SAFELearn: Secure Aggregation for private FEderated Learning (Full Version)*

Hossein Fereidooni[1], Samuel Marchal[2], Markus Miettinen[1], Azalia Mirhoseini[3], Helen Möllering[4], Thien Duc Nguyen[1] Phillip Rieger[1], Ahmad-Reza Sadeghi[1], Thomas Schneider[4], Hossein Yalame[4], and Shaza Zeitouni[1]

[1] System Security Lab, Technical University of Darmstadt, Germany
{hossein.fereidooni, markus.miettinen, ducthien.nguyen, phillip.rieger, ahmad.sadeghi, shaza.zeitouni}@trust.tu-darmstadt.de

[2] Aalto University and F-Secure Corporation, Finland – samuel.marchal@aalto.fi

[3] Google, USA – azalia@google.com

[4] ENCRYPTO, Technical University of Darmstadt, Germany – {moellering, schneider, yalame}@encrypto.cs.tu-darmstadt.de

***Abstract*—Federated learning (FL) is an emerging distributed machine learning paradigm which addresses critical data privacy issues in machine learning by enabling clients, using an aggregation server (aggregator), to jointly train a global model without revealing their training data. Thereby, it improves not only privacy but is also efficient as it uses the computation power and data of potentially millions of clients for training in parallel.**

However, FL is vulnerable to so-called *inference attacks* by malicious aggregators which can infer information about clients' data from their model updates. Secure aggregation restricts the central aggregator to only learn the summation or average of the updates of clients. Unfortunately, existing protocols for secure aggregation for FL suffer from high communication, computation, and many communication rounds.

In this work, we present SAFELearn, a generic design for efficient private FL systems that protects against inference attacks that have to analyze individual clients' model updates using secure aggregation. It is flexibly adaptable to the efficiency and security requirements of various FL applications and can be instantiated with MPC or FHE. In contrast to previous works, we only need 2 rounds of communication in each training iteration, do not use any expensive cryptographic primitives on clients, tolerate dropouts, and do not rely on a trusted third party. We implement and benchmark an instantiation of our generic design with secure two-party computation. Our implementation aggregates 500 models with more than 300K parameters in less than 0.5 seconds.

***Index Terms*—Federated Learning, Inference Attacks, Secure Computation, Data Privacy**

I. Introduction

Federated Learning (FL) became a distributed machine learning (ML) paradigm since it was introduced by Google in 2017 [2]. It aims at improving privacy by enabling data owners to efficiently train a model on their joint training data with the help of a central aggregator and without sharing their potentially sensitive data with each other or with the aggregator. FL offers efficiency and scalability as the training task is distributed between many participants and executed in parallel [3], [4]. Possible applications include, e.g., next word prediction for mobile keyboards from Google [5], the analysis of medical data [6], communication between vehicles [7], and intrusion detection systems [8]. While FL leverages the power of the massive amount of data available at edge devices nowadays, it improves data privacy by enabling to keep data locally at the clients [5]. This becomes particularly relevant not only because of legal obligations such as the GDPR [9] and HIPAA [10], but also in general when working with personal and sensitive data like in the health sector where ML gets increasing attention. In applications being deployed on end-users' devices, FL helps to increase the acceptance as the user's data never leaves its device such that more users might be willing to contribute to a training. A real-world example for such a system is GBoard, a mobile keyboard for Android smartphones using FL for training word suggestions [5], which was downloaded more than a million times by the end of 2020 [11].

Despite these benefits, FL is vulnerable to adversarial attacks aiming at extracting information about the used training data. In these so-called *inference attacks*, the adversary can, for example, infer if a specific image was used in training an image classifier by inferring the *model updates* [12], [13]. This violates the principal design goal of FL, i.e., protecting data privacy. Several secure aggregation protocols have been proposed to address this problem by hindering the aggregator from analyzing clients' model updates [3], [14]–[23]. However, existing approaches are inefficient, impractical, and/or rely on a trusted third party (TTP) [15]–[17]. In particular, they are computationally expensive [3], [14], [21], [24], increase the number of communication rounds [14], [22], and do not tolerate dropouts [15], [16]. Especially the increase in communication rounds is problematic, as FL is typically used in a mobile setting where mobile or edge devices are involved and the network tends to be unstable, slow, and with low bandwidth [2]. Mobile devices regularly go offline such that dropouts must be tolerated. Most importantly, these aggregation schemes hinder the aggregator from accessing the local updates, therefore, making it impossible to analyze these updates for malicious client behavior that sabotage the training [25].

Our Contributions and Outline. In this work, we introduce SAFELearn, an efficient secure aggregation system, prohibiting

*Please cite the version of this paper published at IEEE Deep Learning and Security Workshop (DLS'21) [1].

access to model updates to impede powerful inference attacks on FL. In particular, we provide the following contributions after giving the preliminaries in §II:

- We survey state-of-the-art secure aggregation protocols for FL and analyze their limitations (§III).
- We introduce a generic design called SAFELearn for secure aggregation for FL. It is adaptable to various security and efficiency requirements and multiple aggregation mechanisms (§IV).
- We implement and benchmark an instantiation of SAFELearn using secure two-party computation on multiple FL applications and datasets (§V). Our system aggregates 500 models with more than 300K parameters in less than 0.5 seconds. Our implementation is available as open source at https://github.com/TRUST-TUDa/SAFELearn.

II. Preliminaries

In this section, we introduce FL and inference attacks on FL as well as the cryptographic building blocks we use.

