SAFEFL: MPC-friendly Framework for Private and Robust Federated Learning

Till Gehlhar, Felix Marx, Thomas Schneider, Ajith Suresh, Tobias Wehrle and Hossein Yalame
Technical University of Darmstadt, Germany
{till.gehlhar,felix.marx,tobias.wehrle}@stud.tu-darmstadt.de
{schneider,suresh,yalame}@encrypto.cs.tu-darmstadt.de

***Abstract*—Federated learning (FL) has gained widespread popularity in a variety of industries due to its ability to locally train models on devices while preserving privacy. However, FL systems are susceptible to i) *privacy inference attacks* and ii) *poisoning attacks*, which can compromise the system by corrupt actors. Despite a significant amount of work being done to tackle these attacks individually, the combination of these two attacks has received limited attention in the research community.**

To address this gap, we introduce SAFEFL, a secure multiparty computation (MPC)-based framework designed to assess the efficacy of FL techniques in addressing both privacy inference and poisoning attacks. The heart of the SAFEFL framework is a communicator interface that enables PyTorch-based implementations to utilize the well-established MP-SPDZ framework, which implements various MPC protocols. The goal of SAFEFL is to facilitate the development of more efficient FL systems that can effectively address privacy inference and poisoning attacks.

***Index Terms*—Federated Learning, MPC, Privacy**

1. Introduction

Machine learning (ML) has become a widely adopted technology in various industries such as autonomous driving [25], medical diagnosis [50], natural language processing [52], and finance [31]. In traditional ML, the data was collected and centralized, and the model was trained on the entire data set. However, this approach is often not practical due to growing privacy concerns among data owners and regulations such as the General Data Protection Regulation (GDPR) and the California Consumer Privacy Act (CCPA). These regulations limit the collection and use of personal data, making it necessary to find alternative methods for training machine learning models, thus paving the way for privacy-preserving ML (PPML) techniques [11], [28], [40], [45], [68], [70]. Existing PPML solutions use secure computation techniques such as secure multi-party computation (MPC) [51] and homomorphic encryption (HE) [2], and have relatively high communication and computation costs.

In an attempt to reduce the costs while upholding the user's trust, Google introduced Federated Learning (FL) [46], where users train models locally on their devices, and the results are then combined by a central aggregator to update the model (e.g., FedAvg [59]). This approach ensures that data remains on the user's device, creating a trust-based relationship between the user and the system. The popularity of FL has seen a surge in both academic [39] and industrial research [14], leading to the deployment of several real-world solutions [54], [80] due to its numerous benefits. However, despite its potential benefits, FL has been shown to be susceptible to two orthogonal issues caused by the presence of corrupt actors in the system, namely i) *privacy inference* attacks and ii) *poisoning* attacks.

In privacy inference attacks, an adversary who corrupts the model aggregator attempts to infer sensitive information about the users' private data from the updated local models/gradients [60], [65]. To address this issue, secure aggregation (SA) techniques have been proposed, in which users send encrypted local updates to the aggregator, who can only access the combined update rather than individual ones [56].

In poisoning attacks, corrupt users create fraudulent models and send them to the aggregator to manipulate the training process [34], [75], [76], [79]. These crafted models can either reduce the accuracy of the model, making it ineffective, or incorporate a backdoor that changes its predictions when a specific trigger is present in the input. Robust aggregation schemes were proposed to counteract poisoning attacks, with the goal of either discarding the possibly corrupt local models from the aggregation or minimizing their impact using various scoring measures [9], [10], [20], [58], [66], [75].

Despite the pressing need to address both types of attacks, few works have attempted to tackle both simultaneously [32], [55], [66]. This is primarily due to the conflict between robust aggregation schemes, which require individual analysis of each update, and secure aggregation, which only reveals the aggregated joint model and does not allow for individual analysis. Furthermore, these existing methods are computationally demanding, incur a significant runtime overhead over their privacy-free variants, and assume weaker corruption models. Thus, we intend to answer the following question:

How practical is it to use robust aggregation schemes for privacy-preserving federated learning?

In this paper, we address the above question by offering an MPC-based framework for evaluating the effectiveness

2023 IEEE Security and Privacy Workshops (SPW) | 979-8-3503-1236-2/23/$31.00 ©2023 IEEE | DOI: 10.1109/SPW59333.2023.00012

DOI 10.1109/SPW59333.2023.00012

TABLE 1: High-level comparison of SAFEFL and previous works (arranged chronologically). Techniques: HE—Homomorphic Encryption [2], MPC—Secure Multi-Party Computation [51], TEE—Trusted Execution Environment [62], ZKP—Zero Knowledge Proofs [16], and DP—Differential Privacy [30].

