FLiPD: Privacy-Preserving Federated Learning via Multi-Party Computation and Differential Privacy

Gowri R Chandran[1*], Melek Önen[2] and Thomas Schneider[3]
[1] *Simula UiB, Norway*
[2] *EURECOM, France*
[3] *Technical University of Darmstadt, Germany*

Keywords:
Federated Learning, Multi-Party Computation, Differential Privacy

Abstract:
Federated Learning (FL) is a collaborative Machine Learning (ML) process where clients locally train an ML model on their private inputs, and send it to a server that aggregates these to obtain a global model update. FL is widely used in applications where the training data is distributed among several clients, e.g., for next word prediction in Gboard. Nevertheless, FL faces several privacy and security challenges. 1) Client privacy needs to be preserved by employing defenses against inference attacks using Secure Aggregation (SA) protocols. 2) The security of the model has to be defended against poisoning and backdoor attacks, e.g., by using clustering or filtering algorithms.
In this work, we present FLiPD, an optimised SA protocol for FL that protects against several attacks via a combination of Multi-Party Computation (MPC) and Differential Privacy (DP) mechanisms. We provide defenses against both inference and backdoor attacks. As opposed to existing solutions, in FLiPD, the client-server communication cost is essentially the same as in unprotected FL, which sends plaintext updates. Furthermore, the server-server communication cost is slightly lower (by 11%) than the state-of-the-art Prio+ (Addanki et al., SCN'22). In addition, we examine the accuracy of FLiPD both in the presence and absence of attacks. We achieve 87% accuracy for a Linear Regression model trained on the HAR dataset, and 90% for a Convolution Neural Network trained on the MNIST dataset.

1 Introduction

Federated Learning (FL) [Konečný et al., 2016] is a Machine Learning (ML) paradigm where clients collaboratively train a global model by sharing local model updates rather than raw data. This design ensures that sensitive data remains decentralized while still enabling effective model training. FL has gained traction due to privacy regulations such as HIPAA in the US [Act, 1996] and GDPR in the EU [Regulation, 2016], which restrict central data collection. FL is now widely applied in domains like text prediction in Gboard [Xu et al., 2023], medical research [Wen et al., 2019, Li et al., 2019], and autonomous driving [Chellapandi et al., 2024], making security and privacy in FL particularly critical.

While FL reduces the need to centralize sensitive data, it does not inherently preserve privacy. One major privacy concern are *inference attacks* [Shokri et al., 2017], where a corrupted server (known as the aggregator) can analyze clients' local model updates to infer sensitive information about their private data. Secure Aggregation (SA) [Castelluccia et al., 2005, Önen and Molva, 2007] is a commonly used defense mechanism against such attacks. In SA, the clients send *concealed* (using various cryptographic techniques) local updates so that the aggregator can compute the global model update without learning any individual contribution. The concealment can be achieved using methods such as Differential Privacy (DP) [Agarwal et al., 2018, Zhou et al., 2020, Seif et al., 2020, Triastcyn and Faltings, 2019, Yang et al., 2021], Homomorphic Encryption (HE) [Erkin and Tsudik,

[0]*This work was done while at Technical University of Darmstadt, Germany.

2012, Joye and Libert, 2013, Benhamouda et al., 2016], Multi-Party Computation (MPC) [Kadhe et al., 2020], or Trusted Execution Environments (TEE) [Zhao et al., 2021, Mo et al., 2021, Kato et al., 2023]. However, while SA protects the privacy of client updates, it does not provide comprehensive security: poisoning attacks [Shen et al., 2016], backdoor attacks [Bagdasaryan et al., 2020], and inference attacks [Gao et al., 2023] remain feasible even when SA is employed.

A major security threat in FL is *poisoning attacks* [Biggio et al., 2012], where malicious clients manipulate the training process to degrade the accuracy of the global model. Such attacks can be carried out either by corrupting the local training data (data poisoning) [Shen et al., 2016] or by directly submitting maliciously crafted updates to the server (model poisoning) [Bagdasaryan et al., 2020]. A common line of defense against these attacks is to apply robust aggregation techniques, such as filtering or clustering, which aim to identify and exclude anomalous client updates that deviate significantly from the majority. These defenses rely on various distance metrics, including Hamming Distance (HD) [Dong et al., 2021, Qiu et al., 2024], cosine similarity [Nguyen et al., 2022, Xu et al., 2024], and K-means clustering [Shen et al., 2016], to distinguish between benign and malicious contributions. However, such techniques often incur additional computational costs and may fail against adaptive adversaries.

A few recent studies have explored combining defenses against both inference and poisoning attacks [Dong et al., 2021, Nguyen et al., 2022, Xu et al., 2024]. However, these approaches often introduce significant computation or communication overhead [Nguyen et al., 2022, Xu et al., 2024], which is especially critical in practical FL deployments where clients usually have limited computation and storage capacities and often use WAN or LTE networks for communication. Other defenses rely on strong assumptions, such as the server having a base model [Dong et al., 2021], or provide limited robustness under collusion. As a result, FL remains vulnerable in practical deployments. Taken together, these attacks demonstrate that existing defenses, whether isolated or combined, remain incomplete, leaving FL systems vulnerable when adversaries exploit multiple weaknesses simultaneously.

To address this gap, in this paper, we propose FLiPD, a holistic framework for FL that simultaneously defends against inference and poisoning attacks while minimizing client communication.

Our Contributions

Taking into account the requirements for deploying Federated Learning (FL) in real-world scenarios and the limitations of existing solutions, we propose FLiPD, a communication-efficient Secure Aggregation (SA) protocol that provides holistic defenses against multiple attacks while being resilient to client-server collusion. Our contributions are summarized as follows:

1. **Communication-Efficient Secure Aggregation:** We design FLiPD as an SA protocol fully executed using MPC, reducing the trust required on the aggregator and supporting client dropouts and failures. To minimize client-server communication, we adopt a seed-based secret sharing scheme [Demmler et al., 2014, Suresh et al., 2017, Chandran et al., 2023, Ben-Itzhak et al., 2024], which preserves client communication at the same message size as unprotected FL, achieving full communication efficiency, for the first time in private FL.
2. **Holistic Defense Against Attacks:**
   (a) **Poisoning Attack Mitigation:** We employ Hamming Distance (HD)-based filtering to detect anomalous client updates. HD is computationally inexpensive in the MPC setting and has been shown to perform comparably to more costly measures, such as cosine similarity [Dong et al., 2021].
   (b) **Inference Attack Mitigation:** FLiPD integrates DP noise at the end of aggregation. Unlike FLAME [Nguyen et al., 2022], our DP mechanism is instantiated entirely within MPC, while incurring minimal additional cost. Our distributed DP ensures privacy even under collusion between clients and the server, as long as one client and one server remain honest.

