MPCFL: Towards Multi-party Computation for Secure Federated Learning Aggregation

Hiroki Kaminaga*
Information Security Research Institute,Cybernetica AS
Tallinn, Estonia
hiroki.kaminaga@cyber.ee

Feras M. Awaysheh
Institute of Computer Scince, Delta center,Tartu University
Tartu, Estonia
feras.awaysheh@ut.ee

Sadi Alawadi
Department of Computer Science, Blekinge Institute of Technology
Karlskrona, Sweden
sadi.alawadi@bth.se

Liina Kamm
Information Security Research Institute,Cybernetica AS
Tartu, Estonia
liina.kamm@cyber.ee

ABSTRACT

In the rapidly evolving machine learning (ML) and distributed systems realm, the escalating concern for data privacy naturally comes to the forefront of discussions. Federated learning (FL) emerges as a pivotal technology capable of addressing the inherent issues of centralized data privacy. However, FL architectures with centralized orchestration are still vulnerable, especially in the aggregation phase. A malicious server can exploit the aggregation process to learn about participants' data. This study proposes MPCFL, a secure FL algorithm based on secure multi-party computation (MPC) and secret sharing. The proposed algorithm leverages the Sharemind MPC framework to aggregate local model updates for securely formulating a global model. MPCFL provides practical mitigation of trending FL concerns, e.g., inference attack, gradient leakage attack, model poisoning, and model inversion. The algorithm is evaluated on several benchmark datasets and shows promising results. Our results demonstrate that the proposed algorithm is viable for developing secure and privacy-preserving FL applications, significantly improving all performance metrics while maintaining security and reliability. This investigation is a precursor to deeper explorations to craft robust FL aggregation algorithms.

KEYWORDS

Federated Learning, Multi-party Computation, Secret Sharing, Privacy-preserving, Data Security

ACM Reference Format:
Hiroki Kaminaga, Feras M. Awaysheh, Sadi Alawadi, and Liina Kamm. 2023. MPCFL: Towards Multi-party Computation for Secure Federated Learning Aggregation. In *2023 IEEE/ACM 16th International Conference on Utility and Cloud Computing (UCC '23), December 4–7, 2023, Taormina (Messina), Italy.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3603166.3632144

UCC '23, December 4–7, 2023, Taormina (Messina), Italy

ACM ISBN 979-8-4007-0234-1/23/12…$15.00
https://doi.org/10.1145/3603166.3632144

1 INTRODUCTION

Machine learning (ML) has become a transformational tool from healthcare and finance to social networking. In particular, the emergence of deep neural networks (DNNs) has brought about revolutionary changes and has had a tremendous impact on a wide range of fields, from image processing to recent large language models (LLMs) [13]. At its core, ML relies on vast amounts of data to make predictions without explicitly programming its mechanism. This data-driven nature has led to growing privacy concerns. Many ML models use personal and sensitive big data, especially those in the healthcare and finance sectors [7]. While these data are essential for improving model accuracy, unauthorized access or compromise can cause significant personal and financial harm. In addition, in the case of models that identify intractable diseases, for example, the scarcity of the data itself creates an incentive to collaborate with other data holders. However, with laws like the European Union's General Data Protection Regulation (GDPR) [2] and the California Consumer Privacy Act (CCPA) [9], organizations are mandated to protect user data, making privacy an ethical and legal obligation. Thus, handling data operations at the network edge, closer to the data source, is a convenient solution [8]

Federated learning (FL) is an emerging paradigm that allows ML models to be trained across multiple devices or servers while keeping data localized [26]. Instead of centralizing data, the model is sent to the edge, closer to the data source. This approach enables compliance with the aforementioned laws and promotes cooperation between devices and data holders while preserving privacy. However, sharing models, like in the aggregation phase, raises new privacy issues [24, 31]. One of the significant problems is model inversion [16], where an attacker can infer the original data from a shared model. Suppose a model was overfitted in the first place. In that case, the model is likely a close approximation of the original data structure, so reconstruction is relatively easy, especially for outliers. The gradient leakage attack [33] exposes or leaks sensitive information from the gradient updates shared between the nodes and a central server. Secure multi-party computation (MPC) in FL primarily aims to provide secure aggregation [12]. Specifically, it prevents a central aggregator (or other participants) from learning model updates from individual contributors. MPC enables data merging and computation while keeping the data secret, shared or encrypted.

This paper describes and demonstrates a proof-of-concept solution that uses secret sharing to perform secure FL aggregations over DNN models. As the technical solution, we use Sharemind MPC [10], a framework for privacy-preserving computations, where developers can efficiently program and test their algorithms without knowing the underlying cryptographic algorithms in detail.

The rest of this paper is organized as follows. Section 2 discusses the study background. In Section 3 we discuss related work, and in Section 4, we draft the problem statement and motivation scenario. In Section 5, we discuss our methodology and experiment setup. The experiment results and findings are discussed in Section 6. We discuss open challenges, future work, and opportunities in Section 7 and conclude in Section 8.

2 BACKGROUND

In this section, we discuss Multi-Party Computation (MPC) and the associated security threats on the FL aggregation layer, highlighting the essential challenge of maintaining collaborative model integrity without compromising privacy.

2.1 Secure multi-party computation

Secure multi-party computation (MPC) computes a function f between a group of parties so that none of the parties gains any other knowledge about the input except for what can be inferred from the end result. For example, assume that there is a set of parties $p_1, ..., p_n$ who want to jointly compute $y = f(x_1, ..., x_n)$, where each p_i knows the corresponding secret share x_i. In an MPC protocol, f is correctly computed so that no party p_i learns any information other than its own input x_i and the result y.

The MPC parties can be divided into input parties who input their data, computing parties who carry out the computations, and result parties who receive the results. One entity can have several of these roles, e.g., an input party can be one of the computing parties and also get the results as a result party.

One of the primitives that can be used for MPC is linear secret sharing. Secret sharing is a cryptographic technique for taking a secret s and splitting it into multiple shares, $s_1, s_2, \ldots, s_n$, distributing these shares among parties P_i. For the reconstruction of the secret s, the parties must collaborate, each bringing their respective share into the collective pool. A secret share is determined by means of a sharing algorithm. In case of linear secret sharing between n computing parties, the value s is secret-shared in the following way: the input party p chooses $r_1, \ldots, r_{n-1}$ uniformly from $\mathbb{Z}_q$ and computes $r_n = s - r1 - dots - r_{n-1} \bmod q$. The input party then sends exactly one share to each computing party. It is impossible to restore the original data with $n - 1$ or fewer shares. To reconstruct the secret, all shares must be combined.