Solution	Technique	Model Privacy	Poisoning Resilience	Malicious Servers	Distributed Servers	Efficient &Private	MPC Friendly	No Client Interaction	Open Source
FedAvg [59]	–	✗	✗	✗	✗	✗	✓	✓	✗
Krum [10]	–	✗	✓	✗	✗	✗	✗	✓	✓
SecAgg [15]	Masking	✓	✗	✓	✗	✗	✓	✗	✗
PPDL [71]	HE	✓	✗	✗	✗	✗	✓	✓	✗
Median [87]	–	✗	✓	✗	✗	✗	✗	✓	✗
SecAgg+ [8]	Masking	✓	✗	✓	✗	✓	✓	✗	✗
BatchCrypt [88]	HE	✓	✗	✓	✗	✗	✓	✓	✓
FoolsGold [37]	–	✗	✓	✗	✗	✗	✗	✓	✓
SAFELearn [35]	MPC	✓	✗	✗	✓	✓	✓	✓	✓
BREA [77]	Masking	✓	✓	✗	✗	✗	✗	✗	✗
FLOD [32]	HE+MPC	✓	✓	✗	✓	✗	✓	✓	✗
PEFL [53]†	HE	✓	✓	✗	✗	✗	✗	✓	✗
FLTrust [20]	–	✗	✓	✗	✗	✗	✓	✓	✓
DnC [75]	–	✗	✓	✗	✗	✗	✗	✓	✓
CONTRA [6]	–	✗	✓	✗	✗	✗	✗	✓	✗
PPFL [61]	TEE	✓	✗	✗	✗	✓	✓	✓	✓
PPFDL [83]	HE+MPC	✓	✗	✗	✓	✗	✓	✓	✗
ShieldFL [55]	HE	✓	✓	✗	✓	✗	✗	✓	✗
FLAME [66]	MPC	✓	✓	✗	✓	✗	✗	✓	✗
FLDP [78]	Masking+DP	✓	✗	✗	✗	✓	✓	✗	✗
SignGuard [84]	–	✗	✓	✗	✗	✗	✗	✓	✗
FLARE [81]	–	✗	✓	✗	✗	✗	✗	✓	✗
Romoa [57]	–	✗	✓	✗	✗	✗	✗	✓	✗
EIFFel [26]	MPC+ZKP	✓	✓	✓	✗	✗	✓	✗	✗
ELSA [73]‡	MPC	✓	✓	✗	✓	✓	✓	✓	✓
SAFEFL (**This**)	MPC	✓	✓	✓	✓	✓	✓	✓	✓

† PEFL [53] was completely broken in [74]. ‡ ELSA [73] supports only norm-based aggregation [79].

of robust aggregation techniques against poisoning attacks while also examining the overhead (communication and runtime) of introducing model privacy into it. We anticipate that our framework will aid in the development of more efficient schemes in the future that can effectively protect against both privacy inference attacks and poisoning attacks.

1.1. Our Contributions

We present SAFEFL, an MPC-based framework for assessing the effectiveness and performance of FL techniques that protect against both privacy inference and poisoning attacks. SAFEFL adopts a distributed aggregator setup in contrast to several existing frameworks that use a centralized aggregator. This design choice is motivated by the vulnerabilities demonstrated in recent studies on single aggregator setups in FL [12], [13], [36], [67], [82]. These attacks showed that when the central aggregator is maliciously corrupt, FL privacy can be compromised even if secure aggregation is in place.

The distributed aggregator in SAFEFL is realised using MPC techniques, in which users securely distribute their local updates across two or more servers, which privately compute the aggregation function using an interactive protocol [35], [66], [73]. These practical techniques, used in real-world deployments [1], [3], remove user interaction and efficiently handle user dropouts in the FL context. Concretely, we use the well-known MP-SPDZ [44] framework, which contains implementation of various MPC protocols (see §3.1 for more details).

However, combining MP-SPDZ with a secure aggregation scheme is challenging as MP-SPDZ is primarily designed for continuous secure computing, whereas secure aggregation in FL is interleaved with local user training [42]. To address this issue, we developed a communicator interface that connects the widely used open source ML framework PyTorch [72] and MP-SPDZ by extending the `ExternalIO` library [43] provided in MP-SPDZ. This capability allows us to evaluate the effectiveness and performance of a robust aggregation protocol in different MPC settings with ease, thus facilitating the development of more efficient protocols.

To identify the best candidate for robust aggregation in SAFEFL, we conducted a comprehensive evaluation of different robust aggregation methods against various poisoning attacks, including the state-of-the-art Min-Max attack [75]. Specifically, we tested these attacks on a Linear Regression classifier using the Human Activity Recognition (HAR) dataset [4]. Finally, we used our SAFEFL framework to evaluate the privacy-preserving variant of robust aggregation in FLTrust [20] over various MPC settings supported by MP-SPDZ, as it provided the best trade-off between accuracy

and computational costs among the approaches we studied. Our framework is open-sourced under the MIT License at https://encrypto.de/code/SAFEFL.

Our contributions are summarised as follows:

- We present SAFEFL, an MPC-based framework for evaluating the effectiveness and performance of FL techniques that protect against both privacy inference and poisoning attacks.
- SAFEFL provides a communicator interface between the PyTorch [72] ML framework and the MP-SPDZ [44] library, allowing for the simple translation of a robust aggregation scheme to its private equivalent across many MPC protocols.
- With SAFEFL, we implement a wide range of FL poisoning attacks, including Min-Max [75], and perform a comprehensive evaluation of various robust aggregation schemes and report accuracy.
- We evaluate the computation and communication overhead for the private implementation of the robust aggregation scheme in FLTrust [20] using multiple MPC protocols tailored to various settings.

Tab. 1 provides a high-level comparison of SAFEFL with previous works and the details regarding relevant related work are provided in §3.

2. SAFEFL Framework

This section provides the details of our SAFEFL framework. At a high level, SAFEFL comprises of two modules, Model Training and Aggregation, and a communicator interface that connects them, as illustrated in Fig. 1.

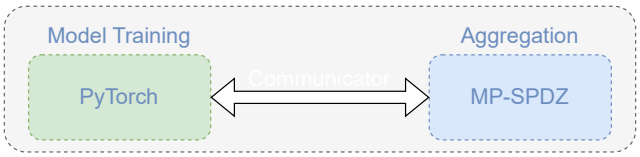

Figure 1: High level overview of SAFEFL framework, comprising of the Model Training module using PyTorch [72] and the Aggregation module using MP-SPDZ [44] along with the communicator interface between them.