A. Federated Learning (FL)

Federated Learning (FL) [2], [4] is a concept for distributed machine learning that links K clients and an aggregator A who collaboratively build a global model G. In each training iteration t, A chooses a subset of the K clients and sends the current global model G_{t-1} to them. Instead of sharing gradients after each training iteration as in standard distributed machine learning, each of these clients $i \in K$ then trains G_{t-1} on multiple batches of its training data for multiple epochs before sending the resulting locally updated model W_i to the aggregator. Then, A aggregates the received updates W_i into the global model G_t. FL results in less global training iterations than in standard distributed machine learning and, hence, in less communication.

Several aggregation mechanisms have been proposed for FL: (1) *Federated-Averaging* (FedAvg) [2], (2) *Krum* [26], (3) *Adaptive Federated Averaging* [27], and (4) *Trimmed mean or median* [28]. In this work, we focus on *FedAvg*, which is the original FL aggregation mechanism, because it is commonly applied in FL and related work on secure aggregation [3], [14], [19]–[21]. In FedAvg, the global model is updated by summing the weighted (by the number of training samples used to train it) models $G_t = \sum_{i=1}^{|K|} \frac{s_i \times W_i}{s}$, where K is the set of clients, $s_i = \|D_i\|$ for training data D_i of a client $i \in K$, $s = \sum_{i=1}^{|K|} s_i$, and W_i is client i's update [2]. To hinder a malicious client from exaggerating its dataset's size to amplify the effect of its update, previous works employ equal weights ($s_i = 1, s = K$) for all clients' contributions [2], [25], [26]. We adopt this approach in §IV.

B. Inference Attacks on FL

In an *inference attack*, an adversary aims at learning information about the data used for training a ML model. *Membership inference* attacks determine whether certain samples were used for training [29], *property inference* attacks infer properties of training samples independent of the original learning task [13], *distribution estimation* attacks estimate the proportions of training labels in the data [30], and *reconstruction* attacks reconstruct training samples [30]. Another distinction can be made between black box attacks, that are restricted to interpret the model predictions [30], and white box attacks, that use model parameters of either the trained model or from the clients' updates during the training [29], [30].

FL protects the privacy of the clients' data against inference attacks run by third parties, as they can only access the global model and cannot relate the information inferred from a global model to a specific client. Additionally, attacks on the global models tend to be weak and fail to achieve good attack performance [29]. However, the aggregator in FL has access to the local updates of each client making FL vulnerable to strong inference attacks by a corrupted aggregator. Thus, in this work, we aim at hindering the aggregator from accessing clients' update to prohibit powerful inference attacks that are leveraging individual local updates of clients while enabling efficient FL.

C. Secure Multi-Party Computation (MPC)

Secure Multi-Party Computation (MPC) enables the secure evaluation of a public function on private data provided by N mutually distrusting parties [31].

Secure two-party computation (STPC) [32]–[35], a special case of MPC with two parties ($N = 2$), allows two parties to securely evaluate a function on their private inputs.

Thereby, the parties have only access to so-called secret-shares of the inputs that are completely random and therefore do not leak any information. The real value can only be obtained if both shares are combined. STPC can be used in an outsourcing scenario [33], where an arbitrary number of weak but even malicious clients can secret-share their private inputs among two non-colluding but well-connected and powerful servers who then run the STPC protocol.

D. Homomorphic Encryption (HE)

Homomorphic Encryption (HE) enables computations on encrypted data. It allows to perform operations on a ciphertext, the decryption of which corresponds to algebraic operations on the plaintext. HE schemes can be classified into partially (PHE), somewhat (SHE), or fully homomorphic encryption (FHE). PHE [36] supports either multiplication or addition under encryption, SHE supports a bounded number of both, and FHE [37] supports both without limitations.

E. Differential Privacy

Although Differential Privacy (DP) [38] is not the focus of our work, we shortly summarize it here for the sake of completeness as it is used in some related works (cf. §III).

Informally, DP [38] randomizes the result of an evaluation (e.g., by adding noise) to reduce information leakage. More formally, a randomized algorithm $\mathcal{M}$ with domain $\mathcal{D}$ satisfies (ϵ, δ)-DP if for all adjacent datasets $d, d* \in \mathcal{D}$ and for all $S \subseteq \text{Range}(\mathcal{M})$ it holds, that $\Pr[\mathcal{M}(d) \in S] \leq e^{\epsilon} \Pr[\mathcal{M}(d*) \in S] + \delta$.

III. Related Work

Several works aim at improving data privacy in FL. They typically prevent access to the local updates using secret sharing techniques, encryption, and/or reduce information leakage by applying noise to achieve differential privacy (DP) [38]. We survey these works in the following. Tab. I shows a comparison between SAFELearn and previous works with respect to their usability and their privacy guarantees.

A. Secure Aggregation for FL with Secret Sharing

Bonawitz et al. [14] introduced secure aggregation for FL. Their protocol can tolerate client dropouts. They use blinding with random values, Shamir's Secret Sharing (SSS) [40], and symmetric encryption to prohibit access to local models. However, their aggregation requires at least 4 communication rounds between each client and the aggregator in each iteration. This causes a significant overhead for clients typically connected via WAN and with limited resources.

VerifyNet [17], and VeriFL [23] use the protocol of Bonawitz et al. [14]. VerifyNet and VeriFL add verifiability on top of [14] to guarantee the correctness of the aggregation, but these protocols rely on a trusted party to generate public/private key pairs for all clients.