This way, secret sharing allows a group of parties to jointly compute a function over their inputs while keeping the data private and only revealing the result. This concept underpins the secure MPCFL, leveraging the distribution of a secret among multiple parties to ensure collective security and confidentiality.

2.2 Threats on the FL aggregation layer

The aggregation layer in FL is a critical component where model updates from various clients are combined to produce a global model. This layer is susceptible to several threats, requiring secure strategies to mitigate these threats effectively. These threats include:

DDoS (Distributed Denial of Service: A DDoS attack in the context of FL could involve overwhelming the central server with a flood of model updates or other traffic, rendering the server unable to process legitimate updates from the participating agents. This can disrupt the aggregation process, preventing the global model from being updated and potentially causing downtime in the FL system.

Model Poisoning: An attacker sends malicious model updates to the central server with the intention of corrupting the global model. These malicious updates can influence the aggregation process, leading to a compromised global model that behaves unpleasantly.

Model Inference: Model inference attacks aim to extract information about the training data by analyzing the model updates. If the aggregation layer is not secure, attackers could potentially use the model updates to gain insights into the training data.

Gradient Leakage: Similarly to the model inference attacks, attackers might be able to infer information about the local data through the analysis of model gradients.

Model Inversion: Like to the previous two, an attacker uses access to the model updates to infer details about the underlying data used for training.

Securing the aggregation layer in FL is crucial to prevent these various types of attacks that can compromise the integrity of the global model, breach user privacy, and disrupt the overall aggregation process.

3 RELATED WORK

Federated learning [26] is a methodology where model training is done in separate data nodes, and the resulting models are aggregated at a central node or peer-to-peer. In a centralized setting, the centralized analysis node chooses a model to be trained and sends it to data nodes. The data nodes train the model locally and send the results back. The central analysis node combines the results. In a decentralized setting, the communication and coordination between the nodes is more complex but prevents the central node from becoming a bottleneck or single point of failure. In this work, we will be looking at centralized federated learning. However, the solution can easily be adapted to the decentralized environment.

To address the security and privacy issues that arise with sharing the models for aggregation, secure aggregation can be used [12, 25, 28, 32]. Secure multi-party computation (MPC) is a methodology where two or more parties compute a function without seeing the private input values of the other parties. Secret sharing, garbled circuits, or homomorphic encryption can be used to enable MPC [15]. MPC has been researched for decades and the technology has been used for different real-world applications to alleviate privacy concerns [4]. Even though MPC versions of any algorithm can be implemented from the ground up, several programmable MPC frameworks are available that allow an implementer to focus on the algorithm rather than the underlying cryptography [18]. In this regard, both FL and MPC can be seen as privacy-by-design technologies [6]. This paper implements our proof-of-concept prototype using the Sharemind MPC platform [10].

Sharemind MPC [10] is a platform for privacy-preserving data processing. It uses secret sharing as the secure storage method and allows computations on the secret shared values without having to declassify them. The input values are secret shared by input parties, and the shares are sent to computing parties so that each gets one share from which they cannot infer information about the original value. The computing parties all have a copy of the computation algorithm and use MPC protocols to execute them to compute secret shared results from secret shared data. The result shares are sent to the result parties, who declassify them into result values. In Sharemind MPC, communication between the input, computing, and result parties is implemented over authenticated secure channels (TLS). Sharemind MPC provides end-to-end security for data processing if a sufficient number of computing parties follow the protocol. If this requirement is not met, it may result in a wrong result or breach of privacy. Currently, efficient protocol sets exist for two and three computing parties. More than three parties increase the communication complexity, which is the main cause of the overhead of these systems. To achieve better computational performance, Sharemind MPC takes advantage of the single instruction, multiple data (SIMD) paradigm. Multiple instances of the same operation are carried out at once with amortised performance [29]. Sharemind MPC has been used to preserve privacy in real-world studies [11] and has been shown to work with statistical machine learning algorithms [27].

Deep neural networks (DNNs) are a special type of neural network characterized by having multiple layers between input and output layers. In 2006, Hinton et al. [19] introduced deep belief nets, demonstrated their capabilities and opened the door to further development. In recent years, as seen in the success of large language models (LLMs) [13], DNNs are changing our lives greatly. The DNN training process is to optimize their networks' weights by minimizing a loss function, typically by stochastic gradient descent (SDG) or its variants [17].

4 PROBLEM FORMULATION

This paper focuses on how to train models collaboratively across multiple data sources without revealing individual data points, models nor model updates. With the benefit of FL, training data remains local. However, privacy and confidentiality issues may arise during the model aggregation phase, where local models are combined to generate a global model. An adversary accessing these updates can potentially infer information about the local datasets, which in turn can lead to privacy leaks.

4.1 Scenario: Cross-Departmental Fraud Detection

In the digital era, the increasing transparency and accountability in financial transactions of public services have become crucial, especially in the realm of government subsidies that span areas like education, housing, and agriculture. The multitude of these subsidies and the attendant databases maintained by various governmental departments, while ensuring data integrity on the one hand, inadvertently contribute to an isolated approach to fraud detection on the other. This isolation, dictated by data privacy concerns, often results in algorithms that, despite being adept at department-specific fraud patterns, are blind to cross-departmental patterns that may suggest fraud.

Modern governments comprise many departments, each maintaining individualized records and deploying isolated fraud detection mechanisms. Though this modular approach is practical for domain-specific fraud detection, it poses a challenge when addressing complex fraud scenarios across multiple departments. For instance, individual subsidy claims across departments may appear genuine when viewed in isolation. However, a combined, holistic view might reveal patterns indicative of fraudulent activity.

To address this challenge, the proposed MPCFL architecture synergizes FL with MPC for collaborative fraud detection. The proposed architecture functions as follows. Each department starts with training a local fraud detection model on its dataset. Post-local training, model updates, or gradients for a global model are produced. Instead of transmitting these raw gradients directly, each department creates shares of its update using secret sharing. A secure central server aggregates these shares without accessing the raw data from any department nor seeing the contents of the update. The resulting global fraud detection model thus encapsulates knowledge from all departments, enabling it to identify fraud patterns that might escape department-specific models.

Consider a hypothetical scenario wherein an entity solicits subsidies across multiple sectors quickly. For instance, an individual might simultaneously apply for educational aid, housing benefits, and agricultural grants. While individual departments might process each of these claims without raising flags, the integrated model developed via our proposed FL and MPC approach would recognize the anomalous pattern in these individual solicitations, potentially identifying a fraud attempt.