Each of these components are detailed next. Along the way, we will also discuss various evaluations carried out using SAFEFL.

Benchmarking Environment. All experiments are run on a 16-core machine,[1] with a 2.8 GHz Intel Core i9-7960X processor and 128GB RAM, running Linux. We evaluated over a LAN setting with bandwidth 10Gbps and a round trip time (RTT) of 1ms.

The evaluations are carried out using a Linear Regression (LR) classifier over the Human Activity Recognition (HAR) [4] dataset. HAR is an unbalanced dataset with human activity data collected from the smartphones of 30 real-world users. We used 75% of each user's data as training examples and the remaining 25% for testing.

1. One machine is used per aggregator for MPC evaluations.

2.1. Model Training Module

This PyTorch-based module is responsible for performing the local model training on behalf of the users. For this, we use the publicly available code of FLTrust [20] as a starting point. The code includes a basic FL setup and their robust aggregation method implemented in Apache MXNet [64], as well as the implementation of the Trim poisoning attack [34]. We changed the code to use PyTorch [72] instead of MXNet and implemented (plaintext variants of) 7 poisoning attacks (cf. §3.2) and 14 different aggregation schemes (cf. §3.3).

Accuracy Evaluation. Our evaluation comprises of 30 users (consistent with the HAR dataset) and the model training was carried out for 2,000 iterations assuming a 20% malicious corruption. Our choice of attacks covers both data poisoning and model poisoning attacks in FL. As discussed in §3.3, FLTrust [20], FLOD [32], and FLARE [81] assume the presence of a 'root dataset'. For this, we sampled 100 data points uniformly at random. Table 2 summarises the various evaluation-specific parameters used.

TABLE 2: Parameters used in SAFEFL for evaluating the effectiveness of various poisoning attacks (cf. §3.2) against different aggregation schemes (cf. §3.3).

Parameter	HAR
# users (n)	30
# malicious users (f)	6 (20%)
# iterations	2,000
learning rate	0.25
batch size	64
size of server dataset	100
β (Trim-mean [87])	6
τ (FLOD [32])	50% of parameters
$\epsilon \vert \delta$ (FLAME [66])	3,000|0.001
$niter \vert b \vert c$ (DnC [75])	5|2,000|1.0
κ (ShieldFL [55])	0

Tab. 3 summarises the result of our accuracy evaluation. We observe that the following three robust aggregation schemes—DnC [75], FLAME [66], and FLTrust [20]—always achieved an accuracy of at most 0.05 less than what a simple FedAvg [59] could attain in the absence of an attack, i.e., 0.97.

2.2. Communicator Interface

After locally training the models in SAFEFL with the Model Training module, the next step is to perform private and robust aggregation in a distributed aggregator setup using MPC techniques. This aggregation could be performed using the Aggregation module running on the MP-SPDZ framework, as described in §2.3. However, since the protocols used in MP-SPDZ are designed for continuous secure computation, they must be compiled and executed on a virtual machine. The training in FL, on the other hand,

TABLE 3: Accuracy evaluation (in plaintext) of various FL aggregations (cf. §3.3) under different attacks (cf. §3.2) using a Linear Regression classifier over the HAR data set [4] (larger is better, best values marked in **bold**).

Aggr. \ Attack	No	LF [34], [76]	Krum [34]	Trim [34]	Min-Max [75]	Min-Sum [75]	Scaling [34]	FLTrust [20]
FedAvg [59]	**0.97**	0.74	0.95	0.69	0.86	0.95	0.95	**0.96**
Krum [10]	0.88	0.91	0.65	0.91	0.91	0.87	0.91	0.90
Trim-mean [87]	0.95	0.93	0.95	0.71	0.91	0.95	0.95	0.91
Median [87]	0.95	0.93	0.95	0.85	0.92	0.94	0.95	0.93
FLTrust [20]	0.96	0.96	0.94	0.94	0.92	0.94	0.94	0.95
DnC [75]	0.97	**0.96**	**0.96**	**0.96**	**0.95**	0.96	0.96	0.95
FoolsGold [37]	0.79	0.31	0.77	0.82	0.84	0.81	0.84	0.51
CONTRA [6]	0.94	0.02	0.17	0.51	0.71	0.91	0.91	0.34
FLARE [81]	0.94	0.63	0.92	0.78	0.77	0.94	0.92	0.91
Romoa [57]	0.96	0.78	0.93	0.89	0.91	**0.97**	0.92	0.92
SignGuard [84]	0.96	0.88	0.93	0.91	0.92	0.92	**0.96**	0.94
FLAME [66]	0.95	0.95	0.96	0.95	0.94	0.96	0.95	0.95
FLOD [32]	0.92	0.89	0.92	0.90	0.87	0.92	0.92	0.92
ShieldFL [55]	0.94	0.92	0.33	0.91	0.78	0.84	0.94	0.84

necessitates interleaved invocations of secure aggregation in between local training.

To solve the issue, we created a communicator interface in SAFEFL that enables bidirectional communication between PyTorch and the MP-SPDZ library. We utilize this communicator to securely transfer secret shares of locally trained models from PyTorch to the MPC servers in MP-SPDZ, who compute the aggregation using specified MPC protocols and return the aggregated model to PyTorch for the next training iteration.

Our starting point is the Banker Bonus example provided by MP-SPDZ [43], which solves the Yao's Millionaires' problem [85] with up to 8 users. In this example, the MPC servers listen on a specified port for the users and accept connections from the user-side interface. When all users are connected, the computation begins and the connection is closed upon completion of the computation. Furthermore, the connection is secured with SSL, and the required keys and certificates are generated upon launch.