Recently, the authors of [3] and [20] introduced secure aggregation protocols with polylogarithmic communication and computation complexity which reduce the overhead compared to [14]. Their key idea is to replace the star topology of the communication network in [14] by random subgroups of clients and to use secret sharing only for a subset of clients instead of for all client pairs. Both approaches require 3 rounds of interaction between the server and clients.

FastSecAgg [21] provides a secure aggregation based on the Fast Fourier Transform multi-secret sharing. It is robust against adaptive adversaries where the clients can adaptively be corrupted during the execution of the protocol. FastSecAgg is a 3 round interactive protocol for private FL.

Turbo-Aggregate [22] reduces the communication and computation overhead of secure aggregation over [14] (cf. Tab. II) and uses a circular communication topology. The main bottleneck of Turbo-Aggregated is its $\mathcal{O}(n/\log n)$ round complexity, where n is the number of updates/clients (cf. §IV).

SAFER [19] reduces communication costs in FL by compressing updates and combines it with a secure aggregation protocol based on arithmetic sharing. However, SAFER considers only training with less than 10 clients and no dropouts. Moreover, SAFER was only benchmarked on independent and identically distributed (IID) data such that it is unclear if it works with the typically non-IID data used in FL.

B. Secure Aggregation for FL with Encryption

Truex et al. [15] combine additively homomorphic encryption (HE) with DP but cannot tolerate client dropouts. Using HE results in a significant runtime overhead and their system also requires 3 rounds of communication. These aspects make it impractical for real-world FL.

EaSTFfly [18] uses either additively HE with packing or Shamir's secret sharing (SSS) [40] in combination with quantization. The clients share their gradients after each training iteration instead of using FL's FedAvg mechanism [14] which significantly increases the number of training iterations. FedAvg requires division which is not possible with additively HE/SSS. Furthermore, in EaSTFfly's HE protocol all clients have to hold the same secret key such that if a client colludes with the aggregator all updates can be decrypted.

BatchCrypt [24] reduces the encryption and communication overhead of HE-based aggregation with a batch encryption technique and requires only a single round of communication. Again, using expensive HE (like [15], [18]) makes it unusable for real-world training with FL.

HybridAlpha [16] uses functional encryption and DP. With functional encryption, public keys for all clients are derived from a private/public master key pair. It improves [15]'s runtime by a factor of $2\times$ and tolerates dropouts. However, HybridAlpha relies on a trusted party that holds the master keys and controls if the aggregator manipulates the aggregation weights.

POSEIDON [39] encrypts the complete FL process including the local training executed by the clients and, thus, adds significant computational overhead on each client's device. The authors suggest to reduce the clients' communication by combining them in a tree-like network instead of the classical star topology where each client directly communicates with the central aggregator. Additionally, a distributed bootstrapping efficiently refreshes ciphertexts and an alternating packing approach enhances the efficiency of neural network training under encryption. However, POSEIDON only supports clients' dropouts when the decentralized bootstrapping is not used.

Generally, all existing protocols for secure aggregation hinder the aggregators from deploying defenses against so-called *backdooring* or *poisoning attacks* [25], [41] that aim at injecting a "backdoor" into the ML model, i.e., the model is manipulated such that it misclassifies a small set of attacker-chosen inputs as attacker-chosen outputs. Bagdasryan et al. [25] show, inter alia, how a single client can manipulate FL by injecting a backdoor that causes green cars to be misclassified as birds. Such targeted image misclassification can, for example, be dangerous for face recognition systems deployed at airports. FLGUARD [42] and BaFFLe [43] combine secure aggregation with defenses against backdoor injections.

C. General Approaches to Privacy-preserving ML

Other works such as [44]–[47] combine secure computation techniques and/or DP with ML. However, these designs are often tailored for specific ML algorithms, and do not focus on FL such that they cannot handle client dropouts, do not scale to a large number of clients, use very expensive cryptography, and/or are not suitable for a distributed training.

IV. Private Federated Learning

In this section, we introduce SAFELearn for the secure aggregation of clients' updates in FL to hamper powerful inference attacks analyzing the clients' updates [29]. SAFELearn

TABLE I

COMPARISON OF PRIVACY-PRESERVING FL FRAMEWORKS. OUR PRIVACY ANALYSIS INCLUDES THE INVOLVEMENT OF A TRUSTED THIRD PARTY AND IF THE SCHEMES ARE ADAPTABLE TO ACTIVE SECURITY. OUR USABILITY ANALYSIS INCLUDES THE COMMUNICATION ROUND-EFFICIENCY, ROBUSTNESS TO DYNAMIC CLIENT DROPOUT, THE USAGE OF EXPENSIVE CRYPTOGRAPHIC OPERATIONS, AND THE AVAILABILITY OF OPEN-SOURCE CODE.