We instantiate our MPC protocol using ABY2.0 [Patra et al., 2021], enabling efficient mixed-protocol conversions and low inter-server communication. Moreover, ABY2.0 optimises the online communication cost, shifting most communication to a precomputation phase.

In summary, FLiPD provides a holistic, efficient, and practical solution for privacy-preserving FL. It reduces communication and computation costs while simultaneously defending against multiple attacks and maintaining accuracy comparable to prior state-of-the-art approaches.

2 System Model

The FLiPD protocol involves two types of entities: clients and servers. Let $C = \{C_1, \cdots, C_{\mathsf{N}}\}$ be the set of clients, each holding input x_i. The clients never interact with each other, only with the servers. We consider two servers S^1 and S^2, acting as the aggregators, collecting secret-shared model updates from the clients to securely compute the global model. The clients are considered to be Augmented Semi-Honest (ASH) [Goldreich, 2004], meaning that they behave semi-honestly while having the ability to manipulate their inputs to the protocol. The servers are considered to be semi-honest, but can collude with a subset of the clients. The two servers are assumed not to collude with each other. We detail the capabilities of the adversary below.

1. ***Data Poisoning.*** The adversary $\mathcal{A}$ may corrupt a subset of clients to inject poisoned data, causing their local updates to deviate from honest behavior, thus biasing the global model update in their favor. This capability is formalised under the ASH adversary model. To mitigate such attacks, FLiPD employs Hamming Distance (HD)-based filtering and elimination of poisoned updates.
2. ***Membership Inference.*** The adversary $\mathcal{A}$ may attempt to learn information about an honest client's input in two ways: (i) by leveraging the local updates sent by the client to the server, and (ii) by monitoring the client's presence or absence in different iterations and comparing the model updates in those iterations. To prevent the first type of inference attack, we employ an MPC-based Secure Aggregation (SA) algorithm, and to defend against the second, we integrate Differential Privacy (DP).
3. ***Collusion.*** The adversary $\mathcal{A}$ may simultaneously corrupt a subset of clients and one server, effectively creating a collusion between the server and the corrupted clients. To maintain the security of FLiPD, at least one server and one client must remain honest at all times. Consequently, the two servers are assumed to be non-colluding, a standard assumption in MPC. Typically, servers are operated by independent entities that collaborate to achieve a common goal but do not collude due to legal, contractual, or reputational constraints.
4. ***Client Dropouts.*** Any client may drop out of the protocol at any point during execution. FLiPD incorporates a verification step between the two servers to ensure that both shares from all participating clients are received before aggregation.

Essentially, FLiPD ensures the integrity, privacy, and correctness of the global model against data poisoning, inference attacks, collusion, and client dropouts, under the assumption that at least one server and one client remain honest.

3 Related Works

We now review existing approaches for secure and robust FL, highlighting their strengths and limitations relative to FLiPD.

3.1 Multi-Party Computation-based FL

Multi-Party Computation (MPC) enables n parties to jointly compute a function on their inputs without revealing them. In Federated Learning, MPC-based Secure Aggregation protocols [Corrigan-Gibbs and Boneh, 2017, Kadhe et al., 2020, Nguyen et al., 2022, Addanki et al., 2022, Rathee et al., 2023, Gehlhar et al., 2023] rely on 2 or more non-colluding servers (or aggregators) to securely combine local model updates. While these models preserve privacy, they incur significant communication overhead between the clients and server.

To address this limitation, several works focus on reducing client-side communication. Prio+ [Addanki et al., 2022] improves upon Prio [Corrigan-Gibbs and Boneh, 2017] by using Boolean secret sharing to reduce the client communication, while [Ben-Itzhak et al., 2024] applies quantization techniques to reduce the size of the transmitted data. In our work, we adopt a seed expansion-based secret sharing scheme [Demmler et al., 2014, Suresh et al., 2017, Chandran et al., 2023, Ben-Itzhak et al., 2024], enabling clients to send essentially the same amount of data as in plaintext.

Furthermore, most of the MPC-based FL protocols assume semi-honest servers, while recent works [Marx et al., 2023, Rathee et al., 2023] consider malicious servers. Additionally, some works [Corrigan-Gibbs and Boneh, 2017, Addanki et al., 2022, Rathee et al., 2023], incorporate input validation by clients to mitigate data poisoning. Assuming semi-honest servers is a practical

and widely adopted trade-off in FL. In many deployments, servers are operated by reputable organizations or independent entities with a vested interest in correctly executing the protocol, but they may still attempt to learn additional information from the data. Following this rationale, we consider semi-honest servers in our protocol, but similar to [Rathee et al., 2023], they may collude with the clients.

3.2 Poisoning and Backdoor Defenses in FL

Since training data remains decentralized at the clients, FL models are inherently susceptible to poisoning and backdoor attacks [Shen et al., 2016, Bagdasaryan et al., 2020]. Numerous defenses have been proposed against such attacks, with the core idea being to detect and eliminate anomalous updates before further computation.

One line of work uses clustering-based defenses [Shen et al., 2016, Blanchard et al., 2017, Nguyen et al., 2022], where clients are grouped using clustering algorithms and suspicious clusters (e.g., the smallest cluster) are discarded. Other approaches rely on distance-based similarity metrics: for example, [Fung et al., 2018] employs cosine similarity to detect updates with large angular deviation. Whereas [Dong et al., 2021] demonstrates that HD is particularly well-suited for identifying anomalies in binary representations, using HD similarity to find updates farthest from a base global model. Another defense strategy is input verification [Corrigan-Gibbs and Boneh, 2017, Khazbak et al., 2020, Rathee et al., 2023], where clients provide proofs (e.g., range proof) that their provided updates lie within an acceptable domain.

In FLiPD, we employ HD-based filtering, but unlike [Dong et al., 2021] we do not rely on a base global model. Instead, we compute pairwise HDs among all client updates and identify inputs with abnormally large aggregate distances, which are then eliminated. This design yields a defense that is both computationally efficient in the MPC setting and robust against collusion, aligning with our threat model.