In SAFEFL, we extended the MP-SPDZ user interface to send an arbitrary amount of data and integrated the user into PyTorch to send the secret-shared local models to the MPC servers and retrieve the aggregated model. For simplicity, we let PyTorch behave as a single user, distributing all local models with the MPC servers. However, this can simply be extended such that each user connects separately. As far as we know, this is the first time a communicator has been developed to connect PyTorch and MP-SPDZ, specifically for federated learning.

2.3. Aggregation Module

This module is responsible for executing the distributed secure aggregation utilizing MPC protocols implemented in MP-SPDZ. We chose FLTrust [20] as the best candidate to adopt as a private and robust aggregation scheme in SAFEFL. This is because it is the most MPC-friendly of the three schemes—DnC, FLAME and FLTrust, which we identified as the best robust aggregation schemes in §2.1. For FLTrust, we allowed a trusted user in PyTorch, which could be a user or an MPC server, to train the server model over the root dataset. MP-SPDZ is used to compute trust scores and aggregate the final model. We also implemented the FedAvg [59] aggregation scheme to serve as a baseline to estimate the cost overhead of adding robust aggregation. We removed the weighting by the data size of a user from FedAvg to make it more efficient and reduce numerical errors.

Note that once an aggregation technique has been implemented in MP-SPDZ, it is simple to evaluate it using any of the MPC protocols available in MP-SPDZ, hence improving usability. The code in Listing 1 shows the simplicity of implementing the FLTrust aggregation in SAFEFL.

```
# Aggregation
@for_range_multithread(N_THREADS, N_PARALLEL,
    WORKERS - 1)
def _(i):
    input[i][:] = trust_score[i] / norm[i]
                  * input[i][:]

global_model_update = sfix.Array(PARAM_NUM)
@for_range_opt(WORKERS - 1)
def _(i):
    global_model_update[:] += input[i][:]

global_model_update = norm[WORKERS - 1] /
        total_trust_score * global_model_update[:]
```

Listing 1: SAFEFL code snippet for FLTrust [20]

Tab. 4 provides the communication and runtime costs for evaluating private variants of FLTrust and FedAvg using our aggregation module. The experiments are run over a 64-bit ring using four different MPC protocols using MP-SPDZ [44], catering to different settings.

The trends observed in Tab. 4 correspond to the complexity of the underlying MPC protocols, with honest majority outperforming dishonest majority and malicious security being more expensive than semi-honest security. When moving from FedAvg to FLTrust in the stronger dishonest majority setting with malicious security, we observe $\approx 16\times$ higher communication and $\approx 3.6\times$ higher runtime. While a similar trend is observed for communication in the malicious

TABLE 4: Communication (Comm. in GB) and computation (Time in hours) costs for privacy-preserving implementation of FLTrust [20] and FedAvg [59] for 50 iterations in various MPC settings using MP-SPDZ [44]. Notations: N - number of MPC servers, DM - dishonest majority.

MPC Protocol	N	FLTrust [20]		FedAvg [59]	
		Comm.	Time	Comm.	Time
Semi2k	2	1,279.50	0.65	81.93	0.38
(DM, semi-honest)	3	7,454.37	2.28	146.47	0.55
SPDZ2k [27]	2	19,086.60	8.27	1,216.15	3.56
(DM, malicious)	3	57,191.00	17.64	3,643.82	4.87
Replicated2k [5]	3	16.25	0.27	0.98	0.12
PsReplicated2k [33]	3	62.52	0.49	3.87	0.33

honest majority setting, the overhead in runtime is reduced to $\approx 1.5\times$. The observations suggest that additional efforts should be made to bridge the communication gap in order to make private and robust aggregation schemes more feasible.

3. Related Work

This section provides a succinct overview of works related to our SAFEFL framework.

3.1. Secure Multi-party Computation (MPC)

MPC [38], [86] enables a set of distrusting parties to compute on their combined input without revealing more information than they could infer from their input and output. MPC protocols can be categorised into several types based on the nature of the corruption; two of these categories are discussed here.

Honest vs. Dishonest Majority. This classification is based on the amount of possible corruption among the MPC parties. In an honest majority setting [5], [18], [21], [22], [48], the majority of the parties are considered to be honest and follow the protocol. Dishonest majority protocols [29], [63], [68], on the other hand, tolerate corruption of all but one party.

Semi-honest vs. Malicious Security. This classification concerns the nature of the corruption. In a semi-honest setting [41], [68], [69], corrupt parties follow the protocol, but are curious and tend to learn more information than intended. The malicious setting [27], [33], [47], [49] models scenarios, where corrupt parties can arbitrarily deviate from the protocol.

3.2. Poisoning Attacks in FL

The poisoning attacks [76] evaluated in SAFEFL can be broadly classified into four categories and the details are provided next.

1) *Label flipping (LF) [34], [76]:* In this attack, corrupt users poison their training data by flipping the labels of some instances from one class (the source class) to another (i.e., the target class). We use the untargeted attack in [34], where the new label is defined as $l_{\text{new}} = L - l_{\text{old}} - 1$, for L classes.

2) *Scaling [20]:* This is a backdoor technique that alters data samples by adding a trigger and modifying the label to a desired target class. To amplify the attack's impact, the compromised models are usually scaled up. In SAFEFL, each corrupt user duplicates a random fraction, $p \in (0, 1]$, of their training data for alteration and scales up by the total number of users.