The combination of the FL collaborative strengths of FL and the privacy-preserving nature of MPC offers an innovative solution for governments. This architecture provides a proactive mechanism to detect and counteract subsidy frauds, ensuring the optimal and genuine allocation of resources without compromising the privacy of departmental data.

In the next sections, we formally model the proposed architecture. We also demonstrate its methodology and provide a proof of concept experiment to validate the primary contribution of MPCFL.

4.2 MPCFL Formulation

We assume that there are N clients in the FL. Each client k has a local dataset D_k and trains a local DNN model $M^{(k)}$ using only local D_k. For simplicity, let $W_j^{(k)}$ represent the parameters of the k^{th} client's DNN model.

4.2.1 *Weighted Average Aggregation.*

The global model is updated by taking the weighted average of the local updates

$$W^{(global)} = \sum_{k=1}^{N} \alpha_k \times W^{(k)} \quad , \tag{1}$$

where α_k is the weight for each client, according to the local dataset size, determined as follows:

$$\alpha_k = \frac{|D_k|}{\sum_{k=1}^{N} |D_k|} \quad . \tag{2}$$

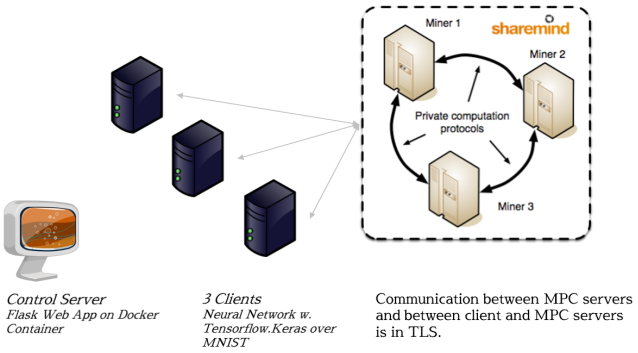


Figure 1: MPCFL Architecture prototype.

4.2.2 *Secure aggregation using MPC.*

We use additive three-party secret-sharing in our prototype. Let $[\![x]\!]$ denote the secret-shared value x among all the computing parties. After locally updating a model, each client secret-shares its local update parameters $W^{(k)}$ and sends $[\![W^{(k)}]\!]$ to each computing party. The parties jointly compute the global $[\![W_j^{(global)}]\!]$in a privacy-preserving manner. The secure aggregation for $[\![W]\!]$ is written as

$$[\![W^{(global)}]\!] = AddFloat([\![\alpha_1 W^{(1)}]\!], \ldots, [\![\alpha_N W^{(N)}]\!]) \ , \quad (3)$$

where $AddFloat()$ represents a floating-point addition as defined in [20]. The global model parameters are reconstructed only after they are sent back to each client. While privacy-preserving floating-point addition has a communication overhead, it has the same precision as the computation would have in the open. Further communication optimisation can be done with using fixed-point numbers instead, however we have decided for the more precise option and showing that even in this case, the privacy-preserving aggregation is feasible.

5 METHODOLOGY

We utilized Sharemind MPC within the context of federated learning to ensure secure aggregation, enabling the construction of the final global ML model through the fusion of locally trained models across multiple client datasets. To assess the viability of this privacy-preserving aggregation method and the practicality of our solution, we have implemented a proof-of-concept prototype. The architectural diagram of our prototype is illustrated in Figure 1.

Considering the roles (input, computing, result parties) of the MPC setting, the clients are input parties, sending their updated models to the computing parties, and result parties, receiving the global model from the computing parties. The computing parties in the prototype are three parties who are considered to be independent from each other. These parties could be chosen from among the input parties, but they can also be contracted third parties. Sharemind MPC enforces access control on each local update, so no client can access nor modify the other client models.

In our experiments, we engaged three clients (referred to as contributors or nodes) within the federation network to validate our approach, with the understanding that this number can be scaled as needed. Also, we used three benchmark datasets: MNIST [23],

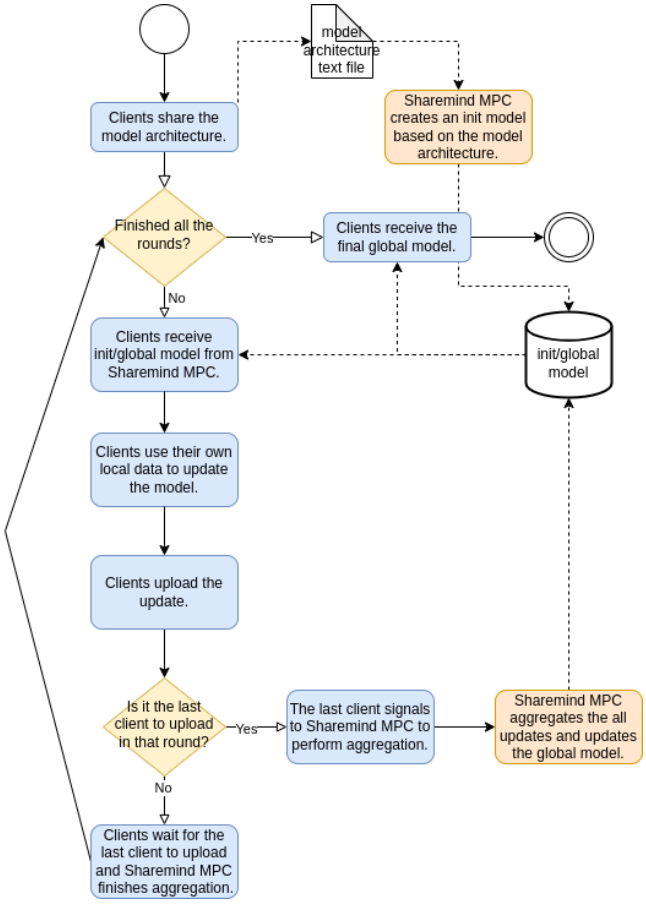


Figure 2: MPCFL workflow diagram.

CIFAR10 [22], and CASA [14]. MNIST and CIFAR10 are used for an image classification task and CASA for daily human activity recognition task. Specifically, the MNIST and CIFAR10 datasets, each consist of 60,000 training samples and 10,000 test samples and 10 classes. The CASA dataset is similarly made up of 10 classes but with 163,689 and 40,922 training and test samples, respectively (80:20 split rate). We randomly and equally distribute these among the three clients.