Proposed Approach	**Privacy**		**Usability**			
	No Trusted Party	**Extend-to-Active**	**Round-efficient**	**Dropout**	**No Expensive Operations**	**Open-Source**
Truex et al. [15]	✗*	✗	✗	✗	✗	✗
HybridAlpha [16]	✗*	✗	✗†	✓	✗	✗
Bonawitz et al. [14]	✓	✓‡	✗	✓	✗	✗
BatchCrypt [24]	✓	✗	✓	✓	✗	✓^{1}
VeriFL [23]	✓	✗	✗	✓	✗	✗
Choi et al. [20]	✓	✗	✗	✓	✗	✗
FastSecAgg [21]	✓	✗	✗	✓	✗	✗
SAFER [19]	✓	✓	✓	✗	✓	✗
POSEIDON [39]	✓	✓	✓	✓⋆	✗	✗
Bell et al. [3]	✓	✓	✗	✓	✗	✗
Turbo-Aggregate [22]	✓	✓	✗	✓	✗	✗
SAFELearn (This work)	✓	✓	✓	✓	✓	✓^{2}

$*$ Assume a third trusted party for key distribution
† Considering the key distribution communication round.
‡ Requires an extra round.
⋆ Can support dropouts when the decentralized bootstrapping is not used.
^{1} https://github.com/marcoszh/BatchCrypt
^{2} https://github.com/TRUST-TUDa/SAFELearn

has three major *design goals* to overcome the limitations of previous work (cf. §III): (G1) prohibiting access to individual model updates to counter powerful inference attacks, (G2) efficiency, and (G3) tolerance to outliers.

A. Adversary Model — Goals and Capabilities

The adversary is a semi-honest[1] aggregator such that we assume it follows the protocol honestly, but attempts to infer sensitive information about clients' data D_i from their model updates W_i [48], [49].

In the standard FL setting, the aggregator has access to all local model updates W_i, such that it can perform model inference attacks on each local model to extract information about the corresponding participant's data D_i used for training. Some existing attacks, e.g., [50], consider that the adversary (either a semi-honest aggregator or client) aims at inferring information about training data from the global model G_t. However, these attacks obtain negligible aggregated information about data (and cannot link it to individual clients) like the number of classes [50]. For example, Nasr et al. [29] show that the success of membership inference attacks on only the global model significantly degrades with an increasing number of clients. Therefore, our goal is to hide local models from the aggregator to impede powerful inference attacks while still enabling efficient and accurate FL.

B. SAFELearn

The simple and generic design of SAFELearn that realizes the adapted FedAvg with equal weights (cf. §II) in a private manner is depicted in Alg. 1. It takes the set of clients K, the initial global model G_0, and the number of training iterations T as input. Then, in each iteration $t \in [1, T]$, a random subset

[1]Can be extended to active security, cf. §V.

Algorithm 1 SAFELearn

```
1: Input: $K, G_0, T$   ▷ $K$: set of clients, $G_0$: initial global model, $T$: # iterations
2: Output: $G_T$   ▷ $G_T$ is the global model after $T$ iterations
3: for each training iteration $t \in [1, T]$ do
4:     for each client $i \in K_t \subseteq K$ do   ▷ $K_t \subseteq K$ is randomly chosen in every iteration.
5:         $[W_i] \leftarrow$ CLIENTUPDATE($[G_{t-1}]$)
6:     end for
7:     $[G_t] \leftarrow \sum_{i=1}^{|K_t|} [W_i]/|K_t|$
8: end for
```

of clients $K_t \subseteq K$ is chosen following the original design of FL [2]. Each client $i \in K_t$ receives the encrypted/secret shared global model $[G_{t-1}]$ from the aggregator(s) which it decrypts to train a new updated local model W_i. The client sends encrypted/secret shared update $[W_i]$ to the aggregator(s) for the aggregation of a new global model $[G_t]$ in Line 6.

Depending on the efficiency and security requirements of the concrete application, SAFELearn can be realized with fully homomorphic encryption (FHE), multi-party computation (MPC), or secure two-party computation (STPC). These secure computation techniques ensure that the aggregator cannot access clients' model updates and intermediate global models to effectively thwart powerful inference attacks (G1, cf. §IV).

1) FHE: If FHE with a *single* semi-honest server (acting as the aggregator) is used, $[G_t]$ and $[W_i]$ are the encryption of the models' parameters. The clients use a multi-party encryption scheme [51] for encrypting their updates. The secret key is securely split among the clients such that each of them can decrypt and access the global updates in plaintext for local training. The clients then re-encrypt the resulting local model updates and return it to the server who aggregates the encrypted data to the new global model $[G_t]$ using the additive homomorphic properties of the encryption scheme. Such FHE is computationally expensive and should only be used if using non-colluding servers is not possible due to the application and its security requirements.

2) MPC/STPC: If MPC/STPC is used, $[G_t]$ and $[W_i]$ are a set of secret shares of the models created with the Arithmetic Sharing technique [52] (cf. §V-A) and held by the $N \geq 2$ non-colluding servers. They jointly run the secure aggregation on these shares (i.e., the N servers together compose the aggregator). After the secure aggregation, in which the division is realized with Boolean circuit-based protocols called Yao's garbled circuits for STPC [32] or BMR for MPC [31] that can efficiently compute non-linear operations, the servers send the secret shares of the new global models to the clients who combine them to receive the plaintext global model for the next local training iteration. Afterwards, they again secretly share their updates and send one share to each server.

3) Secure Aggregation in SAFELearn: While we used FedAvg as an example in Line 6 of Alg. 1 given its popularity in the FL literature (cf. §II-A), different kinds of aggregation can also be realized with SAFELearn. Concretely, MPC and STPC support arbitrary computations expressed as Boolean circuit. Hence, also different aggregation mechanisms can be realized in a straightforward fashion. For example, the aggregation mechanism of Krum [26] consists of Argmin, multiplication, and addition operations. These can be realized by combination of different MPC/STPC protocols: Boolean sharing for the secure evaluation of Argmin and Arithmetic sharing for the secure evaluation of multiplication and addition operations. Similarly also other aggregation mechanisms (e.g., [27], [28]) can be realized by combining different MPC/STPC techniques. Moreover, we can also adopt other aggregation mechanisms for SAFELearn with FHE as FHE supports an unlimited number of additions and multiplications and can, e.g., also realize the (approximated) determination of minimums[2] [53], [54].