3) *AGR-tailored [20], [34]:* Fang et al. [34] proposed a framework for optimizing local model poisoning attacks for any aggregation rule. The framework formulates the attack as a maximization problem to deviate the global model from its expected direction of change. We used the model poisoning attack framework of [34] to optimize the attacks Krum, Trim, and FLTrust [20, §V].

4) *AGR-agnostic [75]:* Here, the attacker lacks knowledge of the aggregation algorithm and its constraints. The two proposed attacks, Min-Max and Min-Sum, were shown to outperform the previously published LIE attack [7].

3.3. Robust Aggregation in FL

In addition to the simple aggregation scheme FedAvg [59], which simply computes a (weighted) average of all inputs, we implement and evaluate 13 other robust aggregation schemes [24] in SAFEFL:

1) *Krum [10]:* In each iteration, Krum selects a global model update from n local updates using a Euclidean distance score. For f malicious users, the score is determined by computing the distance between each pair of models and selecting the model with the lowest sum of distances to the closest $n - f - 2$ models. The user with the minimal score has their local update chosen as the global update.

2) *Trim-mean [87]:* This method aggregates model parameters coordinate-wise by sorting their values in local model updates, removing the largest and smallest β values for a given parameter β, and computing the mean of the remaining values as the final parameter value in the global update.

3) *Median [87]:* This method, similar to Trim-mean, sorts values in each local model update. However, instead of computing the mean after trimming, the median value of each parameter is considered as the global update value.

4) *FLTrust [20]:* FLTrust utilizes a root dataset on the server and assesses the trust score of a local model update based on the deviation from the server's model update. This is achieved through cosine similarity measurement and ReLU clipping.

5) *Divide-and-Conquer (DnC) [75]:* DnC selects a random set of gradient coordinates r, of size less than b, and constructs a subsampled set ∇_r. The mean of ∇_r is then calculated to obtain the centered set ∇_c. The algorithm computes projections along the top right singular eigenvector v and calculates a vector of outlier scores s. A set of $c \cdot f$

gradients with the highest scores are removed, with the remaining gradients being considered "good" and added to a set. This is repeated for $niter$ iterations, with the set of indices being randomized each time, and the good gradients are aggregated by computing the average of the common gradients in all $niter$ good sets.

6–7) *FoolsGold [37], CONTRA [6]:* FoolsGold tracks user updates by aggregating them over multiple iterations. It computes the cosine similarity between aggregated updates and adjusts the learning rate α_i per user based on update similarity and historical information. Similarly, CONTRA also limits similar updates by either reducing their learning rates or discarding them. However, these methods result in significant accuracy drop when good updates are similar (cf. accuracy against LF attack in Tab. 3).

8) *FLARE [81]:* This method found that the penultimate layer (PLR) has a unique ability to differentiate malicious models from benign ones. The PLRs of benign models have a similar distribution, while those of malicious models have a different distribution. FLARE showed that the distances between benign PLRs are smaller than those between benign and malicious PLRs. The method assigns a root score (similar to FLTrust [20]) to each user based on Maximum Mean Discrepancies between PLRs. The model updates are then scaled and averaged, weighted by the root score of each user.

9) *Romoa [57]:* Romoa considers three similarity measures: element-wise cosine similarity, layer-wise cosine similarity, and layer-wise Pearson correlation. Users share their local models with the aggregation server after each iteration, but the aggregation is only performed every t iterations. The calculation of similarity measures is performed every iteration to compute sanitization factors that are used during aggregation to find the aggregate as a weighted sum.

10) *SignGuard [84]:* SignGuard aggregates models through sign-based clustering and norm-based thresholding. The median of local model norms is calculated to determine the norm bound. Local models with normalized norms within the bound (0.1 to 3.0) are added to a set. A 10% random subset of coordinates is selected from local models for sign-based clustering. The cluster with the highest number of elements is considered benign, and the final global model is the average of these benign local models.

11) *FLAME [66]:* This method aggregates local models through clustering, clipping, and adaptive noise addition. Clustering is performed using HDBSCAN [19] with a cosine distance metric and a minimum cluster size of $n/2 + 1$. Outlier models are excluded, while remaining models are clipped, averaged, and modified with adaptive noise based on clipping bounds and privacy parameters ϵ and δ.

12) *FLOD [32]:* This method, like FLTrust, uses a trusted server model to determine if a model should be discarded and to calculate the weighted average of the remaining models. Model updates are converted to Boolean via the sign function and Hamming distance is used for the weighted average of the updates. Models with Hamming distance greater than or equal to the threshold τ are excluded from the average.

13) *ShieldFL [55]:* In ShieldFL, users normalize updates larger than κ or with change exceeding the threshold and set other updates to 0. The server aggregates models by a normalization check, cosine similarity calculation with respect to the last iteration, poison baseline identification, cosine distance calculation as weight, and weighted average adjustment.

4. Conclusion & Future Work

This paper presents SAFEFL, a framework that leverages secure multi-party computation (MPC) to evaluate the effectiveness and performance of federated learning (FL) techniques in protecting against privacy inference and poisoning attacks. The framework features a communicator interface that integrates PyTorch-based implementations with the well-established MP-SPDZ framework [44], providing a solid foundation for creating more efficient FL systems that can effectively protect against privacy breaches and malicious attacks. We carried out a comprehensive evaluation to determine the impact of different poisoning attacks on various robust aggregation methods. We also assessed the computational and communication costs of incorporating MPC for privacy protection in FLTrust, a well-known robust aggregation technique. With the continued development and use of SAFEFL, we believe it will greatly contribute to the advancement of private and robust federated learning systems.