3.3 Inference Attack Defenses in FL

Revealing global model updates makes FL vulnerable to inference attacks, where adversaries infer private client data from changes in the updates [Shokri et al., 2017, So et al., 2023, Gao et al., 2023]. While Secure Aggregation (SA) hides individual updates, it does not prevent leakage from the aggregated result.

A common defense against these attacks is Differential Privacy (DP) [Dwork et al., 2006]. Several works [Zhou et al., 2020, Seif et al., 2020, Stevens et al., 2022, Nguyen et al., 2022] integrate DP into FL by injecting noise into updates. Noise can either be added locally by the clients [Seif et al., 2020, Yang et al., 2021], or globally by the server [Stevens et al., 2022, Nguyen et al., 2022]. Local DP provides strong privacy guarantees but typically requires larger noise magnitudes, leading to reduced Main Task Accuracy (MA). Global noise, on the other hand, achieves better accuracy but relies on the assumption of a trusted server, which is often impractical.

To reduce this trust, recent research has explored Distributed DP, where multiple untrusted parties jointly generate and apply noise [Truex et al., 2019, Kairouz et al., 2021, Stevens et al., 2022]. These protocols eliminate the need for a single trusted server, while maintaining accuracy closer to global DP. Following this line of work, FLiPD employs a distributed setup where two servers independently generate and add noise, which is then combined into the final global update. This design strengthens the robustness against inference attacks while avoiding the strong trust assumptions of prior global DP-based solutions.

4 FLiPD: Lightweight FL Protocol

We now present the construction of our SA protocol. Firstly, we define the ideal functionality F_{SA}, which captures secure aggregation with integrated defenses against poisoning and inference attacks. Our protocol Π_{SA} is then designed to securely realize F_{SA} in the presence of an ASH adversary. The functionality F_{SA} proceeds as follows: 1) N clients C_i having input x_i, for $i \in [1, \mathsf{N}]$, send their inputs to the functionality, 2) perform Hamming Distance (HD)-based filtering (F_{sel}), 3) the filtered inputs are aggregated (F_{agg}), 4) DP noise is added to the aggregate, 5) the functionality returns the noisy aggregated model to the clients, and 6) the clients reconstruct the shares to obtain the global model update.

Now, we present the FLiPD protocol, which securely realizes the ideal functionality F_{SA}.

FLiPD combines MPC-based secure aggregation, Hamming Distance (HD) filtering for robustness against poisoning, and distributed Differential Privacy (DP) noise generation for protection against inference attacks, all while maintaining communication costs comparable to unprotected FL. We consider N clients C_i and two non-colluding servers S^1 and S^2.

1. ***Input sharing.*** Each client C_i, for $i \in [1, \mathsf{N}]$, updates the model locally. Then, the clients use the seed-based technique similar to [Demmler et al., 2014, Suresh et al., 2017, Chandran et al., 2023, Ben-Itzhak et al., 2024] for compact input sharing: Each C_i randomly chooses a seed $\mathsf{sd}_i \leftarrow \{0,1\}^\lambda$, where λ is the computational security parameter set to 128 bits. Each C_i then generates two shares of their input: $s_i^1 = \mathsf{sd}_i$[1], and $s_i^2 = x_i \oplus \mathrm{PRG}(\mathsf{sd}_i)$, where PRG is a pseudorandom generator. Then they send share s_i^k to server S^k, for $k \in \{1,2\}$.
2. ***Filtering.*** Once the servers receive the shares from the clients, they proceed to securely filter the inputs. FLiPD uses a Hamming Distance (HD)-based filtering to eliminate inputs that are potentially malicious or poisoned. The servers compute the Total Hamming Distance (THD) for each client by computing the sum of all pair-wise HDs for each input. The servers then filter out the inputs whose thd lie outside the range $[\mu - 2\sigma, \mu + 2\sigma]$, where μ is the mean and σ is the standard deviation of the thds. At the end of the computation, each server receives a secret share of a bit string sel that indicates which input is accepted and a secret share of a count ct, which indicates how many inputs were accepted. The entire filtering process is oblivious, i.e., the servers never learn which or how many inputs are accepted. In § 5.1, we further analyze the rationale behind choosing HD-based filtering.
3. ***Aggregation.*** Once the servers receive sel and ct, they proceed to compute the aggregate of the remaining inputs after the filtering. At the end of the computation, each server receives a secret share of the aggregate agg.
4. ***DP Noising.*** In this phase, the servers add DP noise to the aggregated value from Step 3. The servers generate the noise distributedly, which is then composed to form the total noise (cf. § 5.2 for details on the noise distribution). The server S^1 locally samples noise ns_1 from the Laplace distribution with density (GS_f/ϵ'), where $\epsilon' = \epsilon/2$, ϵ is a publicly known privacy parameter, and GS_f is the global sensitivity (cf. § 5.2 for the sensitivity analysis). Then, S^1 secret shares the noise to obtain ns_1^1 and ns_1^2 and sends ns_1^2 to S^2. Following similar steps, the server S^2 generates noise ns_2 and sends ns_2^1 to S^1. Each server then adds these shares to the shares of the aggregate to get $\mathsf{agg}'^k = \mathsf{agg}^k + \mathsf{ns}_1^k + \mathsf{ns}_2^k$.
5. ***Resharing.*** To reduce the communication cost of sending both shares to the clients, the servers perform a resharing step, similar to the input sharing in Step 1: The server S^1 randomly chooses a seed $\mathsf{sd} \leftarrow \{0,1\}^\lambda$, and computes $\Delta = \mathsf{agg}'^1 \oplus \mathrm{PRG}(\mathsf{sd})$. S^1 sends Δ to S^2, and sets its final share $\mathsf{SA}^1 = \mathsf{sd}$. Server S^2 sets its final share to $\mathsf{SA}^2 = \Delta \oplus \mathsf{agg}^2$.
6. ***Sending aggregated output.*** The servers send their respective shares SA^k to all clients. The clients reconstruct the aggregated global update $\mathsf{SA} = \mathrm{PRG}(\mathsf{SA}^1) \oplus \mathsf{SA}^2$.

To extend the protocol to accommodate $\mathsf{N_S} > 2$ servers, each client C_i simply needs to select $(\mathsf{N_S} - 1)$ seeds sd_i^k, and set $s_i^k = \mathsf{sd}_i^k$ for $k \in [1, \mathsf{N_S} - 1]$ and $s_i^{\mathsf{N_S}} = x_i \oplus \mathrm{PRG}(\mathsf{sd}_i^1) \oplus \cdots \oplus \mathrm{PRG}(\mathsf{sd}_i^{\mathsf{N_S}-1})$. The share s_i^k is then sent to server S^k, for $k \in [1, \mathsf{N_S}]$. The servers continue the remaining computation using any generic MPC protocol for more than 2 parties, e.g. GMW [Goldreich et al., 1987].