We set up a control server—a simple web application server for synchronizing updates between clients. Each client sends a signal to the control server every time it receives an initial or global model from the computing parties and each time it uploads an update to the computing parties. The last uploading client in each FL round sends a signal to computing parties to perform aggregation. After the clients have completed the upload phase, they wait until the last client finishes uploading the model. Subsequently, all contributions are aggregated by the computing parties to build the global model, and a copy of this global model is sent to each client. The control server responsible for orchestrating this process has been implemented using Flask.

In the following, we define one round as a set of three actions: (1) a client receives an initial or global model from the computing

Table 1: Details of the DNN Models used in the prototype

Parameters	MNIST	CIFAR-10	CASA
Input Shape	(28, 28, 1)	(32, 32, 3)	(1, 36)
Layers	5 Conv + 2 Dense	6 Conv + 2 Dense	1 LSTM (100) + 2 Dense
Neurons in Dense Layers	64, 10	128, 10	32, 10
Activation Functions	ReLU	ReLU	ReLU
Batch Normalization	No	Yes	No
MaxPooling	Yes	Yes	No
Dropout	No	0.3	No
Loss Function	Categorical Cross-Entropy	Categorical Cross-Entropy	Categorical Cross-Entropy
Optimizer	Adam	Adam	Adam
Learning Rate	0.001	0.001	0.001
Batch Size	32	32	32
Local Epochs	2	2	2
Aggregation Time*	9.98s ± 0.13s	16.43s ± 0.32s	6.78s ± 0.05s
Upload Time*	4.58s ± 0.31s	10.10s ± 0.54s	3.06s ± 0.49s
Download Time*	4.83s ± 0.21s	5.53s ± 0.27s	4.32s ± 0.17s

***Average time ± 2×SD of 10 repetitions using Sharemind MPC. The unit is seconds.**

parties, (2) updates the model using local data, and (3) uploads it to the computing parties. Each update only takes two epochs. We ran the model update for two hundred communication rounds.

For simplicity, we implemented weighted average for the aggregation in the prototype. As the data are volume-wise evenly distributed among clients, it is the same as average for this proof-of-concept implementation.

The activity diagram of the whole workflow is shown on Figure 2. The three DNN model architectures we used in the prototype are described in Table 1 with the dataset used for each. The models are implemented using the Tensorflow Keras API.

The initial model is created based on the chosen architecture. In the prototype, the architecture is sent in a text file to Sharemind MPC at the first round, but it can also be distributed via the control server. In accordance with the default Keras initializer[1], Sharemind MPC adopts the glorot uniform as its initializer for weights. Models are uploaded from the clients to Sharemind MPC.

5.1 MPCFL algorithm

MPCFL leverages secret sharing to ensure that no single entity can reconstruct the model updates from individual parties, thus enhancing privacy. Participating clients train or update their respective models and produce secret shares of their updates. Computing parties aggregate these updates using secret sharing protocols and without declassifying them. Following this aggregation, a consolidated global update is generated and sent back to the FL clients.

Algorithm 1 describes the MPCFL process from the client side. Each client node first initialises the process by computing its weight based on its local dataset D (line 8), and initialising the model based on the agreed model architecture (line 9). Next, the client iterates steps 11-15 for r rounds, where r is agreed on between the clients beforehand. The iteration starts with the client training the model based on their local data (11). Then the client secret shares the obtained model using a secret sharing scheme (12), creating as many shares as there are computing parties. For our prototype, we use a three party additive secret sharing scheme as described in [10]. Next, the client sends exactly one share of the model to each of the computing parties (13). The client also sends the weight. If the weight does not change, it can be done just once instead of during every update cycle. Now the client waits until the computing parties have finished and send back the shares of the global model (14). Having received a share from each computing party, the client then combines them to obtain a declassified global model G (15). Once r rounds have been carried out, the final aggregated model is returned as the result.

Algorithm 1 MPCFL Client Node

1: **function** MPCFLClient(A, D, r)
2: **Input:**
3: A: Model architecture from control server,
4: D: local dataset,
5: r: number of update rounds.
6: **Output:**
7: G: global aggregated model.
8: $\alpha \leftarrow$ weight based on D
9: $G \leftarrow$ initial model based on A
10: **for** $i \in [1, r]$ **do**
11: $G \leftarrow$ Update(G, D)
12: $[\![G]\!] \leftarrow$ Classify(G)
13: Send($computingparties, [\![G]\!], \alpha$)
14: Receive($computingparties, [\![G]\!]$)
15: $G \leftarrow$ Declassify($[\![G]\!]$)
16: **end for**
17: **return** G
18: **end function**

[1]https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dense

Algorithm 2 MPCFL Computing Party

1: **function** MPCFLComputingParty
2: $n \leftarrow$ number of clients from the control server
3: **for** $i \in [1, n]$ **do**
4: Receive($client_i$, $[\![M^{(i)}]\!]$, α_i)
5: **end for**
6: $[\![G]\!] = \alpha_1[\![M^{(1)}]\!] + \cdots + \alpha_n[\![M^{(N)}]\!]$
7: **for** $i \in [1, n]$ **do**
8: Send($client_i$, $[\![G]\!]$)
9: **end for**
10: **end function**

Algorithm 2 describes the MPCFL process from a computing party's side. All the computing parties request the number of clients n from the control server (step number 2). Then, each computing party waits until all the client nodes have sent their model shares and weights (step number4). Each computing party only receives the shares that are meant for them, meaning that the data looks like randomness and gives them no additional information about the values. When all the parties have completed the sending, the computing parties obliviously aggregate the model, using floating-point addition and multiplication from [20] without declassifying the clients' models (step number 6). They conclude the round by sending their shares of the global model back to all the client nodes (step number 8). The computing parties then wait until the clients initiate the next round by sending the next model shares.

6 RESULTS

In this section, we discuss our experiment results and the paper main findings.

The experiments were conducted using the Sharemind cluster and a laptop. The Sharemind cluster consists of three servers, all of which are hosted by Cybernetica. Each server has 2 CPUs (Intel X5670 2.93GHz/6.4GT/12M) and 48 GB RAM, respectively, connected via 4x1Gb LAN. While, the laptop is HP OMEN 15 with AMD(R) Ryzen 7 5800h CPU, NVIDIA GeForce RTX 3070 Mobile GPU, and 16GB RAM. All client nodes and the control server are hosted on this laptop.

