGT-Random, where we randomly sample $k = 100$ target samples from the test data. Figure 5 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.

H. Evaluation of Extensions

Here, we evaluate our SafeNet extensions introduced in Section III-E. First, we experiment with our transfer learning extension. We show that, on applying our extension to SafeNet, its inference overhead falls dramatically. We test our approach on Fashion and CIFAR-10 datasets. For the Fashion dataset, we use the same setup as earlier with $m = 10$ data owners, and three-layered neural network as the model architecture, where each data owner fine-tunes only the last layer ($l = 1$) of the

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 × 565.45	n × 154.79	84.07%	100%	82.27%	100%	100%
		SafeNet	156.53	41.39	84.36%	0%	84.48%	0%	32%
4PC [31]	Malicious	PPML	n × 1392.46	n × 280.32	84.12%	100%	82.34%	100%	100%
		SafeNet	356.26	76.43	84.36%	0%	84.54%	0%	32%

TABLE III: 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.

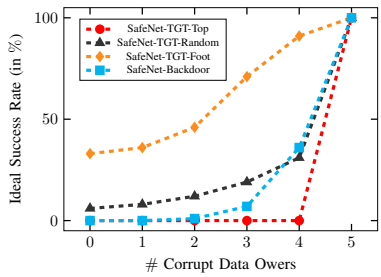

Fig. 5: Worst-case adversarial success against targeted and backdoor attacks of a three-layer neural network trained on Fashion in SafeNet. The adversary can change the set of c poisoned owners per sample. SafeNet achieves robustness on the backdoor, TGT-Top and TGT-Random attacks, up to 4 poisoned owners out of 10. The TGT-Foot attack targeting low-confidence samples has higher success.

pre-trained model. We observe that for each secure inference, SafeNet is now only $1.62\times$ slower and communicates $1.26\times$ more on average than the PPML framework, while the standard SafeNet approach is about $8\times$ slower due to the evaluation of multiple ML models.

We observe even better improvements for CIFAR-10 dataset. Here, we use a state-of-the-art 3PC inference protocol from [61], built specially for ResNet models. In our setting, each owner fine-tunes the last layer of a ResNet-50 model, which was pre-trained on ImageNet data. SafeNet reaches 62.8% accuracy, decaying smoothly in the presence of poisoning: 51.9% accuracy tolerating a single poisoned owner, and 39.8% while tolerating two poisoned owners. The cost of inference for a single model is an average of 59.9s, and SafeNet's overhead is negligible (experimental noise has a larger impact than SafeNet); SafeNet increases communication by only 0.1%, increasing around 7MB over the 6.5GB required for standard inference.

Next, we analyze the behavior of SafeNet under different non-iid settings by varying the concentration parameter α. We use the same Fashion dataset setup from Section IV-G. We observe that as α decreases, i.e., the underlying data distribution of the owners become more non-iid, SafeNet's accuracy decreases, as expected, but SafeNet still achieves reasonable robustness even under high data imbalance (e.g., $\alpha = 1$). In extremely imbalanced settings, such as $\alpha = 0.1$, SafeNet can identify low accuracy during training and data owners can take actions accordingly. We defer the details for this extension to Appendix E-A, which also includes analyzing attack success rates under extreme non-iid conditions.

V. Discussion and Comparison

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 and comparison against other mitigation strategies and federated learning approaches.

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 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. However, we discussed how transfer learning can reduce the inference time of SafeNet.

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.

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 6 to achieve poisoning robustness. The owners agree upon the number of outsourced servers N participating in the MPC, the number of corrupted servers T along with the role of the adversary (semi-honest or malicious) in the MPC and consequently choose an appropriate training framework that satisfies this criteria. The owners then 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 5 and Theorem 6, 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.