Due to space constraints, we defer the formal correctness and security proofs to the full version[2] of this paper.

5 Analysis of Key Components of FLiPD

In this section, we elaborate on two of the core building blocks used to design FLiPD. Specifically, we detail the use of Hamming Distance (HD) as a lightweight yet effective filtering mechanism against poisoning attacks, and Differential Privacy (DP) as a rigorous defense against inference attacks. These components form the foundation of our protocol's robustness and privacy guarantees.

[1]The seed can be used for multiple iterations, therefore its communication amortizes.

[2]https://eprint.iacr.org/2026/324

5.1 Hamming Distance

In our model, malicious clients may submit poisoned inputs to bias the global model update (cf. § 2). To mitigate such attacks, FLiPD incorporates *oblivious* filtering of client updates, excluding anomalous inputs from further computation. Following [Dong et al., 2021], we employ HD as the distance measure for detecting poisoned updates. HD is particularly well-suited for our setting for two reasons. First, HD can be computed very efficiently in MPC: it requires only XOR operations (which are free in the Boolean domain) and additions (which are free in the arithmetic domain), with the only overhead being the conversion between Boolean and arithmetic shares. Second, prior work [Dong et al., 2021] demonstrates that HD achieves accuracy and robustness comparable to defenses such as Krum [Blanchard et al., 2017], Median [Yin et al., 2018], T-Mean [Yin et al., 2018], and FLAME [Nguyen et al., 2022] in the honest majority setting, and significantly outperforms them in the dishonest majority setting.

In [Dong et al., 2021], the HD of a client's input is calculated with respect to a base model maintained by the server. To eliminate this additional assumption, we introduce the Total Hamming Distance (THD). The THD of a client's input is defined as the sum of its HDs with all other client inputs. By analysing the distribution of the THDs, we identify the inputs that deviate the most from the rest, indicating potential maliciousness. Outliers are determined by selecting inputs whose THD falls outside the range $[\mu - 2\sigma,\ \mu + 2\sigma]$, where μ is the mean and σ is the standard deviation of all THDs. By the properties of the standard distribution, this range captures approximately 95.45% of the inputs, which are then retained for further computation.

5.2 Differential Privacy

In addition to poisoning attacks, FL is vulnerable to inference attacks (cf. § 2), where a malicious client attempts to extract information about other clients' input from the global model update. To address this, FLiPD uses a DP mechanism. By injecting DP noise into the aggregated global update, the correlation of the global update to any individual client input is obfuscated, rendering inference attacks ineffective while preserving the utility of the model.

To preserve the decentralized nature of FL and ensure balanced influence between the servers, we generate the DP noise such that both servers contribute equally. Existing protocols [Nguyen et al., 2022] generate DP noise entirely within an MPC circuit, but this approach incurs high computational and communication overhead. Instead, FLiPD employs a distributed noise generation approach: each server independently computes a portion of the DP noise, which is then combined to form the final global noise. The noise is sampled from a pre-defined distribution; specifically, FLiPD uses the Laplace Mechanism to ensure differential privacy of the aggregated model, but this can be easily replaced by any other DP mechanism. The standard deviation of the Laplace distribution can be calibrated according to the global sensitivity GS_f of the aggregation function to achieve the desired privacy level ϵ. We now present the detailed instantiation of this distributed DP mechanism.

Noise distribution Let ϵ be the privacy budget for one execution of the protocol. To preserve symmetry, we divide the budget equally among the two servers, such that each has a privacy budget of $\epsilon' = \epsilon/2$. Then, the servers sample their noise from the Laplace distribution, $\text{Lap}(GS_f/\epsilon') = \text{Lap}(2GS_f/\epsilon)$, where GS_f is the global sensitivity of the ML model. Therefore, there are two mechanisms $\mathcal{M}_1, \mathcal{M}_2$ run by servers S_1 and S_2, respectively, that achieve $(\epsilon/2)$-differential privacy. And, by the DP composition theorem [Vadhan, 2017], the overall SA protocol achieves ϵ-differential privacy.

Sensitivity Analysis The model's sensitivity determines the quantity of noise added to the aggregated model. For the Laplace mechanism, the sensitivity is computed as the maximum ℓ_1 distance between two inputs. In the presence of a data poisoning attack, the sensitivity may also increase significantly as the input can vary arbitrarily. However, since we perform HD-based filtering, the inputs with large distances from other inputs are eliminated. Based on this, we can calculate a bound for the maximum ℓ_1 distance between two inputs. For binary inputs, the ℓ_1 distance is the same as the Hamming distance. Therefore, we get the maximum ℓ_1 distance between two inputs to be 4σ, where σ is the standard deviation of the distribution of the THD. Hence, we set the global sensitivity of the model to $GS_f = 4\sigma$.

Secure computation From the noise distribution and sensitivity analysis, we see that $GS_f = 4\sigma$ and the privacy parameter for one server is ϵ'. During the secure computation of the DP noise, each server independently samples noise from a Laplace distribution $Lap(4\sigma/\epsilon')$. Then, each server secret shares the noise and sends one share to the other server. Therefore, each server can now compute the share of the total noise, which is then added to the aggregate.

Collusion between server and clients Our distributed noise generation preserves privacy even in the case of collusion between one server and $\mathsf{N}-1$ clients. The details of privacy in this case are discussed in the full version[3].

6 Evaluation

To validate our design choices and theoretical analyses, we conduct a series of experiments evaluating both the security and efficiency of our FLiPD protocol. Our evaluation focuses on two key aspects:

1. the communication efficiency of FLiPD, with particular emphasis on reducing client–server communication overhead, and
2. the ability of FLiPD to defend against attacks while maintaining high task accuracy.

To assess efficiency, we instantiate our MPC protocol using ABY2.0 [Patra et al., 2021] and measure the resulting server-server communication costs (§ 6.1). We evaluate the accuracy of FLiPD using the SAFEFL framework [Gehlhar et al., 2023] under different attack scenarios (§ 6.2). We also compare the performance of FLiPD with prior works in terms of both communication efficiency and accuracy.