C. Privacy & Usability of SAFELearn

SAFELearn needs only 2 rounds of communication per iteration (Line 5). It allows an arbitrary number of clients to drop out and rejoin as the aggregator(s) can simply adjust the division factor $|K_t|$ by the number of clients that respond to CLIENTUPDATE(..). Thus, SAFELearn offers efficiency (G2, §IV) with respect to the number of communication rounds[3] and tolerance to outliers (G3, §IV).

The aggregating server(s) do only learn the number of clients in each training iteration and, hence, SAFELearn effectively prevents the individual aggregation server(s) from running inferences attacks on the clients' local model updates. Moreover, the aggregator(s) only hold secret shared or encrypted global models such that they also cannot run inference attacks on the global model. Even when the adversary controls a client, who still has access to the plaintext global model parameters, the aggregation of the models averages the parameters of all contributing clients and, thus, makes inference attacks harder and less effective [29]. Additionally, information extracted from a global model cannot be linked to a single client. Therefore, SAFELearn supports the anonymisation of the individual contributions.

To summarize, SAFELearn is a generic system for secure aggregation in FL and supports a wide range of applications by choosing the number of servers based on the specific security and efficiency requirements. It addresses all design goals (G1-G3) from the beginning of §IV. FHE/MPC/STPC support many operations such as addition, multiplication, and comparison. Those atomic operations also enable to privately realize aggregation functions beyond FedAvg, e.g., Krum [26]. For our MPC/STPC-based version of SAFELearn, this only requires to create the corresponding Boolean/Arithmetic circuit which can be automated with tools like HyCC [55]. Additionally, efficiency at the server side could be further improved by decrypting the intermediate global model in Line 6 of Alg. 1 *after* the aggregation and adding noise instead to achieve DP.

V. EXPERIMENTAL EVALUATION OF SAFELEARN WITH STPC

Our framework SAFELearn is generic and can be instantiated with one (FHE) or multiple non-colluding servers (MPC/STPC). We implement one instantiation of SAFELearn using STPC as it is often a good trade-off between security and efficiency. We securely outsource the computation of the SAFELearn algorithm to two servers that are (1) non-colluding and (2) semi-honest. These properties and assumptions are described and justified next.

1) Non-colluding semi-honest servers: We assume that the two servers are *non-colluding*. In our envisioned FL applications mentioned in §I, the two non-colluding servers could, e.g., be run by two competing antivirus software companies for network intrusion detection or by two competing smartphone manufacturers for word prediction. These parties are assumed to not collude to protect their business secrets and customers' data. Moreover, the two servers are assumed to be *semi-honest*, meaning that they honestly follow the protocol, but seek to learn as much information as possible. Service providers, like antivirus companies or smartphone manufacturers, have an inherent motivation to follow the protocol because they want to offer a privacy-preserving service to their customers and if cheating would be detected, this would seriously damage their reputation, which is the foundation of their business models. This assumption of non-colluding semi-honest servers is also justified by legal regulations like the GDPR [9] that mandate companies to properly protect users' data. Moreover, it allows to build highly efficient STPC protocols. Hence, it is very common in private ML applications, e.g., [34], [46].

2) Two servers: We choose $N = 2$ servers, i.e., STPC, as a reasonable trade-off between security and efficiency: With only one server, we would not need the non-collusion assumption, but this requires very expensive cryptographic primitives like fully homomorphic encryption [37] or several

[2]This typically has a higher computational overhead than with MPC/STPC.

[3]Its efficiency w.r.t. communication and computation heavily depends on its concrete instantiation and cannot be generally assessed. We benchmark it for an instantiation with STPC in §V to show its practicality.

rounds of interaction with the clients. As discussed in §I, it is beneficial to minimize the number of communication rounds because of the mobile setting with unstable and slow connections between clients and the aggregator in which FL is typically used.

Using protocols such as [44] with three or even more non-colluding servers of which at most one can be corrupted allows to construct even more efficient protocols than with two servers, but this has a larger attack surface because an attacker can attack any of the $N \geq 3$ servers and also more non-colluding parties have to be found to run these servers. However, our implementation can be easily extended to 3 semi-honest servers by using the ABY^3 [44] framework, N semi-honest servers by using the MOTION [31] framework, or p malicious servers by using MP-SPDZ [56].

A. Benchmarks

For our instantiation, we use a combination of two STPC techniques, which are implemented with state-of-the-art optimizations in the ABY framework [33]: Boolean sharing using Yao's garbled circuits [32] for secure evaluation of Boolean division circuits in a constant number of rounds, as well as Arithmetic sharing for secure evaluation of additions using the GMW protocol of Goldreich-Micali-Wigderson [52]. We use the PyTorch framework [57] for neural network training. All STPC results are averaged over 10 experiments and run on two separate servers with Intel Core i9-7960X CPUs with 2.8 GHz and 128 GB RAM connected over a 10 Gbit/s LAN with 0.2 ms RTT.