We provide a concrete example on parameter selection as follows: We instantiate our Fashion dataset setup, with $m = 10$ data owners participating in SafeNet. For the MPC, we set $N = 3$ servers, tolerating $T = 1$ corruption in the presence of a malicious adversary and choose [31] as our privacy-preserving training framework. For poisoning robustness, we set $\phi = 0.3$ and the number of poisoned owners to $t = 2$. 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 clean models are filtered during training and the attack success rate of the adversary for backdoor attacks remains the same even after poisoning 3 owners, while our analysis holds for $t = 2$ poisoned owners. Thus, in practice SafeNet is able tolerate more poisoning than our suggested analysis.

C. 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 [24], [66], [90], [93], but these strategies work only for Convolutional Neural Networks trained on image datasets; Severi et al. [84] showed that these approaches fail when tested on other data modalities and models. Furthermore, recent work by Goldwasser et.al [45] 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 poison only a subset of models, making the ensemble robust against poisoning. In certain instances SafeNet 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], [40], which are possible when data owners train their models locally. SafeNet tolerates model poisoning because each model only contributes to a single vote towards the final ensemble prediction. In fact, all our empirical and theoretical analysis of SafeNet is computed for arbitrarily corrupted models.

D. 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 [41], [55], [103], but this involves large communication overhead and is susceptible to poisoning [68]. For example, recent work [6], [8], [97] 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 [101], coordinate-wise mean [100] and Krum [11], which have been show to be susceptible to backdoor and model poisoning attacks [40]. Recent work in FL such as FLTrust [16] and DeepSight [83] 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 [13]. SafeNet prevents such an attack, as it directly violates the security guarantee of the MPC, when instantiated for the malicious setting.

We experimentally compare SafeNet to the federated learning-based approach of Cao et al. [17], who also gave provable robustness guarantees in the federated averaging scenario. We instantiate their strategy for EMNIST dataset and compare their Certified Accuracy metric to SafeNet's, with $m = 100$ data owners, $k = \{2, 4\}$ and FedAvg as the base algorithm. To ensure both approaches have similar inference times, we fix the ensemble size to 100 models, each trained using federated learning with 50 global and local iterations.

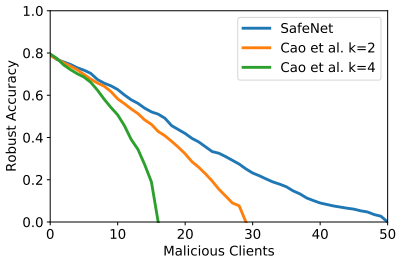

Fig. 6: Certified Accuracy of our framework compared to Cao et al. [17]. We fix the size of the Cao et al. ensemble to 100, to match the test runtime of SafeNet.

Figure 6 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 also particularly expensive at training time when instantiated in MPC. During training, their approach requires data owners to interact inside MPC to train models over multiple rounds. By contrast, SafeNet only requires interaction in MPC at the beginning of the training phase, making it significantly faster.

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. We then propose SafeNet, which performs ensembling in MPC, and show that our design has provable robustness and privacy guarantees, beyond those offered by existing approaches. We evaluate SafeNet using logistic regression and neural networks models trained on five datasets by varying the distribution similarity across data owners. We consider both end-to-end and transfer learning scenarios. We demonstrate experimentally that SafeNet achieves even higher robustness than its theoretical analysis against backdoor and targeted poisoning attacks, at a significant performance improvement in the training time and communication complexity compared to existing PPML frameworks.

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] Avrim Blum and Moritz Hardt. The ladder: A reliable leaderboard for machine learning competitions. In *International Conference on Machine Learning*, pages 1006–1014. PMLR, 2015.

[13] 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.

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

[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] Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. Data poisoning attacks to local differential privacy protocols. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 947–964, 2021.

[19] N. Carlini, S. Chien, M. Nasr, S. Song, A. Terzis, and F. Tramer. Membership inference attacks from first principles. In *IEEE Symposium on Security and Privacy (SP)*, 2022.

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

[21] H. Chaudhari, J. Abascal, A. Oprea, M. Jagielski, F. Tramèr, and J. Ullman. Snap: Efficient extraction of private properties with poisoning. arXiv, 2022.