We validate our approach over three different datasets belonging to different tasks associated with distinct models as shown in Table 1. The proposed approach was evaluated in terms of communication and computation cost, model performance (accuracy), the model prediction agreement with the real value (Cohen's kappa), F1 and AUC measures over the client's local data in federated learning settings. The last 3 rows of Table 1 contain the computation and communication times for secure aggregation. Fig.3 shows the performance of a global model for all tasks as bar plots trained using centralized ML, FL, and FL with Sharemind MPC on the CIFAR10 (Fig.3a), MNIST (Fig.3b) and CASA (Fig.3c) datasets.

The results show that FL achieves comparable performance to centralized ML, while FL with Sharemind MPC achieves slightly lower performance. However, it must be noted that FL with Sharemind MPC provides better privacy protection than a standard FL solution. One possible explanation for the lower performance of FL with Sharemind MPC is that it uses fewer local epochs per communication round. Sharemind MPC is a more computationally expensive protocol than vanilla FL. However, it is possible to improve the performance of FL with Sharemind MPC by using more local epochs per communication round.

However, it can be seen that, for a particular dataset as Fig.3b suggests, the results are not only comparable but even slightly better than the other two. Moreover, Fig.3c demonstrates the exact behaviour of the global model as it performed on the CIFER10 dataset. Each of these figures shows the scores of centralized learning (200 epochs), the average scores of vanilla FL (FL-plain) and FL with Sharemind MPC (2 epochs locally per communication round and 200 rounds in total). The results show that FL performs similarly to the traditional centralized approach. However, the proposed algorithm achieves slightly lower performance, but it provides better privacy protection. This performance is due to FL with Sharemind MPC encrypting the model updates from the participants before they are shared with the server, which prevents the server from learning sensitive information about the participants' data.

The figure demonstrates that Federated Learning (FL) outperforms FL with Sharemind MPC on the MNIST dataset. This observation is likely due to the higher computational cost associated with Sharemind MPC compared to standard FL. Nonetheless, it is important to note that the difference in performance is relatively small, with FL using Sharemind MPC still achieving an impressive accuracy of 99.2

This process ensures that the server remains completely blind to the participants' data. In contrast, in vanilla FL, the server has access to unencrypted model updates from the participants, which could potentially allow it to glean sensitive information about the participants' data through analysis of these updates.

These promising results on the MNIST dataset are significant because MNIST is a widely recognized benchmark dataset for machine learning tasks. These findings suggest that FL with Sharemind MPC holds promise for training FL models across various real-world applications where data privacy is a paramount concern.

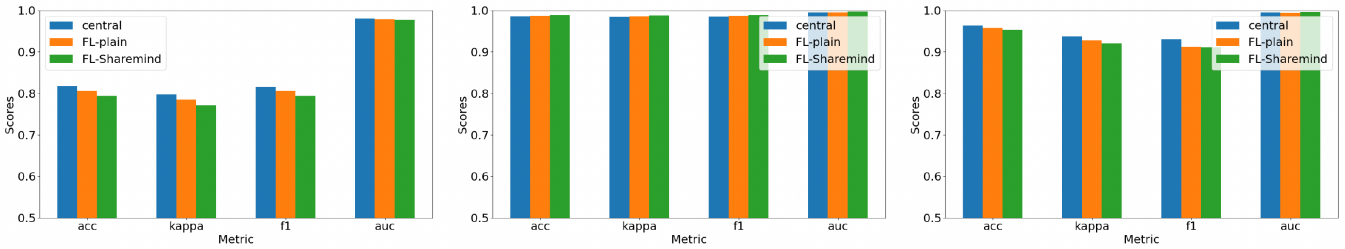
Additionally, it is important to highlight the AUC scores observed in both Fig.3a and Fig.3b for FL with Sharemind MPC. Notably, unlike other performance metrics, the AUC scores for Sharemind MPC are higher than those achieved by the other two methods.

However, it is essential to emphasize that, overall, even though the performance of FL with Sharemind MPC may lag slightly behind the other two methods, this difference is negligible, especially when considering the significant privacy advantages that secure aggregation offers.

It is worth acknowledging that the MPC environment inherently introduces a substantial computational overhead. Nevertheless, since Sharemind MPC exclusively handles aggregations and not the actual model training, the process remains efficient and operates within reasonable time frames, as demonstrated in Table1.

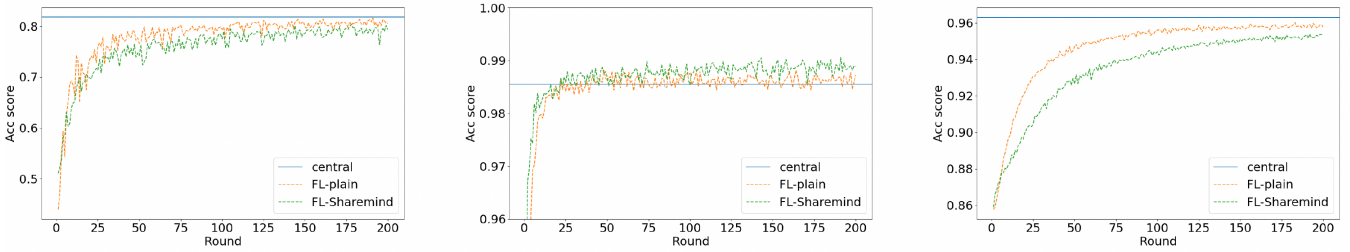
Furthermore, let us delve into the specific scores recorded in each round, as illustrated in Fig.4 to Fig.7. It is apparent from the results obtained for the CIFAR10 and CASA datasets that FL with Sharemind MPC exhibits a slower convergence compared to FL-plain, but it does not fall behind in terms of performance.

Notably, in Fig.4b and Fig.4c, the AUC scores gradually decrease with each round for both FL methods. However, this decline occurs slower in FL with Sharemind MPC. While a comprehensive



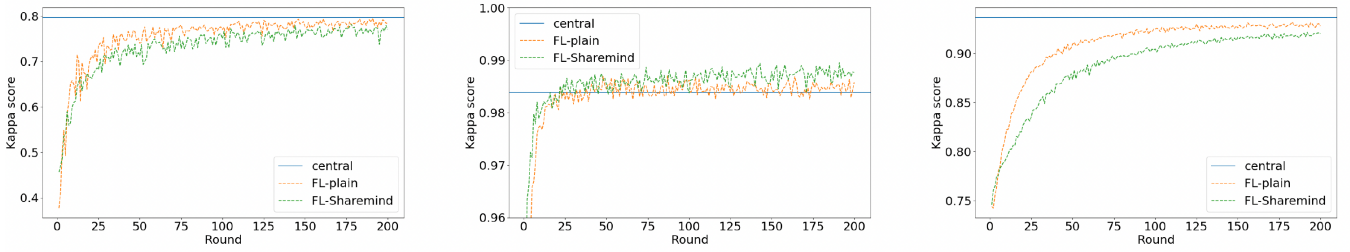
(a) Scores on CIFAR10. (b) Scores on MNIST. (c) Scores on CASA.