As future work, we plan to expand the compatibility of our framework to include other MPC frameworks such as MOTION [17] and Silph [23]. Additionally, we aim to enhance the accuracy evaluation of our framework by testing it with more complex architectures, such as Deep Neural Networks.

Acknowledgements

This project received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No. 850990 PSOTI). It was co-funded by the Deutsche Forschungsgemeinschaft (DFG) within SFB 1119 CROSSING/236615297 and GRK 2050 Privacy & Trust/251805230.

References

[1] J. Aas and T. Geoghegan. (2022) Introducing ISRG Prio Services for Privacy Respecting Metrics. https://www.abetterinternet.org/post/introducing-prioservices/.

[2] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, "A Survey on Homomorphic Encryption Schemes: Theory and Implementation," *ACM Computing Surveys*, 2018.

[3] S. Addanki, K. Garbe, E. Jaffe, R. Ostrovsky, and A. Polychroniadou, "Prio+: Privacy Preserving Aggregate Statistics via Boolean Shares," in *SCN*, 2022.

[4] D. Anguita, A. Ghio, L. Oneto, X. Parra Perez, and J. L. Reyes Ortiz, "A Public Domain Dataset for Human Activity Recognition Using Smartphones," in *ESANN*, 2013.

[5] T. Araki, J. Furukawa, Y. Lindell, A. Nof, and K. Ohara, "High-Throughput Semi-Honest Secure Three-Party Computation with an Honest Majority," in *CCS*, 2016.

[6] S. Awan, B. Luo, and F. Li, "Contra: Defending Against Poisoning Attacks in Federated Learning," in *ESORICS*, 2021.

[7] G. Baruch, M. Baruch, and Y. Goldberg, "A Little Is Enough: Circumventing Defenses For Distributed Learning," in *NeurIPS*, 2019.

[8] J. H. Bell, K. A. Bonawitz, A. Gascón, T. Lepoint, and M. Raykova, "Secure Single-Server Aggregation with (Poly)Logarithmic Overhead," in *CCS*, 2020.

[9] Y. Ben-Itzhak, H. Möllering, B. Pinkas, T. Schneider, A. Suresh, O. Tkachenko, S. Vargaftik, C. Weinert, H. Yalame, and A. Yanai, "ScionFL: Secure Quantized Aggregation for Federated Learning," *CoRR*, vol. abs/2210.07376, 2022.

[10] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent," in *NeurIPS*, 2017.

[11] F. Boemer, R. Cammarota, D. Demmler, T. Schneider, and H. Yalame, "MP2ML: A Mixed-Protocol Machine Learning Framework for Private Inference," in *ARES*, 2020.

[12] F. Boenisch, A. Dziedzic, R. Schuster, A. S. Shamsabadi, I. Shumailov, and N. Papernot, "When the Curious Abandon Honesty: Federated Learning Is Not Private," *CoRR*, vol. abs/2112.02918, 2021.

[13] ——, "Is Federated Learning a Practical PET Yet?" *CoRR*, vol. abs/2301.04017, 2023.

[14] K. A. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečný, S. Mazzocchi, B. McMahan, T. V. Overveldt, D. Petrou, D. Ramage, and J. Roselander, "Towards Federated Learning at Scale: System Design," in *MLSys*, 2019.

[15] K. A. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical Secure Aggregation for Privacy-Preserving Machine Learning," in *CCS*, 2017.

[16] D. Boneh, E. Boyle, H. Corrigan-Gibbs, N. Gilboa, and Y. Ishai, "Zero-Knowledge Proofs on Secret-Shared Data via Fully Linear PCPs," in *CRYPTO*, 2019.

[17] L. Braun, D. Demmler, T. Schneider, and O. Tkachenko, "MOTION–A Framework for Mixed-Protocol Multi-Party Computation," *TOPS*.

[18] M. Byali, H. Chaudhari, A. Patra, and A. Suresh, "FLASH: Fast and Robust Framework for Privacy-preserving Machine Learning," *PETS*, 2020.

[19] R. J. Campello, D. Moulavi, and J. Sander, "Density-based Clustering based on Hierarchical Density Estimates," in *PAKDD*, 2013.

[20] X. Cao, M. Fang, J. Liu, and N. Z. Gong, "FLTrust: Byzantine-robust Federated Learning via Trust Bootstrapping," in *NDSS*, 2021, https://people.duke.edu/~zg70/code/fltrust.zip.

[21] H. Chaudhari, A. Choudhury, A. Patra, and A. Suresh, "ASTRA: High Throughput 3PC over Rings with Application to Secure Prediction," in *CCSW@CCS*, 2019.

[22] H. Chaudhari, R. Rachuri, and A. Suresh, "Trident: Efficient 4PC Framework for Privacy Preserving Machine Learning," in *NDSS*, 2020.

[23] E. Chen, J. Zhu, A. Ozdemir, R. S. Wahby, F. Brown, and W. Zheng, "Silph: A Framework for Scalable and Accurate Generation of Hybrid MPC Protocols," in *IEEE S&P*, 2023.

[24] H. Chen and F. Koushanfar, "Tutorial: Towards Robust Deep Learning against Poisoning Attacks," *ACM Transactions on Embedded Computing Systems*, 2022.

[25] L. Chen, Y. Li, C. Huang, B. Li, Y. Xing, D. Tian, L. Li, Z. Hu, X. Na, Z. Li *et al.*, "Milestones in Autonomous Driving and Intelligent Vehicles: Survey of Surveys," *IEEE Transactions on Intelligent Vehicles*, 2022.