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

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

[24] 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.

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

[26] Albert Cheu, Adam Smith, and Jonathan Ullman. Manipulation attacks in local differential privacy. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 883–900. IEEE, 2021.

[27] 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.

[28] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik. Emnist: Extending mnist to handwritten letters. In *International Joint Conference on Neural Networks (IJCNN)*, 2017.

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

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

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

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

[33] 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.

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

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

[36] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

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

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

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

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

[41] 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.

[42] K. Ganju, Q. Wang, W. Yang, C.A. Gunter, and N. Borisov. Property inference attacks on fully connected neural networks using permutation invariant representations. 2018.

[43] 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.

[44] 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.

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

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

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

[48] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[49] T.M.Harry Hsu, H.Qi, and M.Brown. Measuring the effects of non-identical data distribution for federated visual classification. In *IACR ePrint*, 2019.
[50] Y. Ishai, J. Kilian, K. Nissim, and E. Petrank. Extending Oblivious Transfers Efficiently. In *CRYPTO*, 2003.
[51] Y. Ishai, R. Kumaresan, E. Kushilevitz, and A. Paskin-Cherniavsky. Secure computation with minimal interaction, revisited. In *CRYPTO*, 2015.
[52] 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.
[53] J. Jia, X. Cao, and N. Gong. Intrinsic certified robustness of bagging against data poisoning attacks. In *AAAI*, 2021.
[54] Jinyuan Jia, Xiaoyu Cao, and Neil Zhenqiang Gong. Intrinsic certified robustness of bagging against data poisoning attacks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 7961–7969, 2021.
[55] 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.
[56] M. Keller. MP-SPDZ: A versatile framework for multi-party computation. In *ACM CCS*, 2020.
[57] Pang Wei Koh, Jacob Steinhardt, and Percy Liang. Stronger data poisoning attacks break data sanitization defenses. *Machine Learning*, 111(1):1–47, 2022.
[58] P.W. Koh and P. Liang. Understanding black-box predictions via influence functions. In *ICML*, 2017.
[59] S. Kornblith, J. Shlens, and Q.V. Le. Do better imagenet models transfer better? In *CVPR*, 2019.
[60] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.
[61] N. Kumar, M. Rathee, N. Chandran, D. Gupta, A. Rastogi, and R. Sharma. Cryptflow: Secure tensorflow inference. In *IEEE Security & Privacy*, 2020.
[62] R. Lehmkuhl, P. Mishra, A. Srinivasan, and R.A. Popa. Muse: Secure inference resilient to malicious clients. In *USENIX*, 2021.
[63] Alexander Levine and Soheil Feizi. Deep partition aggregation: Provable defense against general poisoning attacks. *arXiv preprint arXiv:2006.14768*, 2020.
[64] Y. Lindell. Fast cut-and-choose-based protocols for malicious and covert adversaries. In *J. Cryptology*, 2016.
[65] Y. Lindell and B. Pinkas. An efficient protocol for secure two-party computation in the presence of malicious adversaries. In *EUROCRYPT*, 2007.
[66] K. Liu, B. Dolan, and S. Garg. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *RAID*, 2018.
[67] Yuntao Liu, Yang Xie, and Ankur Srivastava. Neural trojans. In *2017 IEEE International Conference on Computer Design (ICCD)*, pages 45–48. IEEE, 2017.
[68] 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.
[69] S. Mahloujifar, E. Ghosh, and M. Chase. Property inference from poisoning. In *IEEE Symposium on Security and Privacy (SP)*, 2022.