6.1 Communication Costs

We assume that each client holds a vector of m weights of 32-bits each. Secure aggregation is then performed among two servers using the ABY2.0 protocols [Patra et al., 2021].

Client-Server Communication The only communication between the clients and the servers occurs when clients transmit their local model updates. Using the optimized secret sharing scheme described in § 4, each client sends $32|m| + \kappa$ bits, which is almost identical in size to plaintext FL with $32|m|$ bits. For a model with $m = 100k$ parameters, this corresponds to approximately 0.4 MB per client, while for $m = 500k$ parameters, it increases to about 2 MB. Compared to other MPC-based SA protocols (cf. Tab. 1), FLiPD achieves the best communication efficiency on the client side.

Table 1: Comparison of client communication costs (in MB per client) of FLiPD with MPC-based private FL protocols FLOD [Dong et al., 2021], Prio [Corrigan-Gibbs and Boneh, 2017], and Elsa [Rathee et al., 2023]. Best results are marked in bold.

# Params	FLOD	Prio	Elsa	FLiPD
$100k$	0.80	59.10	51.60	**0.40**
$500k$	4.00	262.20	258.00	**2.00**

Server-Server Communication Once the servers receive the shares from the clients, server S^1 expands its shares using the PRG. The most communication-intensive phase of FLiPD is the filtering step, which requires pairwise HD computation, standard deviation computation, and comparison to identify outliers. Among these, the pairwise HD computation dominates the cost, as it requires Bit-to-Arithmetic (Bit2A) conversions [Patra et al., 2021], by far the most expensive operation in this phase. For N clients, the protocol requires $\mathsf{N}(\mathsf{N}-1)l$ Bit2A conversions, where l is the bit length of a single parameter. Consequently, the communication cost of the filtering phase increases quadratically, i.e., $O(\mathsf{N}^2)$. In contrast, all other phases of our protocol scale at most linearly with N.

Table 2: Inter-server communication costs (in GB) for the different phases of our protocol for $\mathsf{N} = 100$ clients and $\mathsf{m} = 100k$ parameters.

Phase	**Offline**	**Online**	**Total**
Filtering	4.26	2.54	6.80
Aggregation	0.19	0.00	0.19
DP Noising	—	0.00	0.00
Resharing	0.00	0.00	0.00
Total	4.45	2.55	7.00

Using the mixed protocol conversion of ABY2.0 [Patra et al., 2021], we compute the server-server communication costs for a single ag-

[3] https://eprint.iacr.org/2026/324

gregation round. In Tab. 2, we provide the communication cost for each phase of our protocol. Most of the communication in FLiPD occurs in the offline preprocessing phase (about 63.5% of the total communication), making the online phase very efficient.

Table 3: Comparison of inter-server communication costs (in GB) of FLiPD with MPC-based private FL protocols FLOD [Dong et al., 2021], Prio+ [Addanki et al., 2022], and Elsa [Rathee et al., 2023].

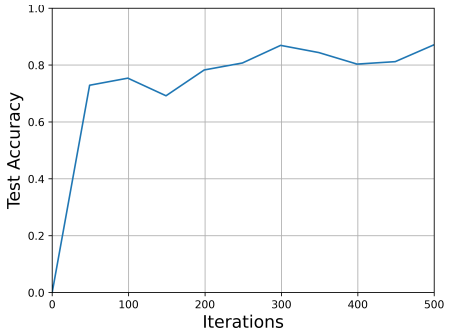
N	m	FLOD	Prio+	Elsa	FLiPD
50	$100k$	2.37	3.88	4.50	3.46
100	$100k$	4.54	7.75	9.00	7.00
50	$500k$	11.86	19.38	22.50	17.30
100	$500k$	22.68	38.75	45.00	34.99

For 100 clients with $500k$ parameters each, the server-server communication is 34.99 GB. Favorably, about 63.5% of the total communication occurs in the offline preproccessing phase, making the online phase very efficient. The comparison of communication costs with other MPC-based FL algorithms [Corrigan-Gibbs and Boneh, 2017, Addanki et al., 2022, Rathee et al., 2023] is discussed in the full version.

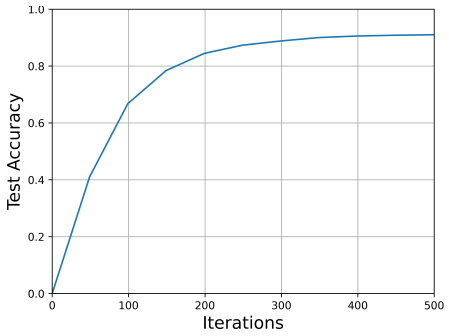
Tab. 3 reports the inter-server communication costs for various MPC-based SA protocols. From the results, FLOD [Dong et al., 2021] achieves the lowest inter-server communication, while FLiPD incurs approximately 1.5× higher communication than that. However, FLiPD provides defense against inference attacks, which is not the case for FLOD. Moreover, the increase in communication is acceptable in practice, as the client–server communication, the main bottleneck in FL deployments, remains minimal and comparable to plaintext FL.

In Tab. 2, we provide the communication cost per server per client for 100 clients with $100k$ parameters. Most of our communication occurs in the offline preprocessing phase (about 63.5% of the total communication), with a very efficient online phase.

A closely related work, ScionFL [Ben-Itzhak et al., 2024], proposes an FL protocol that has the same client communication as our work, in addition to 8× lower inter-server communication. However, they assume the presence of 3 MPC servers with an honest majority, whereas we operate in the stronger 2 server setting.



((a)) HAR dataset



((b)) MNIST dataset

Figure 1: Accuracy of FLiPD on HAR and MNIST datasets, without any attacks.

6.2 Accuracy Evaluation

To evaluate the accuracy of our aggregation under different attack scenarios, we implement the FLiPD plaintext algorithm within the SAFEFL framework [Gehlhar et al., 2023]. We consider two model-dataset pairs: 1) Linear Regression (LR) classifier trained on the Human Activity Recognition (HAR) dataset [Anguita et al., 2013], and 2) Convolution Neural Network (CNN) trained on the MNIST dataset [LeCun et al., 1998]. While SAFEFL already included training for the HAR dataset, we extended[4] it to incorporate MNIST with the CNN model.