[26] A. R. Chowdhury, C. Guo, S. Jha, and L. van der Maaten, "EIFFeL: Ensuring Integrity for Federated Learning," in *CCS*, 2022.

[27] R. Cramer, I. Damgård, D. Escudero, P. Scholl, and C. Xing, "SPDZ2k: Efficient MPC mod 2^k for Dishonest Majority," in *CRYPTO*, 2018.

[28] A. P. K. Dalskov, D. Escudero, and M. Keller, "Secure Evaluation of Quantized Neural Networks," *PETS*, 2020.

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

[30] D. Desfontaines and B. Pejó, "SoK: Differential Privacies," *PETS*, 2020.

[31] M. F. Dixon, I. Halperin, and P. Bilokon, *Machine Learning in Finance: From Theory to Practice*. Springer, 2020.

[32] Y. Dong, X. Chen, K. Li, D. Wang, and S. Zeng, "FLOD: Oblivious Defender for Private Byzantine-Robust Federated Learning with Dishonest-Majority," in *ESORICS*, 2021.

[33] H. Eerikson, M. Keller, C. Orlandi, P. Pullonen, J. Puura, and M. Simkin, "Use your Brain! Arithmetic 3PC For Any Modulus with Active Security," in *ITC*, 2019.

[34] M. Fang, X. Cao, J. Jia, and N. Gong, "Local Model Poisoning Attacks to Byzantine-Robust Federated Learning," in *USENIX Security*, 2020.

[35] H. Fereidooni, S. Marchal, M. Miettinen, A. Mirhoseini, H. Möllering, T. D. Nguyen, P. Rieger, A.-R. Sadeghi, T. Schneider, H. Yalame, and S. Zeitouni, "SAFELearn: Secure Aggregation for Private FEderated Learning," in *DLS@S&P*, 2021.

[36] L. H. Fowl, J. Geiping, W. Czaja, M. Goldblum, and T. Goldstein, "Robbing the Fed: Directly Obtaining Private Data in Federated Learning with Modified Models," in *ICLR*, 2022.

[37] C. Fung, C. J. Yoon, and I. Beschastnikh, "The Limitations of Federated Learning in Sybil Settings," in *RAID*, 2020.

[38] 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.

[39] C. He, S. Li, J. So, M. Zhang, H. Wang, X. Wang, P. Vepakomma, A. Singh, H. Qiu, L. Shen, P. Zhao, Y. Kang, Y. Liu, R. Raskar, Q. Yang, M. Annavaram, and S. Avestimehr, "FedML: A Research Library and Benchmark for Federated Machine Learning," *CoRR*, vol. abs/2007.13518, 2020.

[40] A. Hegde, H. Möllering, T. Schneider, and H. Yalame, "SoK: Efficient Privacy-preserving Clustering," *PETS*, vol. 2021.

[41] T. Heldmann, T. Schneider, O. Tkachenko, C. Weinert, and H. Yalame, "LLVM-based Circuit Compilation for Practical Secure Computation," in *ACNS*, 2021.

[42] M. Keller. https://github.com/data61/MP-SPDZ/issues/614.

[43] ——. https://github.com/data61/MP-SPDZ/tree/master/ExternalIO.

[44] ——, "MP-SPDZ: A Versatile Framework for Multi-Party Computation," in *CCS*, 2020.

[45] M. Keller and K. Sun, "Secure Quantized Training for Deep Learning," in *ICML*, 2022.

[46] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *CoRR*, vol. abs/1610.05492, 2016.

[47] N. Koti, M. Pancholi, A. Patra, and A. Suresh, "SWIFT: Superfast and Robust Privacy-Preserving Machine Learning," in *USENIX Security*, 2021.

[48] N. Koti, S. Patil, A. Patra, and A. Suresh, "MPClan: Protocol Suite for Privacy-Conscious Computations," *CoRR*, vol. abs/2206.12224, 2022.

[49] N. Koti, A. Patra, R. Rachuri, and A. Suresh, "Tetrad: Actively Secure 4PC for Secure Training and Inference," in *NDSS*, 2022.

[50] R. Lamsal, A. Harwood, and M. R. Read, "Socially Enhanced Situation Awareness from Microblogs Using Artificial Intelligence: A Survey," *ACM Computing Surveys*, 2022.

[51] Y. Lindell, "Secure Multiparty Computation," *Communications of the ACM*, 2020.

[52] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig, "Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing," *ACM Computing Surveys*, 2023.

[53] X. Liu, H. Li, G. Xu, Z. Chen, X. Huang, and R. Lu, "Privacy-Enhanced Federated Learning Against Poisoning Adversaries," *IEEE TIFS*, 2021.

[54] H. Ludwig, N. Baracaldo, G. Thomas, Y. Zhou, A. Anwar, S. Rajamoni, Y. Ong, J. Radhakrishnan, A. Verma, and M. Sinn, "IBM Federated Learning: an Enterprise Framework White Paper V0.1," *CoRR*, vol. abs/2007.10987, 2020.

[55] Z. Ma, J. Ma, Y. Miao, Y. Li, and R. H. Deng, "ShieldFL: Mitigating Model Poisoning Attacks in Privacy-Preserving Federated Learning," *IEEE TIFS*, 2022.

[56] M. Mansouri, M. Önen, W. B. Jaballah, and M. Conti, "SoK: Secure Aggregation Based on Cryptographic Schemes for Federated Learning," *PETS*, 2023.

[57] Y. Mao, X. Yuan, X. Zhao, and S. Zhong, "Romoa: Robust Model Aggregation for the Resistance of Federated Learning to Model Poisoning Attacks," in *ESORICS*, 2021.