[70] H.B McMahan, D. Ramage, K. Talwar, and L. Zhang. Learning differentially private recurrent language models. In *ICLR*, 2018.
[71] P. Mishra, R. Lehmkuhl, A. Srinivasan, W. Zheng, and R.A. Popa. Delphi: A cryptographic inference service for neural networks. In *USENIX*, 2020.
[72] P. Mohassel and M. K. Franklin. Efficiency tradeoffs for malicious two-party computation. In *PKC*, 2006.
[73] P. Mohassel and P. Rindal. ABY^3: A Mixed Protocol Framework for Machine Learning. In *ACM CCS*, 2018.
[74] P. Mohassel, M. Rosulek, and Y. Zhang. Fast and Secure Three-party Computation: Garbled Circuit Approach. In *CCS*, 2015.
[75] P. Mohassel and Y. Zhang. Secureml: A system for scalable privacy-preserving machine learning. In *IEEE S&P*, 2017.
[76] 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.
[77] J. Newsome, B. Karp, and D. Song. Paragraph: Thwarting signature learning by training maliciously. In *RAID*, 2006.
[78] J. B. Nielsen and C. Orlandi. Cross and clean: Amortized garbled circuits with constant overhead. In *TCC*, 2016.
[79] N. Papernot, M. Abadi, Ú. Erlingsson, I. Goodfellow, and K. Talwar. Semi-supervised knowledge transfer for deep learning from private training data. In *ICLR*, 2017.
[80] N. Papernot, S. Song, I. Mironov, A. Raghunathan, K. Talwar, and Ú. Erlingsson. Scalable private learning with pate. 2018.
[81] A. Patra, T. Schneider, A. Suresh, and H. Yalame. Aby2.0: Improved mixed-protocol secure two-party computation. In *USENIX*, 2021.
[82] 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.
[83] P. Rieger, T. Nguyen, M. Miettinen, and A. Sadeghi. Deepsight: Mitigating backdoor attacks in federated learning through deep model inspection. In *NDSS*, 2022.
[84] G. Severi, J. Meyer, S. Coull, and A. Oprea. Explanation-guided backdoor poisoning attacks against malware classifiers. In *USENIX*, 2021.
[85] R. Shokri, M. Stronati, and V. Song, C.and Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017.
[86] 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.
[87] A. Suri and D. Evans. Formalizing and estimating distribution inference risks. *Proceedings on Privacy Enhancing Technologies (PETS)*, 2022.
[88] X. Tang, S. Mahloujifar, L. Song, V. Shejwalkar, M. Nasr, A. Houmansadr, and P. Mittal. Mitigating membership inference attacks by {Self-Distillation} through a novel ensemble architecture. In *31st USENIX Security Symposium*, 2022.
[89] F. Tramèr, R. Shokri, A.S. Joaquin, H. Le, M. Jagielski, S. Hong, and N. Carlini. Truth Serum: Poisoning machine learning models to reveal their secrets. In *ACM Computer and Communications Security (CCS)*, 2022.
[90] B. Tran, J. Li, and A. Madry. Spectral signatures in backdoor attacks. In *NeurIPS*, 2018.
[91] S. Wagh, D. Gupta, and N. Chandran. SecureNN: Efficient and private neural network training. In *PoPETS*, 2019.
[92] 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.
[93] 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.
[94] Bolun Wang, Yuanshun Yao, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. With great training comes great vulnerability: Practical attacks against transfer learning. In *27th USENIX security symposium (USENIX Security 18)*, pages 1281–1297, 2018.
[95] J.L. Watson, S. Wagh, and R.A. Popa. Piranha: A gpu platform for secure computation. In *USENIX*, 2022.
[96] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.
[97] C. Xie, S. Koyejo, and I. Gupta. Fall of empires: Breaking byzantine-tolerant SGD by inner product manipulation. In *UAI*, 2019.
[98] A. C. Yao. Protocols for Secure Computations. In *FOCS*, 1982.
[99] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha. Privacy risk in machine learning: Analyzing the connection to overfitting. In *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*. IEEE, 2018.
[100] D. Yin, Y. Chen, K. Ramchandran, and P. Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. In *ICML*, 2018.
[101] D. Yin, Y. Chen, K. Ramchandran, and P. Bartlett. Defending against saddle point attack in byzantine-robust distributed learning. In *ICML*, 2019.
[102] W. Zhang, S. Tople, and O. Ohrimenko. Leakage of dataset properties in Multi-Party machine learning. In *30th USENIX Security Symposium*, 2021.
[103] H. Zhu, R. Mong Goh, and W. Ng. Privacy-preserving weighted federated learning within the secret sharing framework. In *IEEE Access*, 2020.

