SAFEFL: MPC-friendly Framework for Private and Robust Federated Learning (Full Version*)

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 multi-party 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 [26], medical diagnosis [53], natural language processing [55], and finance [32]. 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], [29], [42], [48], [71], [73]. Existing PPML solutions use secure computation techniques such as secure multi-party computation (MPC) [54] 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) [49], 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 [62]). 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 [41] and industrial research [14], leading to the deployment of several real-world solutions [57], [83] 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/gradienteqs [63], [68]. 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 [8], [15], [36], [59].

In poisoning attacks, corrupt users create fraudulent models and send them to the aggregator to manipulate the training process [35], [78], [79], [82]. 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], [21], [61], [69], [78].

Despite the pressing need to address both types of attacks, few works have attempted to tackle both simultaneously [33], [58], [69]. 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:

*Please cite the version of this paper published at IEEE Deep Learning Security and Privacy Workshop (DLSP’23) [39].
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 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], [37], [70], [85]. 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 [36], [69], [76]. 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 [47] 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 [45]. To address this issue, we developed a communicator interface that connects the widely used open source ML framework PyTorch [75] and MP-SPDZ by extending the ExternalIO library [46] 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 [78]. 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 [21] 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 [75] ML framework and the MP-SPDZ [47] 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 [78], 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 [21] 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 [75] and the Aggregation module using MP-SPDZ [47] 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, 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.

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

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.

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 [21] as a starting point. The code includes a basic FL setup and their robust aggregation method implemented in Apache MXNet [67], as well as the implementation of the Trim poisoning attack [35]. We changed the code to use PyTorch [75] 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 [21], FLOD [33], and FLARE [84] 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>
<thead>
<tr>
<th>Parameter</th>
<th>HAR</th>
</tr>
</thead>
<tbody>
<tr>
<td># users (n)</td>
<td>30</td>
</tr>
</tbody>
</table>
| # malicious users (f) | 6 (20%)
| # iterations | 2,000 |
| learning rate | 0.25 |
| batch size | 64 |
| size of server dataset | 100 |
| β (Trim-mean [90]) | 6 |
| τ (FLOD [33]) | 50% of parameters |
| c/d (FLAME [69]) | 5,000/0.001 |
| niter|b|c (DnC [78]) | 5/2,000/1.0 |
| κ (ShieldFL [58]) | 0 |

Tab. 3 summarises the result of our accuracy evaluation. We observe that the following three robust aggregation schemes—DnC [78], FLAME [69], and FLTrust [21]—always achieved an accuracy of at most 0.05 less than what a simple FedAvg [62] 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, 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, which 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 [46], which solves the Yao’s Millionaires’ problem [88] 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 extended such that each user connects separately. As far as we know, this is the first time a communicator has been extended such that each user connects separately. As far as we know, this is the first time a communicator has been extended such that each user connects separately. As far as we know, this is the first time a communicator has been extended such that each user connects separately. As far as we know, this is the first time a communicator has been extended such that each user connects separately. As far as we know, this is the first time a communicator has been extended such that each user connects separately. As far as we know, this is the first time a communicator has been extended such that each user connects separately. As far as we know, this is the first time a communicator has been extended such that each user connects separately. As far as we know, this is the first time a communicator has been extended such that each user connects separately. As far as we know, this is the first time a communicator has been extended such that each user connects separately. As far as we know, this is the first time a communicator has been extended such that each user connects separately. As far as we know, this is the first time a communicator has been extended such that each user connects separately.