B. Applications

We test SAFELearn on three datasets for typical FL applications:

1) Natural Language Processing (NLP): We use a recurrent neural network with 20M parameters from two long short-term memory (LSTM) and one linear output layer [25]. In each iteration t, $K_t = 100$ clients are randomly chosen to train the model. We use the Reddit dataset from November 2017 [58] with 20.6M records. Each Reddit user with at least 150 posts and less than 500 posts is considered as a FL client. We generated a dictionary based on the most frequent 50 000 words.

2) Image Classification (IC): Following [25], we used the CIFAR-10 dataset [59] with 50 000 images and a lightweight version of ResNet-18 with 2.7M parameters from 4 convolutional layers and a fully connected output layer. We split the dataset among 100 clients as done in [25]. In each training iteration, the clients locally update the model with a learning rate of 0.1.

3) Network Intrusion Detection System (NIDS): We use the IoT NIDS DÏoT [8] with three datasets by [8], [60] and one self-collected dataset from homes and offices located in Germany and Australia. Following [8], we extracted device-type-specific datasets capturing the communication behavior of a smart weather station. We simulate the FL setup by splitting the data among 106 clients, each having three hours of traffic measurements and select 100 clients from them at random in each training iteration. The learning rate is 0.1.

C. Impact on the the accuracy of the resulting model

To measure SAFELearn's impact on the model's accuracy, we run experiments on all three datasets presented in §V-B. Tab. III shows our results as well as the experimental setup, including the number of local training epochs, the number of previously trained rounds and the total number of clients for that dataset, from which a subset of 100 clients is randomly chosen to perform the training in each training iteration. The results show that SAFELearn has the same accuracy as plaintext FedAvg.

D. Efficiency of SAFELearn

The results of our efficiency evaluation of SAFELearn with STPC between the two servers for the three datasets with different numbers of clients per training iteration, ranging from 10 to 500[4], are shown in Figs. 1 and 2. The runtime scales linearly with the number of clients and the communication is about constant. For NIDS, aggregating 500 models takes 0.5 seconds and the communication between the two servers is 8 MB. Even for the very large NLP model with more than 20M parameters, the aggregation takes less than 80 seconds and 316 MB between the two servers.

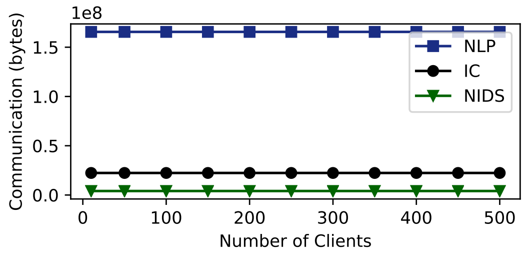

Fig. 1. Communication Costs per Server in SAFELearn

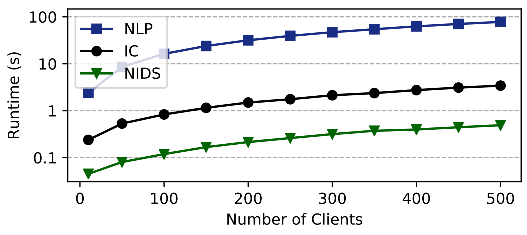

Fig. 2. Total Execution Time of SAFELearn

E. Analytical Complexity

Tab. II shows the substantially improved complexities of SAFELearn over the five previous works on secure aggregation for FL that consider dropouts and are not based on computationally expensive HE [3], [14], [20]–[22].

[4]We aim at assessing SAFELearn's communication and computation complexity here. Thus, this range is already sufficient to show the approximately constant communication costs and the linear scaling of the communication costs w.r.t. the number of clients.

TABLE II
COMPUTATION, COMMUNICATION, AND COMMUNICATION ROUNDS (BETWEEN SERVER AND CLIENTS) PER TRAINING ITERATION OF SAFELEARN AND RELATED WORKS BASED ON SECRET SHARING. HERE n IS THE TOTAL NUMBER OF LOCAL MODELS (I.E., NUMBER OF CLIENTS) AND m IS THE LENGTH OF MODEL UPDATES. BEST MARKED IN BOLD.

Approach	Computation (Server)	Communication (Server)	Computation (Client)	Communication (Client)	Rounds
Turbo-Aggregate [22]	$\mathcal{O}(m \log n \log^2 \log n)$	$\mathcal{O}(mn \log n)$	$\mathcal{O}(m \log n \log^2 \log n)$	$\mathcal{O}(m \log n)$	$n/\log n$
Bonawitz et al. [14]	$\mathcal{O}(mn^2)$	$\mathcal{O}(mn + n^2)$	$\mathcal{O}(mn + n^2)$	$\mathcal{O}(m + n)$	4
Bell et al. [3]	$\mathcal{O}(mn \log n + n \log^2 n)$	$\mathcal{O}(mn + n \log n)$	$\mathcal{O}(m \log n + \log^2 n)$	$\mathcal{O}(m + \log n)$	3
FastSecAgg [21]	$\boldsymbol{\mathcal{O}(m \log n)}$	$\mathcal{O}(mn + n^2)$	$\mathcal{O}(m \log n)$	$\mathcal{O}(m + n)$	3
Choi et al. [20]	$\mathcal{O}(mn \log n)$	$\mathcal{O}(n\sqrt{n \log n} + mn)$	$\mathcal{O}(n \log n + m\sqrt{n \log n})$	$\mathcal{O}(\sqrt{n \log n} + m)$	3
SAFELearn (This work)	$\mathcal{O}(mn)$	$\boldsymbol{\mathcal{O}(mn)}$	$\boldsymbol{\mathcal{O}(m)}$	$\boldsymbol{\mathcal{O}(m)}$	**2**

TABLE III
EXPERIMENTAL SETUP AND ACCURACY OF FEDAVG AND SAFELEARN FOR THE FL APPLICATIONS NATURAL LANGUAGE PROCESSING (NLP), IMAGE CLASSIFICATION (IC), AND NETWORK INTRUSION DETECTION SYSTEM (NIDS).