Figure 3: Summary of performances on MNIST, CIFAR10, and CASA datasets.



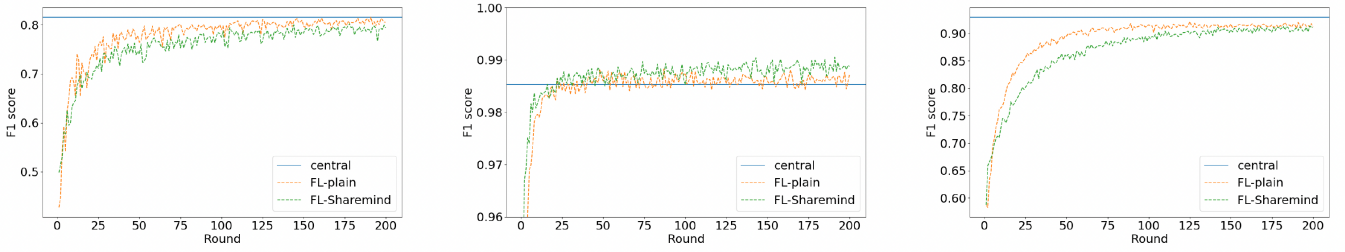
(a) Accuracy on CIFAR10. (b) Accuracy on MNIST. (c) Accuracy on CASA.

Figure 4: Accuracy scores on MNIST, CIFAR10, and CASA datasets.



(a) KAPPA on CIFAR10. (b) KAPPA on MNIST. (c) KAPPA on CASA.

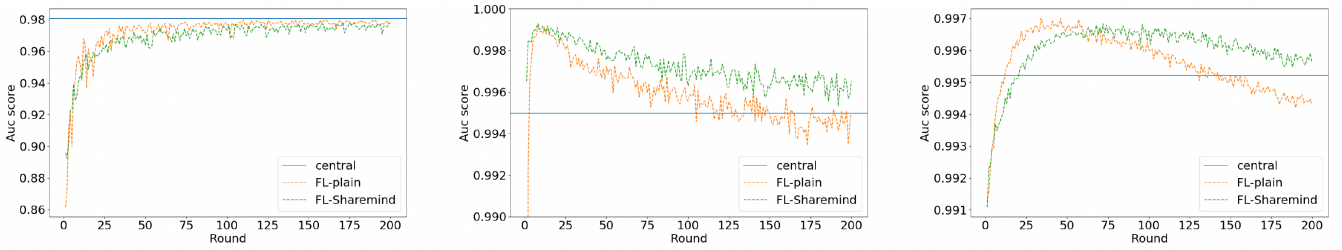
Figure 5: KAPPA scores on MNIST, CIFAR10, and CASA datasets.



(a) F1 measure of CIFAR10. (b) F1 measure of MNIST. (c) F1 measure of CASA.

Figure 6: F1 measure of MNIST, CIFAR10, and CASA datasets.

investigation into why only AUC scores decrease from round to round, unlike other metrics, remains a potential area for future research, we propose that overfitting might play a role. If this hypothesis holds, it is plausible that the inaccuracies introduced by



(a) AUC value of CIFAR10. (b) AUC value of MNIST. (c) AUC value of CASA.

Figure 7: AUC scores on MNIST, CIFAR10, and CASA datasets.

floating-point rounding errors, as mentioned earlier, contribute to preventing overfitting.

Particularly, Figures 6b to 6c highlight the accuracy of a federated learning (FL) model trained on the three datasets. A noteworthy achievement is that FL with Sharemind MPC can attain performance comparable to centralized ML on the MNIST dataset while offering even stronger privacy protection. This breakthrough opens up exciting possibilities for applying FL to a wider array of applications, especially those where data privacy is of paramount importance.

Another notable strength of this accomplishment is its validation on the MNIST benchmark dataset. This choice enhances the generalizability of the results to other datasets and scenarios.

As we look to the future, research efforts should concentrate on enhancing the performance of FL with Sharemind MPC, exploring strategies to make it even more efficient and scalable. Moreover, there is considerable potential for developing new and innovative applications for this privacy-preserving technology, expanding its utility across various domains and addressing a broader spectrum of real-world challenges.

7 DISCUSSION

Implementing an FL and MPC system for secure aggregation poses several challenges. In this section, we discuss these challenges and draft future work. We also discuss the opportunities behind such implementation and the attacks to be addressed by MPCFL.

Scalability concerns: As more clients (departments) participate, the system must be able to handle an increasing number of clients, model updates, and secure aggregations without significant slowdowns or resource constraints. this is particulary important in decentralized FL and IoT applications [3].

Communication overhead: The exchange of model updates or gradients, especially in the form of MPC shares, introduces significant communication overhead, especially when many clients are involved or when the models are large.

Computational complexity: MPC, while providing robust privacy guarantees, can be computationally intensive, particularly when dealing with complex models or when aggregating updates from numerous clients.

Trust issues among clients: Some clients might be skeptical about sharing even encrypted or transformed data, fearing potential leaks or misuse. Establishing trust and ensuring transparency in the system is crucial. Utilizing advanced authentication mechanisms might be inevitable, e.g., blockchain techniques [21].

System integration: Integrating such a system into existing departmental IT infrastructures without causing disruptions can be challenging. There might be compatibility issues, legacy systems to deal with, or even bureaucratic red tape.

Latency: Real-time or near-real-time fraud detection is preferable. However, the combined latency from FL and MPC operations might impact the timeliness of fraud alerts.

Regulatory and compliance issues: Implementing such a system might have legal implications. For instance, certain data might be protected by laws that prevent its use, even indirectly, outside its original context.

Addressing these challenges would require a combination of technological innovations, strategic planning, and collaboration across participating entities.

7.1 Challenges

Computational overhead: One of the predominant concerns when combining MPC with FL is the added computational overhead. MPC protocols inherently demand a series of secure multi-party interactions, ensuring data remains private during computations. The cumulative computational requirements could be substantial once integrated into an FL framework, where each participant locally trains models. This escalation in overhead not only implies potential delays in the learning process but also raises concerns about the feasibility of such a combination in applications with stringent real-time requirements.