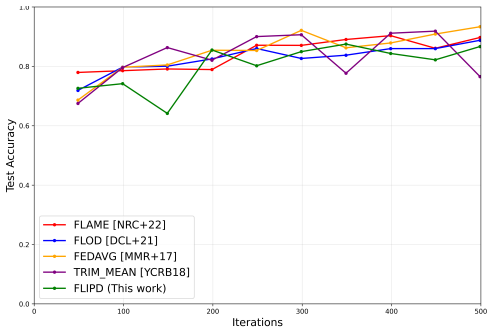
First, we evaluate the accuracy of FLiPD in the absence of any attacks, and compare it against other state-of-the-art SA protocols. Fig. 1 shows the performance of FLiPD on both datasets, and Fig. 2 shows that its accuracy is comparable to existing protocols. Maintaining high accuracy under no-attack conditions demonstrates that the protocol's security mechanisms do not compromise the utility of the aggregated model.

Next, we evaluate the accuracy of FLiPD under backdoor attacks. We consider four attack

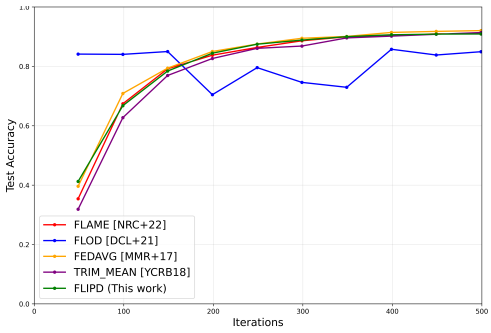
[4]https://github.com/encryptogroup/FLiPD.

Table 4: Accuracy result of various secure aggregation protocols (FedAvg [McMahan et al., 2017], FLTrust [Cao et al., 2021], TrimMean [Yin et al., 2018], FLAME [Nguyen et al., 2022], and FLOD [Dong et al., 2021]) against different backdoor attacks, for both LR model trained on HAR dataset and CNN model trained on MNIST with 20% corruption rate.

Attack	Dataset	**FedAvg**	**FLTrust**	**TrimMean**	**FLAME**	**FLOD**	**FLiPD**
No	HAR	**93.3**	89.7	76.6	89.7	88.8	86.7
	MNIST	**92.0**	91.7	91.4	90.9	84.9	90.9
Krum	HAR	**91.0**	87.4	74.3	80.1	86.5	83.8
	MNIST	89.2	**89.8**	86.4	87.9	73.6	85.5
Label Flipping	HAR	63.3	**90.5**	86.9	86.9	80.8	66.5
	MNIST	77.2	**84.4**	75.2	83.6	55.0	68.8
Trim	HAR	84.2	84.7	88.2	88.2	**89.1**	77.1
	MNIST	75.4	**89.5**	67.6	87.7	16.1	76.9
Scaling	HAR	**93.7**	89.8	92.0	90.1	86.5	88.3
	MNIST	**90.7**	90.1	85.8	88.1	26.1	89.6

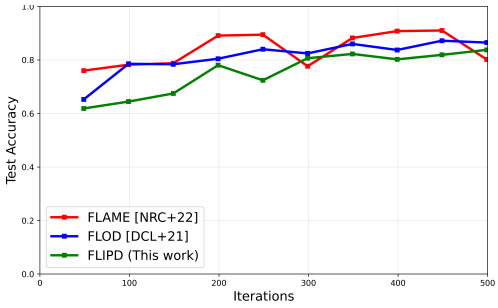


((a)) HAR dataset

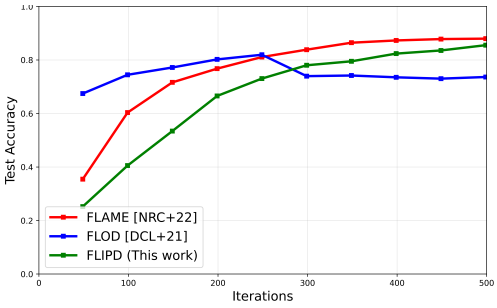


((b)) MNIST dataset

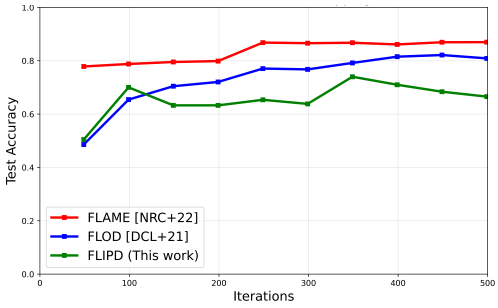
Figure 2: Accuracy comparison of different Secure Aggregation protocols without any attack.



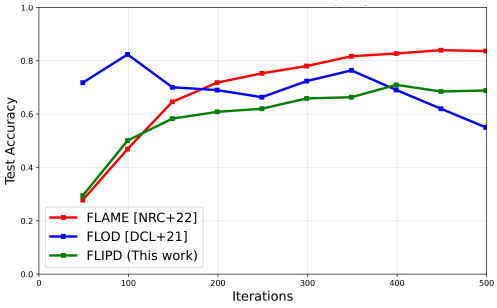
((a)) Krum attack, HAR dataset



((b)) Krum attack, MNIST dataset



((c)) Label flipping attack, HAR dataset



((d)) Label flipping attack, MNIST dataset

Figure 3: Accuracy comparison of FLAME [Nguyen et al., 2022], FLOD [Dong et al., 2021], and FLiPD protocols, under Krum and label flipping attacks on HAR and MNIST datasets.

types: label flipping, Krum, Trim, and Scaling [Fang et al., 2020], and compare the results against several aggregation rules, including FLAME [Nguyen et al., 2022], FLOD [Dong et al., 2021], FedAvg [McMahan et al., 2017], FLTrust [Cao et al., 2021], and Trim Mean [Yin et al., 2018]. The results, summarized in Tab. 4, show that FLiPD achieves reasonable accuracy compared to existing defenses. Among the attacks, the untargeted label flipping attack proves to be the most challenging for our protocol.

Because FLAME and FLOD share the closest design characteristics with FLiPD, we provide a more detailed comparison with them. In terms of accuracy, FLAME achieves the strongest performance overall, while FLiPD delivers comparable results on both HAR and MNIST. FLOD, in contrast, performs well on HAR but shows instability on MNIST. We attribute this fluctuation to the use of the sgn encoding in FLOD, which can negatively affect robustness across datasets. Fig. 3 illustrates the accuracy comparison of FLAME, FLOD and FLiPD under various attack settings.

7 Conclusion

In this paper, we proposed FLiPD, an MPC-based Secure Aggregation protocol that addresses key challenges in practical Federated Learning deployments. First, FLiPD provides robust defenses against multiple attacks in FL while maintaining high task accuracy. Second, it optimizes client–server communication, achieving the same communication cost as in unprotected, plaintext FL. Future work includes integrating additional defenses against emerging threats in FL and further optimizing the inter-server communication costs.