	NLP	IC	NIDS
Local Epochs	250	2	10
Pretrained Rounds	5 000	10 000	10
Available Clients	80 000	100	106
FedAvg	22.5%	91.7%	100.0%
SAFELearn (This work)	22.5%	91.7%	100.0%

VI. CONCLUSION

In this paper, we introduce SAFELearn, a generic private federated learning design that enables to efficiently thwart strong inference attacks that need access to clients' individual model updates. Moreover, SAFELearn tolerates dropouts and does not require expensive cryptographic operations, making it more efficient than previous works with respect to communication and computation. Furthermore, it does not rely on a trusted third party. Our evaluation shows that aggregating 500 models with more than 300K parameters takes less than 0.5 seconds on commodity hardware.

Future work can realize more instantiations of SAFELearn. Also the combination of privacy and security in FL which was considered to be contradicting by Bagdasaryan et al. [25] can be investigated. SAFELearn's design could also enable to integrate a defense against manipulations of malicious clients.

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), was co-funded by the DFG — SFB 1119 CROSSING/236615297 and GRK 2050 Privacy & Trust/251805230, and by the BMBF and HMWK within ATHENE. It was partially funded by the European Commission through the SHERPA Horizon 2020 project under grant agreement No. 786641. It was partially funded by the Private AI Collaborative Research Institute (PrivateAI) established by Intel, Avast and Borsetta.

REFERENCES

[1] 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 privateFEderated Learning," in *DLS*, 2021.

[2] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *International Conference on Artificial Intelligence and Statistics*, 2017.

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

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

[5] B. McMahan and D. Ramage, "Federated Learning: Collaborative Machine Learning without Centralized Training Data," in *Google Research Blog*. Google AI, 2017, https://ai.googleblog.com/2017/04/federated-learning-collaborative.html.

[6] T. S. Brisimi, R. Chen, T. Mela, A. Olshevsky, I. C. Paschalidis, and W. Shi, "Federated Learning of Predictive Models from Federated Electronic Health Records," *International Journal of Medical Informatics*, 2018.

[7] S. Samarakoon, M. Bennis, W. Saad, and M. Debbah, "Federated Learning for Ultra-Reliable Low-Latency V2V Communications," in *GLOBECOM*, 2018.

[8] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A.-R. Sadeghi, "DÏoT: A Federated Self-learning Anomaly Detection System for IoT," in *ICDCS*, 2019.

[9] "General Data Protection Regulation," 2018, https://eur-lex.europa.eu/eli/reg/2016/679/oj.

[10] "Health Insurance Portability and Accountability Act," 1996, https://www.govinfo.gov/content/pkg/PLAW-104publ191/pdf/PLAW-104publ191.pdf.

[11] "Gboard - the Google Keyboard - Apps on Google Play." [Online]. Available: https://play.google.com/store/apps/details?id=com.google.android.inputmethod.latin

[12] N. Carlini, C. Liu, Ú. Erlingsson, J. Kos, and D. Song, "The Secret Sharer: Evaluating and Testing Unintended Memorization in Neural Networks," in *USENIX Security*, 2019.

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

[14] K. 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.

[15] S. Truex, N. Baracaldo, A. Anwar, T. Steinke, H. Ludwig, and Y. Zhou, "A Hybrid Approach to Privacy-preserving Federated Learning," in *AISec*, 2019.

[16] R. Xu, N. Baracaldo, Y. Zhou, A. Anwar, and H. Ludwig, "Hybridalpha: An Efficient Approach for Privacy-preserving Federated Learning," in *AISec*, 2019.

[17] G. Xu, H. Li, S. Liu, K. Yang, and X. Lin, "VerifyNet: Secure and Verifiable Federated Learning," in *Transactions on Information Forensics and Security*, 2020.

[18] Y. Dong, X. Chen, L. Shen, and D. Wang, "EaSTFLy: Efficient and Secure Ternary Federated Learning," in *Computers & Security*, 2020.

[19] C. Beguier and E. Tramel, "SAFER: Sparse Secure Aggregation for Federated Learning," 2020, https://arxiv.org/abs/2007.14861.

[20] B. Choi, J.-y. Sohn, D.-J. Han, and J. Moon, "Communication-Computation Efficient Secure Aggregation for Federated Learning," 2020, https://arxiv.org/abs/2012.05433.

[21] S. Kadhe, N. Rajaraman, O. O. Koyluoglu, and K. Ramchandran, "FastSecAgg: Scalable Secure Aggregation for Privacy-Preserving Federated

Learning," *ICML Workshop on Federated Learning for User Privacy and Data Confidentiality*, 2020.

[22] J. So, B. Güler, and A. S. Avestimehr, "Turbo-Aggregate: Breaking the Quadratic Aggregation Barrier in Secure Federated Learning," *Journal on Selected Areas in Information Theory*, 2021.

[23] X. Guo, Z. Liu, J. Li, J. Gao, B. Hou, C. Dong, and T. Baker, "VeriFL: Communication-Efficient and Fast Verifiable Aggregation for Federated Learning," *TIFS*, 2020.

[24] 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.