APPENDIX A
SAFENET 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 5. *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 ith 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 $\delta > 0$.

□

As a visual interpretation of Lemma 5, Figure 7 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.

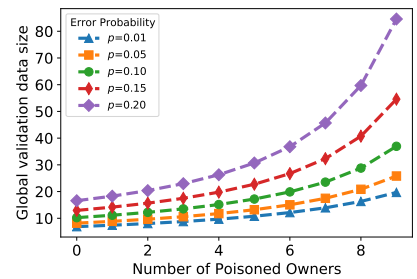

Fig. 7: Minimum number of samples in the validation dataset as a function of maximum error probability p and number of poisoned owners t for $m = 20$ data owners. We set the filtering threshold $\phi = 0.03$ and failure probability $\epsilon = 0.01$.

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 6. *Assume that the conditions in Lemma 5 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 5 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 $X \sim \text{Bin}(m-t, p)$. 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 5. 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
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 [4], [14], [22], [23], [35], [46]. 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 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 OP, where 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 [14], [73], [75], [92]:

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 **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 8 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 $[\![D_{\mathsf{val}}]\!] = \cup_{k=1}^{m}[\![D_k^{\mathsf{v}}]\!]$ 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 $[\![\mathrm{AccVal}_k]\!]$. The output $[\![\mathrm{AccVal}_k]\!]$ 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 $\mathrm{AccVal}_k > \phi$.

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 $D_{\mathsf{val}} = \cup_{k=1}^m D_k^{\mathsf{v}}$.
- 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 $[\![D_{\mathsf{val}}]\!]$ and obtain $[\![\mathrm{PREDS}_k]\!]$, where $\mathrm{PREDS}_k = \mathcal{M}_k(D_{\mathsf{val}})$
  - Execute Π_{acc} with inputs as $[\![\mathrm{PREDS}_k]\!]$ and $[\![\mathbf{y}_{D_{\mathsf{val}}}]\!]$ and obtain $[\![\mathrm{AccVal}_k]\!]$ as the output.
  - Locally subtract $[\![\cdot]\!]$-shares of AccVal_k with ϕ to obtain $[\![\mathrm{AccVal}_k - \phi]\!]$.
  - Execute Π_{comp} with input as $[\![\mathrm{AccVal}_k - \phi]\!]$ and obtain $[\![b']\!]$, where $b' = 1$ iff $\mathrm{AccVal}_k > \phi$. Set the kth 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. 8: SafeNet Training Protocol

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

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 **Preds**.
  - Locally add $[\![\mathbf{z}]\!] = [\![\mathbf{z}]\!] + [\![\mathbf{Preds}]\!]$ to update $\mathbf{z}$.
- Execute Π_{amx} protocol with input as $[\![\mathbf{z}]\!]$ and obtain $[\![\mathrm{OP}]\!]$ as the output.

Output: $[\![\cdot]\!]$-shares of OP

Fig. 9: 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 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 **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 **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 $[\![\mathrm{OP}]\!]$ from Π_{amx}, where OP denotes the predicted class label by the ensemble. The appropriate $[\![\cdot]\!]$-shares of OP is forwarded to the client for reconstruction.

Theorem 7. *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. The proof is given below in Appendix C. □

Appendix C
Security Proofs

For concise security proofs, we assume the adversary $\mathcal{A}^{\mathrm{p}}_{\mathrm{soc}}$ performs a semi-honest corruption in the SOC paradigm, but our proofs can also be extended to malicious adversaries in the MPC. We prove that protocol Π_{train} is secure against an adversary of type $\mathcal{A}^{\mathrm{p}}_{\mathrm{soc}}$. 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 $\mathcal{A}^{\mathrm{p}}_{\mathrm{soc}}$.