Acknowledgments

This project received funding from the European Research Council (ERC) under the European Union's research and innovation programs Horizon Europe (PRIVTOOLS/101124778) and Horizon 2020 (PSOTI/850990). It was co-funded by the Deutsche Forschungsgemeinschaft (DFG) within SFB 1119 CROSSING/236615297. It was supported by the German Federal Ministry of Research, Technology and Space (BMFTR), and the Hessian Ministry of Science and Research, Arts and Culture (HMWK) within the National Research Center for Applied Cybersecurity ATHENE.

This work has been partially supported by the French government, through the 3IA Côte d'Azur Investments in the project managed by the National Research Agency (ANR) with reference number ANR-23-IACL-0001

REFERENCES

[Act, 1996] Act, A. (1996). Healt Insurance Portability and Accountability Act.

[Addanki et al., 2022] Addanki, S., Garbe, K., Jaffe, E., Ostrovsky, R., and Polychroniadou, A. (2022). Prio+: Privacy preserving aggregate statistics via boolean shares. In *SCN*.

[Agarwal et al., 2018] Agarwal, N., Suresh, A. T., Yu, F. X., Kumar, S., and McMahan, B. (2018). cpsgd: Communication-efficient and differentially-private distributed SGD. In *NeurIPS*.

[Anguita et al., 2013] Anguita, D., Ghio, A., Oneto, L., Parra, X., Reyes-Ortiz, J. L., et al. (2013). A public domain dataset for human activity recognition using smartphones. In *Esann*.

[Bagdasaryan et al., 2020] Bagdasaryan, E., Veit, A., Hua, Y., Estrin, D., and Shmatikov, V. (2020). How to backdoor federated learning. In *AISTATS*.

[Ben-Itzhak et al., 2024] Ben-Itzhak, Y., Möllering, H., Pinkas, B., Schneider, T., Suresh, A., Tkachenko, O., Vargaftik, S., Weinert, C., Yalame, H., and Yanai, A. (2024). Scionfl: Efficient and robust secure quantized aggregation. In *SaTML*.

[Benhamouda et al., 2016] Benhamouda, F., Joye, M., and Libert, B. (2016). A new framework for privacy-preserving aggregation of time-series data. *ACM Trans. Inf. Syst. Secur.*

[Biggio et al., 2012] Biggio, B., Nelson, B., and Laskov, P. (2012). Poisoning attacks against support vector machines. In *ICML*. icml.cc / Omnipress.

[Blanchard et al., 2017] Blanchard, P., Mhamdi, E. M. E., Guerraoui, R., and Stainer, J. (2017). Machine learning with adversaries: Byzantine tolerant gradient descent. In *NIPS*.

[Cao et al., 2021] Cao, X., Fang, M., Liu, J., and Gong, N. Z. (2021). Fltrust: Byzantine-robust federated learning via trust bootstrapping. In *NDSS*. The Internet Society.

[Castelluccia et al., 2005] Castelluccia, C., Mykletun, E., and Tsudik, G. (2005). Efficient aggregation of encrypted data in wireless sensor networks. In *The Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services*.

[Chandran et al., 2023] Chandran, G. R., Nieminen, R., Schneider, T., and Suresh, A. (2023). Privmail: A privacy-preserving framework for secure emails. In *ESORICS (2)*.

[Chellapandi et al., 2024] Chellapandi, V. P., Yuan, L., Brinton, C. G., Zak, S. H., and Wang, Z. (2024). Federated learning for connected and automated vehicles: A survey of existing approaches and challenges. *IEEE Trans. Intell. Veh.*

[Corrigan-Gibbs and Boneh, 2017] Corrigan-Gibbs, H. and Boneh, D. (2017). Prio: Private, robust, and scalable computation of aggregate statistics. In *NSDI*.

[Demmler et al., 2014] Demmler, D., Herzberg, A., and Schneider, T. (2014). RAID-PIR: practical multi-server PIR. In *CCSW*, pages 45–56. ACM.

[Dong et al., 2021] Dong, Y., Chen, X., Li, K., Wang, D., and Zeng, S. (2021). FLOD: oblivious defender for private byzantine-robust federated learning with dishonest-majority. In *ESORICS*.

[Dwork et al., 2006] Dwork, C., McSherry, F., Nissim, K., and Smith, A. D. (2006). Calibrating noise to sensitivity in private data analysis. In *TCC*.

[Erkin and Tsudik, 2012] Erkin, Z. and Tsudik, G. (2012). Private computation of spatial and temporal power consumption with smart meters. In *ACNS*.

[Fang et al., 2020] Fang, M., Cao, X., Jia, J., and Gong, N. Z. (2020). Local model poisoning attacks to byzantine-robust federated learning. In *USENIX Security Symposium*.

[Fung et al., 2018] Fung, C., Yoon, C. J. M., and Beschastnikh, I. (2018). Mitigating sybils in federated learning poisoning. *CoRR*, abs/1808.04866.

[Gao et al., 2023] Gao, J., Hou, B., Guo, X., Liu, Z., Zhang, Y., Chen, K., and Li, J. (2023). Secure aggregation is insecure: Category inference attack on federated learning. *IEEE Trans. Dependable Secur. Comput.*

[Gehlhar et al., 2023] Gehlhar, T., Marx, F., Schneider, T., Suresh, A., Wehrle, T., and Yalame, H. (2023). Safefl: Mpc-friendly framework for private and robust federated learning. In *IEEE S&P (Workshops)*. Code: https://github.com/encryptogroup/SAFEFL/.

[Goldreich, 2004] Goldreich, O. (2004). *The Foundations of Cryptography - Volume 2: Basic Applications*. Cambridge University Press.

[Goldreich et al., 1987] Goldreich, O., Micali, S., and Wigderson, A. (1987). How to play any mental game or A completeness theorem for protocols with honest majority. In *STOC*.

[Joye and Libert, 2013] Joye, M. and Libert, B. (2013). A scalable scheme for privacy-preserving aggregation of time-series data. In *Financial Cryptography*.

[Kadhe et al., 2020] Kadhe, S., Rajaraman, N., Koyluoglu, O. O., and Ramchandran, K. (2020). Fastsecagg: Scalable secure aggregation for privacy-preserving federated learning. *CoRR*, abs/2009.11248.