[25] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to Backdoor Federated Learning," in *AISTATS*, 2020.

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

[27] L. Muñoz-González, K. T. Co, and E. C. Lupu, "Byzantine-Robust Federated Machine Learning through Adaptive Model Averaging," 2019, https://arxiv.org/abs/1909.05125.

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

[29] 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 *S&P*, 2019.

[30] A. Salem, A. Bhattacharya, M. Backes, M. Fritz, and Y. Zhang, "Updates-leak: Data Set Inference and Reconstruction Attacks in Online Learning," in *USENIX Security*, 2020.

[31] L. Braun, D. Demmler, T. Schneider, and O. Tkachenko, "MOTION–A Framework for Mixed-Protocol Multi-Party Computation," 2020.

[32] A. C.-C. Yao, "How to Generate and Exchange Secrets," in *FOCS*, 1986.

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

[34] A. Patra, T. Schneider, A. Suresh, and H. Yalame, "ABY2. 0: Improved Mixed-protocol Secure Two-party Computation," in *USENIX Security*, 2020.

[35] H. Yalame, H. Farzam, and S. Bayat-Sarmadi, "Secure Two-Party Computation Using an Efficient Garbled Circuit by Reducing Data Transfer," in *Applications and Techniques in Information Security*, 2017.

[36] P. Paillier, "Public-key Cryptosystems Based on Composite Degree Residuosity Classes," in *EUROCRYPT*, 1999.

[37] C. Gentry, "A Fully Homomorphic Encryption Scheme," Ph.D. dissertation, Stanford University, 2009.

[38] C. Dwork and A. Roth, "The Algorithmic Foundations of Differential Privacy," in *Foundations and Trends in Theoretical Computer Science*, 2014.

[39] S. Sav, A. Pyrgelis, J. R. Troncoso-Pastoriza, D. Froelicher, J.-P. Bossuat, J. S. Sousa, and J.-P. Hubaux, "POSEIDON: Privacy-Preserving Federated Neural Network Learning," in *NDSS*, 2021.

[40] A. Shamir, "How to Share a Secret," in *Communications of the ACM*, 1979.

[41] T. D. Nguyen, P. Rieger, M. Miettinen, and A.-R. Sadeghi, "Poisoning Attacks on Federated Learning-Based IoT Intrusion Detection System," in *Workshop on Decentralized IoT Systems and Security @ NDSS*, 2020.

[42] T. D. Nguyen, P. Rieger, H. Yalame, H. Möllering, H. Fereidooni, S. Marchal, M. Miettinen, A. Mirhoseini, A.-R. Sadeghi, T. Schneider, and S. Zeitouni, "FLGUARD: Secure and Private Federated Learning," 2021, https://ia.cr/2021/025.

[43] S. Andreina, G. A. Marson, H. Möllering, and G. Karame, "BaFFLe: Backdoor Detection via Feedback-based Federated Learning," in *ICDCS*, 2021.

[44] P. Mohassel and P. Rindal, "ABY^3: A Mixed Protocol Framework for Machine Learning," in *CCS*, 2018.

[45] N. Dowlin, R. Gilad-Bachrach, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy," in *ICML*, 2016.

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

[47] S. Wagh, S. Tople, F. Benhamouda, E. Kushilevitz, P. Mittal, and T. Rabin, "FALCON: Honest-Majority Maliciously Secure Framework for Private Deep Learning," *PETS*, 2021.

[48] A. Pyrgelis, C. Troncoso, and E. De Cristofaro, "Knock Knock, Who's There? Membership Inference on Aggregate Location Data," in *NDSS*, 2018.

[49] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership Inference Attacks Against Machine Learning Models," in *S&P*, 2017.

[50] L. Wang, S. Xu, X. Wang, and Q. Zhu, "Eavesdrop the Composition Proportion of Training Labels in Federated Learning," 2019, https://arxiv.org/abs/1910.06044.

[51] C. Mouchet, J. R. Troncoso-Pastoriza, and J.-P. Hubaux, "Multiparty Homomorphic Encryption from Ring-Learning-With-Errors," 2020, https://ia.cr/2020/304.

[52] O. Goldreich, S. Micali, and A. Wigderson, "How to Play ANY Mental Game," in *STOC*, 1987.

[53] J. H. Cheon, D. Kim, D. Kim, H. H. Lee, and K. Lee, "Numerical Method for Comparison on Homomorphically Encrypted Numbers," in *ASIACRYPT*, 2019.

[54] J. H. Cheon, D. Kim, and D. Kim, "Efficient Homomorphic Comparison Methods with Optimal Complexity," in *ASIACRYPT*, 2020.

[55] N. Büscher, D. Demmler, S. Katzenbeisser, D. Kretzmer, and T. Schneider, "HyCC: Compilation of Hybrid Protocols for Practical Secure Computation," in *CCS*, 2018.

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

[57] "Pytorch," 2019, https://pytorch.org.

[58] "Reddit dataset," 2017, https://bigquery.cloud.google.com/dataset/fh-bigquery:reddit_comments.

[59] A. Krizhevsky and G. Hinton, "Learning Multiple Layers of Features from Tiny Images," Tech. Rep., 2009, https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

[60] A. Sivanathan, H. H. Gharakheili, F. Loi, A. Radford, C. Wijenayake, A. Vishwanath, and V. Sivaraman, "Classifying IoT Devices in Smart Environments Using Network Traffic Characteristics," in *Transactions on Mobile Computing*, 2018.