[58] F. Marx, T. Schneider, A. Suresh, T. Wehrle, C. Weinert, and H. Yalame, "HyFL: A Hybrid Approach For Private Federated Learning," *CoRR*, vol. abs/2302.09904, 2023.

[59] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *AISTATS*, 2017.

[60] L. Melis, C. Song, E. D. Cristofaro, and V. Shmatikov, "Exploiting Unintended Feature Leakage in Collaborative Learning," in *IEEE S&P*, 2019.

[61] F. Mo, H. Haddadi, K. Katevas, E. Marin, D. Perino, and N. Kourtellis, "PPFL: Privacy-preserving Federated Learning with Trusted Execution Environments," in *ACM MobiSys*, 2021.

[62] F. Mo, Z. Tarkhani, and H. Haddadi, "SoK: Machine Learning with Confidential Computing," *CoRR*, vol. abs/2208.10134, 2022.

[63] J.-P. Münch, T. Schneider, and H. Yalame, "VASA: Vector AES Instructions for Security Applications," in *ACSAC*, 2021.

[64] MXNet. https://mxnet.apache.org/.

[65] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive Privacy Analysis of Deep Learning: Passive and Active White-box Inference Attacks against Centralized and Federated Learning," in *IEEE S&P*, 2019.

[66] T. D. Nguyen, P. Rieger, H. Chen, H. Yalame, H. Möllering, H. Fereidooni, S. Marchal, M. Miettinen, A. Mirhoseini, S. Zeitouni, F. Koushanfar, A.-R. Sadeghi, and T. Schneider, "FLAME: Taming Backdoors in Federated Learning," in *USENIX Security*, 2022.

[67] D. Pasquini, D. Francati, and G. Ateniese, "Eluding Secure Aggregation in Federated Learning via Model Inconsistency," in *CCS*, 2022.

[68] A. Patra, T. Schneider, A. Suresh, and H. Yalame, "ABY2.0: Improved Mixed-Protocol Secure Two-Party Computation," in *USENIX Security*, 2021.

[69] ——, "SynCirc: Efficient Synthesis of Depth-Optimized Circuits for Secure Computation," in *IEEE HOST*, 2021.

[70] A. Patra and A. Suresh, "BLAZE: Blazing Fast Privacy-Preserving Machine Learning," in *NDSS*, 2020.

[71] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai, "Privacy-Preserving Deep Learning via Additively Homomorphic Encryption," *IEEE TIFS*, 2018.

[72] PyTorch. https://pytorch.org/.

[73] M. Rathee, C. Shen, S. Wagh, and R. A. Popa, "ELSA: Secure Aggregation for Federated Learning with Malicious Actors," *IEEE S&P*, 2023.

[74] T. Schneider, A. Suresh, and H. Yalame, "Comments on "Privacy-Enhanced Federated Learning Against Poisoning Adversaries"," *IEEE TIFS*, 2023.

[75] V. Shejwalkar and A. Houmansadr, "Manipulating the Byzantine: Optimizing Model Poisoning Attacks and Defenses for Federated Learning," in *NDSS*, 2021.

[76] V. Shejwalkar, A. Houmansadr, P. Kairouz, and D. Ramage, "Back to the Drawing Board: A Critical Evaluation of Poisoning Attacks on Production Federated Learning," in *IEEE S&P*, 2022.

[77] J. So, B. Güler, and A. S. Avestimehr, "Byzantine-Resilient Secure Federated Learning," *IEEE JSAC*, 2021.

[78] T. Stevens, C. Skalka, C. Vincent, J. Ring, S. Clark, and J. P. Near, "Efficient Differentially Private Secure Aggregation for Federated Learning via Hardness of Learning with Errors," in *USENIX Security*, 2022.

[79] Z. Sun, P. Kairouz, A. T. Suresh, and H. B. McMahan, "Can You Really Backdoor Federated Learning?" *CoRR*, vol. abs/1911.07963, 2019.

[80] N. F. L. Team. (2021) Federated Learning for Healthcare Using NVIDIA Clara (White Paper). https://developer.nvidia.com/blog/federated-learning-clara/.

[81] N. Wang, Y. Xiao, Y. Chen, Y. Hu, W. Lou, and Y. T. Hou, "FLARE: Defending Federated Learning against Model Poisoning Attacks via Latent Space Representations," in *ASIACCS*, 2022.

[82] Y. Wen, J. Geiping, L. Fowl, M. Goldblum, and T. Goldstein, "Fishing for User Data in Large-Batch Federated Learning via Gradient Magnification," in *ICML*, 2022.

[83] G. Xu, H. Li, Y. Zhang, S. Xu, J. Ning, and R. H. Deng, "Privacy-Preserving Federated Deep Learning With Irregular Users," *IEEE TDSC*, 2022.

[84] J. Xu, S.-L. Huang, L. Song, and T. Lan, "SignGuard: Byzantine-robust Federated Learning through Collaborative Malicious Gradient Filtering," in *ICDCS*, 2022.

[85] A. C. Yao, "Protocols for secure computations," in *FOCS*, 1982.

[86] A. C.-C. Yao, "How to Generate and Exchange Secrets (Extended Abstract)," in *FOCS*, 1986.

[87] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-Robust Distributed Learning: Towards Optimal Statistical Rates," in *ICML*, 2018.

[88] C. Zhang, S. Li, J. Xia, W. Wang, F. Yan, and Y. Liu, "BatchCrypt: Efficient Homomorphic Encryption for Cross-Silo Federated Learning," in *USENIX ATC*, 2020.