[Kairouz et al., 2021] Kairouz, P., Liu, Z., and Steinke, T. (2021). The distributed discrete gaussian mechanism for federated learning with secure aggregation. In *ICML*.

[Kato et al., 2023] Kato, F., Cao, Y., and Yoshikawa, M. (2023). OLIVE: Oblivious Federated Learning on Trusted Execution Environment against the risk of sparsification. In *49th International conference on Very Large DataBases (VLDB)*.

[Khazbak et al., 2020] Khazbak, Y., Tan, T., and Cao, G. (2020). Mlguard: Mitigating poisoning attacks in privacy preserving distributed collaborative learning. In *ICCCN*.

[Konečný et al., 2016] Konečný, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T., and Bacon, D. (2016). Federated learning: Strategies for improving communication efficiency. *CoRR*, abs/1610.05492.

[LeCun et al., 1998] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proc. IEEE*.

[Li et al., 2019] Li, W., Milletarì, F., Xu, D., Rieke, N., Hancox, J., Zhu, W., Baust, M., Cheng, Y., Ourselin, S., Cardoso, M. J., and Feng, A. (2019). Privacy-preserving federated brain tumour segmentation. In *Machine Learning in Medical Imaging*. Springer International Publishing.

[Marx et al., 2023] Marx, F., Schneider, T., Suresh, A., Wehrle, T., Weinert, C., and Yalame, H. (2023). Hyfl: A hybrid approach for private federated learning. *CoRR*, abs/2302.09904.

[McMahan et al., 2017] McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. (2017). Communication-efficient learning of deep networks from decentralized data. In *AISTATS*, volume 54 of *Proceedings of Machine Learning Research*, pages 1273–1282. PMLR.

[Mo et al., 2021] Mo, F., Haddadi, H., Katevas, K., Marin, E., Perino, D., and Kourtellis, N. (2021). Ppfl: Privacy-preserving federated learning with trusted execution environments. In *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '21.

[Nguyen et al., 2022] Nguyen, T. D., Rieger, P., Chen, H., Yalame, H., Möllering, H., Fereidooni, H., Marchal, S., Miettinen, M., Mirhoseini, A., Zeitouni, S., Koushanfar, F., Sadeghi, A., and Schneider, T. (2022). FLAME: taming backdoors in federated learning. In *USENIX Security Symposium*.

[Önen and Molva, 2007] Önen, M. and Molva, R. (2007). Secure data aggregation with multiple encryption. In *Wireless Sensor Networks*.

[Patra et al., 2021] Patra, A., Schneider, T., Suresh, A., and Yalame, H. (2021). ABY2.0: improved mixed-protocol secure two-party computation. In *USENIX Security Symposium*.

[Qiu et al., 2024] Qiu, P., Zhang, X., Ji, S., Fu, C., Yang, X., and Wang, T. (2024). Hashvfl: Defending against data reconstruction attacks in vertical federated learning. *IEEE Trans. Inf. Forensics Secur.*, 19:3435–3450.

[Rathee et al., 2023] Rathee, M., Shen, C., Wagh, S., and Popa, R. A. (2023). ELSA: secure aggregation for federated learning with malicious actors. In *IEEE S&P*.

[Regulation, 2016] Regulation, G. D. P. (2016). Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation) .

[Seif et al., 2020] Seif, M., Tandon, R., and Li, M. (2020). Wireless federated learning with local differential privacy. In *ISIT*.

[Shen et al., 2016] Shen, S., Tople, S., and Saxena, P. (2016). Auror: defending against poisoning attacks in collaborative deep learning systems. In *ACSAC*.

[Shokri et al., 2017] Shokri, R., Stronati, M., Song, C., and Shmatikov, V. (2017). Membership inference attacks against machine learning models. In *IEEE S&P*.

[So et al., 2023] So, J., Ali, R. E., Güler, B., Jiao, J., and Avestimehr, A. S. (2023). Securing secure aggregation: Mitigating multi-round privacy leakage in federated learning. In *AAAI*.

[Stevens et al., 2022] Stevens, T., Skalka, C., Vincent, C., Ring, J. H., Clark, S., and Near, J. P. (2022). Efficient differentially private secure aggregation for federated learning via hardness of learning with errors. In *USENIX Security Symposium*.

[Suresh et al., 2017] Suresh, A. T., Yu, F. X., Kumar, S., and McMahan, H. B. (2017). Distributed mean estimation with limited communication. In *ICML*.

[Triastcyn and Faltings, 2019] Triastcyn, A. and Faltings, B. (2019). Federated learning with bayesian differential privacy. In *IEEE BigData*.

[Truex et al., 2019] Truex, S., Baracaldo, N., Anwar, A., Steinke, T., Ludwig, H., Zhang, R., and Zhou, Y. (2019). A hybrid approach to privacy-preserving federated learning. In *AISec@CCS*.

[Vadhan, 2017] Vadhan, S. P. (2017). The complexity of differential privacy. In *Tutorials on the Foundations of Cryptography*. Springer International Publishing.

[Wen et al., 2019] Wen, Y., Li, W., Roth, H., and Dogra, P. (2019). Federated Learning for Healthcare using NVIDIA Clara.

[Xu et al., 2024] Xu, R., Gao, S., Li, C., Joshi, J., and Li, J. (2024). Dual defense: Enhancing privacy and mitigating poisoning attacks in federated learning. In *NeurIPS*.

[Xu et al., 2023] Xu, Z., Zhang, Y., Andrew, G., Choquette-Choo, C. A., Kairouz, P., McMahan, H. B., Rosenstock, J., and Zhang, Y. (2023). Federated learning of gboard language models with differential privacy. In *ACL (industry)*.

[Yang et al., 2021] Yang, G., Wang, S., and Wang, H. (2021). Federated learning with personalized local differential privacy. In *ICCCS*.

[Yin et al., 2018] Yin, D., Chen, Y., Ramchandran, K., and Bartlett, P. L. (2018). Byzantine-robust distributed learning: Towards optimal statistical rates. In *ICML*.

[Zhao et al., 2021] Zhao, L., Jiang, J., Feng, B., Wang, Q., Shen, C., and Li, Q. (2021). Sear: Secure and efficient aggregation for byzantine-robust federated learning. *IEEE Transactions on Dependable and Secure Computing*.

[Zhou et al., 2020] Zhou, C., Sun, Y., and Wang, D. (2020). Federated learning with gaussian differential privacy. In *RICAI*.