Theorem 2. *Given a privacy-preserving machine learning training framework that tolerates T out of N server corruptions, and an error bound of at most p on each clean model in the ensemble, protocol Π_{train} is secure against adversary $\mathcal{A}^p_{soc}$ who poisons $t < \frac{m}{2}\frac{1-2p}{1-p}$ out of m data owners and corrupts T out of N servers.*

Proof. Given the training framework securely realizes each of the building block used in protocol Π_{train}, we now argue Π_{train} securely realizes functionality $\mathcal{F}_{\mathsf{pTrain}}$. Let $\mathcal{A}^{\mathrm{p}}_{\mathrm{soc}}$ 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 $\mathcal{A}^{\mathrm{p}}_{\mathrm{soc}}$. Note that, in the semi-honest setting $\mathcal{S}_f$ already posses the input of $\mathcal{A}^{\mathrm{p}}_{\mathrm{soc}}$ 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} 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 $\mathcal{A}^{\mathrm{p}}_{\mathrm{soc}}$.
- 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 $\mathcal{A}^{\mathrm{p}}_{\mathrm{soc}}$.

– $\mathcal{S}_f$ 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 $\mathbf{y}_{D_{\text{val}}}$ and eventually sends and receives appropriate $[\![\cdot]\!]$-shares of AccVal_k to and from $\mathcal{A}^{\text{p}}_{\text{soc}}$.
– 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 $\text{AccVal} - \phi$ and at the end $\mathcal{S}_f$ instead sends the original shares of $\mathbf{b}^{\mathsf{val}}_k$ as shares of b' associated to $\mathcal{A}^{\text{p}}_{\text{soc}}$.
– No simulation is required to assign $[\![\mathbf{b}^{\mathsf{val}}_k]\!] = [\![b']\!]$.

The proof now follows from the fact that simulated view and real-world view of the adversary are computationally indistinguishable and 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 6 that E correctly classifies a sample with probability at least p_c. As a result the underlying trained model also provides poisoning robustness against $\mathcal{A}^{\text{p}}_{\text{soc}}$.

□

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

Theorem 7. *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 $\mathcal{A}^{\text{p}}_{\text{soc}}$ 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 $\mathcal{A}^{\text{p}}_{\text{soc}}$. $\mathcal{S}_f$ 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 $\mathcal{A}^{\text{p}}_{\text{soc}}$.
– 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 $\mathcal{A}^{\text{p}}_{\text{soc}}$.
  – 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 $\mathcal{A}^{\text{p}}_{\text{soc}}$.
  – 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 $\mathcal{A}^{\text{p}}_{\text{soc}}$.

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 $\mathcal{A}^{\text{p}}_{\text{soc}}$ in Theorem 2. □

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

Appendix D
SafeNet Extensions

A. Inference phase in Transfer Learning Setting

We provide a modified version of SafeNet's Inference 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 Inference for 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 $\mathbf{x}$ upto its last l layers, where $\mathbf{p}$ denotes the output vector from that layer.
– For $k \in [1, m]$:
- Run forward pass on the last l layers 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. 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 E
Additional Experiments

A. Evaluation of SafeNet Extensions

a) Integration Testing: Here, we evaluate the performance of SafeNet by varying the concentration parameter α to manipulate the degree of data similarity among the owners. The experiments are performed with the same neural network architecture from Section IV-G on the Fashion dataset.