Communication costs: Beyond computational complexities, communication overhead becomes a pivotal challenge. The secure nature of MPC often mandates the exchange of multiple encrypted messages among participants. In a federated environment, where nodes frequently communicate model updates, the sheer volume of this secure exchange can be magnified. The implication of this is twofold: firstly, the iterative process of FL could become considerably protracted, making timely convergence challenging. Secondly, the increased communication demands are prohibitively expensive or infeasible for nodes operating under bandwidth constraints.

Scalability: With every additional participant, the complexity of secure computations and aggregations in MPC rises, potentially exponentially. This escalation in complexity can render the combined system inefficient and unmanageable when scaled to large networks of participants. It underscores the need for scalable MPC protocols that gracefully handle the intricacies of federated environments.

Table 2: Description of potential attacks to be addressed by MPCFL

Attack	Goal	Difficulty	Impact	MPCFL Solution
DDoS	Overwhelm the system with traffic	Easy	Making it unavailable to legitimate clients	MPCFL is a distributed aggregation system that can be scaled to multiple parties.
Model poisoning	Modify updates maliciously	Easy	Causing unpleasant behaviors in the global model	MPCFL provides end-to-end security. In addition, the access control of Sharemind MPC prevents unauthorised users.
Model inference	Learn the distribution of the training data	Medium	Limited information about the training data	Secret sharing prevents the attacker from learning anything about the individual updates.
Gradient leakage	Learn the gradients of the model parameters regarding the input data	Medium	Information about the model parameters, which can be used to improve the performance of an attack	Secure aggregation aggregates the clients' updates without revealing anything neither to the aggregator nor to an attacker. This is done using secret sharing protocols that ensure the aggregate update is computed correctly, even though the individual updates are kept secret.
Model inversion	Reconstruct the actual training data	Difficult	Information about the training data, including the actual values of the features	Secret sharing and secure aggregation ensure that no single entity can reconstruct the model updates from individual parties.

7.2 Opportunities

Federated Learning with Sharemind MPC (FL with MPC) is a novel approach to machine learning that offers significant advantages over traditional FL aggregation methods in terms of privacy protection and robustness to malicious attacks. FL with MPC enables multiple participants to train a shared model without sharing their data. It is particularly attractive for applications where data privacy is critical, such as healthcare, finance, Code-Smell Detection [1], and manufacturing.

One of the critical opportunities of FL with MPC is its ability to facilitate training ML models on sensitive data without compromising participants' privacy. This mitigation of privacy is because FL with MPC can use encryption approaches or secret sharing to update the participants before they are aggregated at the servers, preventing the servers from learning any sensitive information about the participants' data with minimum impact on performance [5]. For example, FL with MPC could be used to train ML models to diagnose diseases, predict patient outcomes, and develop new treatments in the healthcare sector without compromising patients' privacy. Similarly, FL with MPC could be used to train ML models to detect fraud, predict stock market prices, and develop personalized financial advice in the finance sector without compromising customers' privacy.

In addition to its enhanced privacy protection capabilities, FL with MPC is more robust to malicious attacks than traditional FL aggregation methods. In traditional FL aggregation, a malicious server can exploit the aggregation process to learn about the participants' data [30]. This approach makes learning sensitive information much more difficult for a malicious server. This enhanced robustness to malicious attacks makes FL with MPC a particularly attractive choice for applications in sensitive domains such as healthcare and finance.

Overall, FL with MPC is a promising new approach to machine learning with a wide range of potential applications, especially in domains where data privacy is a critical concern. FL with MPC offers advantages over traditional FL aggregation methods regarding privacy protection and robustness to malicious attacks.

Combining MPC and FL offers an avenue for private and decentralized machine learning. Table 2 summarizes the potential attacks on an FL model at the aggregation phase. Addressing these challenges is instrumental in determining the future research trajectory in this domain and its real-world applicability.

7.3 Future Work

In future research endeavors, a comprehensive analysis will be orchestrated to delineate the nuances between secret sharing and homomorphic encryption within multi-party computation (MPC). This comparison is anticipated to shed light on these paramount encryption methodologies' distinctive attributes and potential synergies in securing data during FL aggregation. Furthermore, to cultivate a more panoramic view of the application spectrum, the investigation will be extended to incorporate additional datasets, namely imbalanced and non-IID. This inclusion aims to substantiate the robustness and versatility of the proposed model across varied data realms. Moreover, to hone the precision and efficiency of FL implementations, we envisage exploring alternative aggregation algorithms, transcending the confines of the currently utilized FedAvg. In tandem, a meticulous tuning of the algorithmic configurations

will be undertaken, encompassing parameters such as the number of epochs, local learning rounds, and communication rounds. This iterative tuning process aims to optimize the convergence rates and the overall performance of the FL framework, thereby propelling it towards a more adaptive and resilient paradigm for collaborative learning environments.

8 CONCLUSION

This research has highlighted the inherent vulnerabilities of the model aggregation phase in FL settings and provided a solution by introducing secure aggregation using secure multi-party computation. As we demonstrated, secure aggregation of DNNs in FL is feasible without sacrificing model accuracy. Both secure multi-party computation and federated learning stand as quintessential privacy-preserving techniques, adept at conducting data analysis without jeopardizing individual data confidentiality. Their integration promises robust solutions tailored for a broad spectrum of federated learning applications, thus marking a significant stride toward secure, privacy-centric data analysis and model development. We advocate using MPC and FL as privacy-by-design techniques for handling data processing without exposing the underlying data. In tandem, these methods can provide robust solutions for ML applications.

9 ACKNOWLEDGMENTS

This work was prepared and presented as a project in the MegaData: Federated Learning Summer University[2], University of Tartu, Tartu Estonia July 31 to August 11, 2023. This research was also supported by EU Horizon Framework grant agreement 101070186 (TEADAL).

REFERENCES

[1] Sadi Alawadi, Khalid Alkharabsheh, Fahed Alkhabbas, Victor Kebande, Feras M Awaysheh, and Fabio Palomba. 2023. FedCSD: A Federated Learning Based Approach for Code-Smell Detection. *arXiv preprint arXiv:2306.00038* (2023).

[2] Jan Philipp Albrecht. 2016. How the GDPR will change the world. *Eur. Data Prot. L. Rev.* 2 (2016), 287.