### 2.3. Aggregation Module

This module executes the distributed secure aggregation utilizing MPC protocols implemented in MP-SPDZ. We chose FLTrust [21] 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 [62] 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 _(_):
    input[i][i] = trust_score[i] / norm[i]
    input[i][i] += input[i][i]
    global_model_update = sfix.Array(PARAM_NUM)
    @for_range_opt(WORKERS - 1)
def _(_):
    global_model_update[i] += input[i][i]
    global_model_update = norm[WORKERS - 1] / total_trust_score * global_model_update
```

Listing 1: SAFEFL code snippet for FLTrust [21]

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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>FedAvg [62]</td>
<td>0.97</td>
<td>0.74</td>
<td>0.95</td>
<td>0.69</td>
<td>0.86</td>
<td>0.95</td>
<td>0.95</td>
<td>0.96</td>
<td></td>
</tr>
<tr>
<td>Krum [10]</td>
<td>0.88</td>
<td>0.91</td>
<td>0.65</td>
<td>0.91</td>
<td>0.91</td>
<td>0.87</td>
<td>0.91</td>
<td>0.90</td>
<td></td>
</tr>
<tr>
<td>Trim-mean [90]</td>
<td>0.95</td>
<td>0.93</td>
<td>0.95</td>
<td>0.71</td>
<td>0.91</td>
<td>0.95</td>
<td>0.95</td>
<td>0.91</td>
<td></td>
</tr>
<tr>
<td>Median [90]</td>
<td>0.95</td>
<td>0.93</td>
<td>0.95</td>
<td>0.85</td>
<td>0.92</td>
<td>0.94</td>
<td>0.95</td>
<td>0.93</td>
<td></td>
</tr>
<tr>
<td>FLTrust [21]</td>
<td>0.96</td>
<td>0.96</td>
<td>0.94</td>
<td>0.94</td>
<td>0.92</td>
<td>0.94</td>
<td>0.94</td>
<td>0.95</td>
<td></td>
</tr>
<tr>
<td>DnC [78]</td>
<td>0.97</td>
<td><strong>0.96</strong></td>
<td><strong>0.96</strong></td>
<td><strong>0.96</strong></td>
<td><strong>0.95</strong></td>
<td><strong>0.96</strong></td>
<td><strong>0.96</strong></td>
<td><strong>0.95</strong></td>
<td></td>
</tr>
<tr>
<td>FoolGold [38]</td>
<td>0.79</td>
<td>0.31</td>
<td>0.77</td>
<td>0.82</td>
<td>0.84</td>
<td>0.81</td>
<td>0.84</td>
<td>0.51</td>
<td></td>
</tr>
<tr>
<td>CONTRA [6]</td>
<td>0.94</td>
<td>0.02</td>
<td>0.17</td>
<td>0.51</td>
<td>0.71</td>
<td>0.91</td>
<td>0.91</td>
<td>0.34</td>
<td></td>
</tr>
<tr>
<td>FLARE [84]</td>
<td>0.94</td>
<td>0.63</td>
<td>0.92</td>
<td>0.78</td>
<td>0.77</td>
<td>0.94</td>
<td>0.92</td>
<td>0.91</td>
<td></td>
</tr>
<tr>
<td>Romoa [60]</td>
<td>0.96</td>
<td>0.78</td>
<td>0.93</td>
<td>0.89</td>
<td>0.91</td>
<td>0.97</td>
<td>0.92</td>
<td>0.92</td>
<td></td>
</tr>
<tr>
<td>SignGuard [87]</td>
<td>0.96</td>
<td>0.88</td>
<td>0.93</td>
<td>0.91</td>
<td>0.92</td>
<td>0.92</td>
<td>0.96</td>
<td>0.94</td>
<td></td>
</tr>
<tr>
<td>FLAME [69]</td>
<td>0.95</td>
<td>0.95</td>
<td>0.96</td>
<td>0.95</td>
<td>0.94</td>
<td>0.96</td>
<td>0.95</td>
<td>0.95</td>
<td></td>
</tr>
<tr>
<td>FLOD [33]</td>
<td>0.92</td>
<td>0.89</td>
<td>0.92</td>
<td>0.90</td>
<td>0.87</td>
<td>0.92</td>
<td>0.92</td>
<td>0.92</td>
<td></td>
</tr>
<tr>
<td>ShieldFL [58]</td>
<td>0.94</td>
<td>0.92</td>
<td>0.33</td>
<td>0.91</td>
<td>0.78</td>
<td>0.84</td>
<td>0.94</td>
<td>0.84</td>
<td></td>
</tr>
</tbody>
</table>

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).
3. Related Work

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

3.1. Secure Multi-party Computation (MPC)

MPC [40], [89] 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], [19], [22], [23], [51], the majority of the parties are considered to be honest and follow the protocol. Dishonest majority protocols [18], [30], [66], [71], 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 [43], [71], [72], corrupt parties follow the protocol, but are curious and tend to learn more information than intended. The malicious setting [28], [34], [50], [52] models scenarios, where corrupt parties can arbitrarily deviate from the protocol.

3.2. Poisoning Attacks in FL

The poisoning attacks [79] evaluated in SAFEFL can be broadly classified into four categories and the details are provided next.

1) **Label flipping (LF) [35], [79]:** 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 [35], where the new label is defined as \( l_{\text{new}} = L - l_{\text{old}} - 1 \), for \( L \) classes.

2) **Scaling [21]:** 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 [21], [35]:** Fang et al. [35] 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 [35] to optimize the attacks Krum, Trim, and FLTrust [21, §V].

4) **AGR-agnostic [78]:** 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 [62], which simply computes a (weighted) average of all inputs, we implement and evaluate 13 other robust aggregation schemes [25] 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 its local update chosen as the global update.

2) **Trim-mean [90]:** This method aggregates model parameters coordinate-wise by sorting their values in local model updates, removing the largest and smallest \( \beta \) values for a given parameter \( \beta \), and computing the mean of the remaining values as the final parameter value in the global update.

3) **Median [90]:** 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 [21]:** 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) [78]: DnC selects a random set of gradient coordinates \( r \), of size less than \( b \), and constructs a subsampled set \( \nabla_c \). The mean of \( \nabla_c \) is then calculated to obtain the centered set \( \nabla_c \). The algorithm computes projections along the top right singular eigenvector \( v \) and calculates a vector of outlier scores \( s \). A set of \( c : f \) gradients with the highest scores are removed, with the remaining gradients being considered “good” and added to a set. This is repeated for \( n\text{iter} \) 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 \( n\text{iter} \) good sets.

6–7) FoolsGold [38], 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 \( \alpha \) 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 [84]: 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 [21]) 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 [60]: 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 [87]: 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 [69]: This method aggregates local models through clustering, clipping, and adaptive noise addition. Clustering is performed using HDBSCAN [20] 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 \( \epsilon \) and \( \delta \).

12) FLOD [33]: 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 \( \tau \) are excluded from the average.

13) ShieldFL [58]: In ShieldFL, users normalize updates larger than \( \kappa \) 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 [47], 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 [24]. 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