Algorithm 4 SafeNet Training 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 $D_k \setminus D_k^{\mathrm{v}}$ 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 $D_k \setminus D_k^{\mathrm{v}}$.
- 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 $\mathcal{M}_k$ on $D_k \setminus D_k^{\mathrm{v}}$.
2. Construct a common validation dataset $D_{\mathrm{val}} = \cup_{i=1}^{m} D_i^{\mathrm{v}}$ and collect ensemble of models $E = \{\mathcal{M}_i\}_{i=1}^{m}$
3. Initialize a vector $\mathbf{b}^{\mathsf{val}}$ of zeros and of size m.
4. For $k \in [1, m]$:
- $\mathrm{AccVal}_k = Accuracy(\mathcal{M}_k, D_{\mathrm{val}})$
- If $\mathrm{AccVal}_k > \phi$:
  – Set $\mathbf{b}_k^{\mathsf{val}} = 1$

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

Figure 10 gives a comprehensive view of the variation in test accuracy and attack success rate for backdoor and targeted attacks over several values of α.

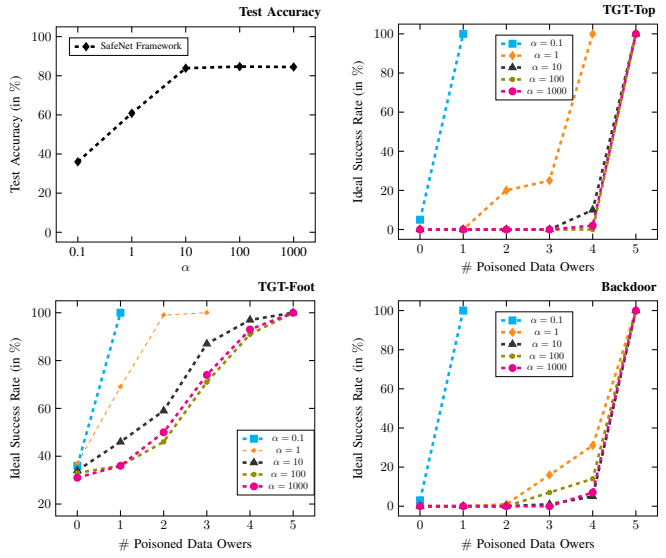

Fig. 10: Test Accuracy and Worst-case Adversarial Success in a three layer neural network model trained on Fashion dataset using SafeNet for varying data distributions. Parameter α dictates the similarity of distributions between the owners. Higher values of α denote greater similarity in data distributions 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 10 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.

b) Low Resource Users: 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 retaining its robustness against poisoning and privacy attacks.

B. 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 IV 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.

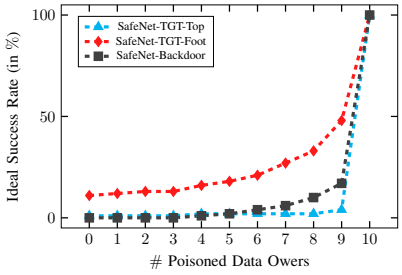

Fig. 11: Worst-case adversarial success of multi-class logistic regression on MNIST in the SafeNet framework for backdoor and targeted attacks. The adversary can change the set of c poisoned owners per sample. SafeNet achieves certified robustness up to 9 poisoned owners out of 20 against backdoor and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

C. Evaluation on Deep Learning Models

Experiments on Fashion Dataset. We present results on one and two layer deep neural networks trained on the Fashion

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 [31]	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%

TABLE IV: 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.

dataset. We perform the same set of backdoor and targeted attacks as described in Section IV. Tables V and VI provide 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× and 43× 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 (attacker can choose any subset of corrupted owners per sample). Results are similar to Figure 5.

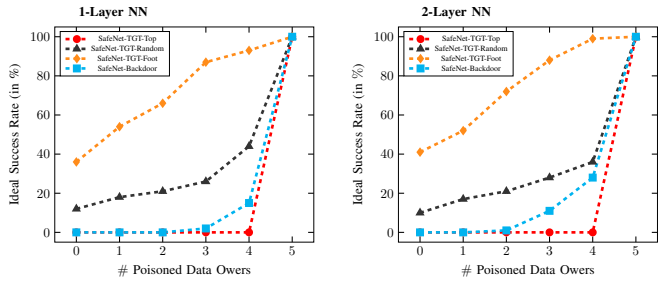

Fig. 12: Worst-case adversarial success of one and two layer Neural Networks on FASHION dataset in SafeNet framework for varying poisoned owners.