[3] Fahed Alkhabbas, Mohammed Alsadi, Sadi Alawadi, Feras M Awaysheh, Victor R Kebande, and Mahyar T Moghaddam. 2022. Assert: A blockchain-based architectural approach for engineering secure self-adaptive iot systems. *Sensors* 22, 18 (2022), 6842.

[4] David W. Archer, Dan Bogdanov, Y. Lindell, Liina Kamm, Kurt Nielsen, Jakob Illeborg Pagter, Nigel P. Smart, and Rebecca N. Wright. 2018. From Keys to Databases – Real-World Applications of Secure Multi-Party Computation. , 1749-1771 pages. http://dx.doi.org/10.1093/comjnl/bxy090

[5] Feras M Awaysheh. 2022. From the Cloud to the Edge Towards a Distributed and Light Weight Secure Big Data Pipelines for IoT Applications. In *Trust, Security and Privacy for Big Data*. CRC Press, 50–68.

[6] Feras M Awaysheh, Sadi Alawadi, and Sawsan AlZubi. 2022. FLIoDT: A Federated Learning Architecture from Privacy by Design to Privacy by Default over IoT. In *2022 Seventh International Conference on Fog and Mobile Edge Computing (FMEC)*. IEEE, 1–6.

[7] Feras M Awaysheh, Mamoun Alazab, Sahil Garg, Dusit Niyato, and Christos Verikoukis. 2021. Big data resource management & networks: Taxonomy, survey, and future directions. *IEEE Communications Surveys & Tutorials* 23, 4 (2021), 2098–2130.

[8] Feras M Awaysheh, Riccardo Tommasini, and Ahmed Awad. 2023. Big Data Analytics from the Rich Cloud to the Frugal Edge. In *2023 IEEE International Conference on Edge Computing and Communications (EDGE)*. IEEE, 319–329.

[9] Catherine Barrett. 2019. Are the EU GDPR and the California CCPA becoming the de facto global standards for data privacy and protection? *Scitech Lawyer* 15, 3 (2019), 24–29.

[10] Dan Bogdanov. 2013. *Sharemind: programmable secure computations with practical applications*. Ph. D. Dissertation. University of Tartu. http://hdl.handle.net/10062/29041

[11] Dan Bogdanov, Liina Kamm, Baldur Kubo, Reimo Rebane, Ville Sokk, and Riivo Talviste. 2016. Students and Taxes: a Privacy-Preserving Study Using Secure Computation. *PoPETs* 2016, 3 (2016), 117–135. http://www.degruyter.com/view/j/popets.2016.2016.issue-3/popets-2015-0019/popets-2016-0019.xml

[12] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical Secure Aggregation for Privacy-Preserving Machine Learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (Dallas, Texas, USA) *(CCS '17)*. Association for Computing Machinery, New York, NY, USA, 1175–1191. https://doi.org/10.1145/3133956.3133982

[13] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[14] Diane Cook, Aaron Crandall, and Brian Thomas. 2019. Human Activity Recognition from Continuous Ambient Sensor Data. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5D60P.

[15] R. Cramer, I. Damgård, and J. Nielsen. 2015. *Secure Multiparty Computation and Secret Sharing*. Cambridge University Press.

[16] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. 2015. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*. 1322–1333.

[17] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep learning*. MIT press.

[18] Marcella Hastings, Brett Hemenway, Daniel Noble, and Steve Zdancewic. 2019. SoK: General Purpose Compilers for Secure Multi-Party Computation. In *2019 IEEE Symposium on Security and Privacy (SP)*. 1220–1237. https://doi.org/10.1109/SP.2019.00028

[19] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. 2006. A fast learning algorithm for deep belief nets. *Neural computation* 18, 7 (2006), 1527–1554.

[20] Liina Kamm and Jan Willemson. 2015. Secure floating point arithmetic and private satellite collision analysis. *International Journal of Information Security* 14, 6 (2015), 531–548.

[21] Victor R Kebande, Feras M Awaysheh, Richard A Ikuesan, Sadi A Alawadi, and Mohammad Dahman Alshehri. 2021. A blockchain-based multi-factor authentication model for a cloud-enabled internet of vehicles. *Sensors* 21, 18 (2021), 6018.

[22] Alex Krizhevsky. 2009. Learning multiple layers of features from tiny images. https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

[23] Yann LeCun, Corinna Cortes, and CJ Burges. 2010. MNIST handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist* 2 (2010).

[24] Lingjuan Lyu, Han Yu, and Qiang Yang. 2020. Threats to federated learning: A survey. *arXiv preprint arXiv:2003.02133* (2020).

[25] Mohamad Mansouri, Melek Onen, Wafa Ben Jaballah, and Mauro Conti. 2023. Sok: Secure aggregation based on cryptographic schemes for federated learning. *Proc. Priv. Enhancing Technol* 1 (2023), 140–157.

[26] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*. PMLR, 1273–1282.

[27] Andre Ostrak, Jaak Randmets, Ville Sokk, Sven Laur, and Liina Kamm. 2021. Implementing Privacy-Preserving Genotype Analysis with Consideration for Population Stratification. *Cryptography* 5, 3 (2021). https://doi.org/10.3390/cryptography5030021

[28] Dario Pasquini, Danilo Francati, and Giuseppe Ateniese. 2022. Eluding secure aggregation in federated learning via model inconsistency. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. 2429–2443.

[29] Jaak Randmets. 2017. *Programming Languages for Secure Multi-party Computation Application Development*. Ph. D. Dissertation. University of Tartu. http://hdl.handle.net/10062/56298

[30] Mayank Rathee, Conghao Shen, Sameer Wagh, and Raluca Ada Popa. 2023. Elsa: Secure aggregation for federated learning with malicious actors. In *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1961–1979.

[31] Ahmed Roushdy Elkordy, Jiang Zhang, Yahya H Ezzeldin, Konstantinos Psounis, and Salman Avestimehr. 2022. How Much Privacy Does Federated Learning with Secure Aggregation Guarantee? *arXiv e-prints* (2022), arXiv–2208.

[32] Joshua C Zhao, Atul Sharma, Ahmed Roushdy Elkordy, Yahya H Ezzeldin, Salman Avestimehr, and Saurabh Bagchi. 2023. Secure aggregation in federated learning is not private: Leaking user data at large scale through model modification. *arXiv preprint arXiv:2303.12233* (2023).

[33] Ligeng Zhu, Zhijian Liu, and Song Han. 2019. Deep leakage from gradients. *Advances in neural information processing systems* 32 (2019).

[2] https://megadata.cs.ut.ee/