MPC	Setting	No. Hidden Layers	Framework	Training Time (s)	Communication (GB)
3PC [4]	Semi-Honest	1	PPML	n×382.34	n× 96.37
			SafeNet	65.71	14.58
		2	PPML	n×474.66	n× 125.58
			SafeNet	108.12	27.98
4PC [31]	Malicious	1	PPML	n×869.12	n× 174.12
			SafeNet	152.68	26.89
		2	PPML	n×1099.06	n×227.23
			SafeNet	258.72	51.66

TABLE V: Training Time (in seconds) and Communication (in GB) of PPML and SafeNet frameworks for one and two layer neural network on Fashion dataset, where n denotes the number of epochs. The time and communication reported for SafeNet framework is for end-to-end execution.

MPC	Setting	No. Hidden Layers	Framework	Test Accuracy	Backdoor Attack	Targeted Attack	
					Success Rate	Success Rate-Top	Success Rate-Foot
3PC [4]	Semi-Honest	1	PPML	82.40%	100%	100%	100%
			SafeNet	84.45%	0%	0%	38%
		2	PPML	83.92%	100%	100%	100%
			SafeNet	84.93%	0%	0%	46%
4PC [31]	Malicious	1	PPML	82.82%	100%	100%	100%
			SafeNet	84.44%	0%	0%	38%
		2	PPML	83.80%	100%	100%	100%
			SafeNet	84.86%	0%	0%	46%

TABLE VI: Test Accuracy and Success Rate of PPML and SafeNet frameworks for one and two layer neural network on Fashion dataset, in presence of a single poisoned owner.

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 [31]	PPML	n×223.15	n×16.49
		SafeNet	179.58	19.29
4PC	Malicious [31]	PPML	n×18.54	n×1.69
		SafeNet	14.67	2.53

TABLE VII: 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.

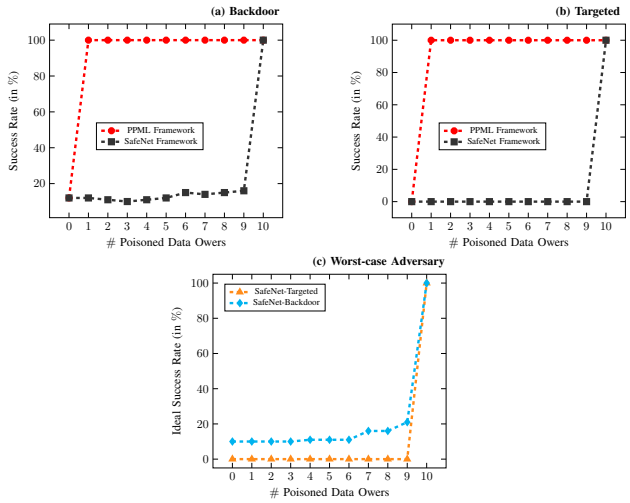

Fig. 13: Attack Success Rate and a Neural Network in PPML and SafeNet frameworks, trained over Adult dataset, for varying corrupt owners launching Backdoor (a) and Targeted (b) attacks. Plot (c) gives the worst-case adversarial success of SafeNet when a different set of poisoned owners is allowed per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

Experiments on Adult Dataset. We use a similar attack strategy as used for logistic regression model in Section IV-E. 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 $\{x_i^t\}_{i=1}^k$ with true label $y_s = 0$, and create backdoored samples $\{Pert(x_i^t), y_t = 1\}_{i=1}^k$, where $Pert(\cdot)$ function sets the capital-loss feature in x_t to 1. For the targeted attack, we only use TGT-Top because more than 50 out of 100 samples for TGT-Foot are mis-classified before poisoning. Table VII 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.