Strengthening Privacy in Robust Federated Learning through Secure Aggregation

Tianyue Chu  
IMDEA Networks Institute  
Universidad Carlos III de Madrid

Devriş İşler  
IMDEA Networks Institute  
Universidad Carlos III de Madrid

Nikolaos Laoutaris  
IMDEA Networks Institute

Abstract—Federated Learning (FL) has evolved into a pivotal paradigm for collaborative machine learning, enabling a centralised server to compute a global model by aggregating the local models trained by clients. However, the distributed nature of FL renders it susceptible to poisoning attacks that exploit its linear aggregation rule called FedAVG. To address this vulnerability, FedQV has been recently introduced as a superior alternative to FedAVG, specifically designed to mitigate poisoning attacks by taxing more than linearly deviating clients. Nevertheless, FedQV remains exposed to privacy attacks that aim to infer private information from clients’ local models. To counteract such privacy threats, a well-known approach is to use a Secure Aggregation (SA) protocol to ensure that the server is unable to inspect individual trained models as it aggregates them. In this work, we show how to implement SA on top of FedQV in order to address both poisoning and privacy attacks. We mount several privacy attacks against FedQV and demonstrate the effectiveness of SA in countering them.

I. INTRODUCTION

Federated Learning (FL) [15], [18] is a recent distributed learning paradigm designed for machine learning across multiple clients. It enables clients to collectively train a global model via a centralised server, all without divulging their raw local training data to the server. Generally, an FL involves an iterative process encompassing three key steps: the server sends the current global model to the clients or a selected subset of them; each selected client trains their local model using its local training data and sending the local model updates back to the server; then the server aggregates the received local model updates adhering to an aggregation method, and utilises it to update the global model. A prominent example of an FL aggregation method is FedAVG [18] developed by Google and applied in tasks such as Google’s emoji [19] and next-word prediction [13] for mobile device keyboards. FedAVG employs a weighted averaging mechanism for local model updates. This weighting is determined by the sizes of the local training datasets, making FedAVG an effective and widely adopted approach in the realm of FL.

However, FedAVG is vulnerable to poisoning attacks, where even a single malicious client can arbitrarily manipulate the global model [2]. This arises from the equal treatment of all local data points, resembling the “one person one vote (1p1v)” election rule. To address this inherent vulnerability, Chu et al. [8] recently proposed a novel method called FedQV, as a superior alternative for FedAVG in the aggregation process. FedQV draws inspiration from Quadratic Voting [16], showcasing improved efficiency and robustness compared to 1p1v. Functioning as a truthful mechanism, FedQV compels clients, including potentially malicious ones, to provide truthful information rather than misinformation. This commitment to truthfulness is reinforced by its masked voting role and limited budget mechanism. FedQV stands out for its simplicity and adaptability, which can be seamlessly integrated into Byzantine-robust FL defence schemes, enhancing their defence capabilities. This characteristic positions FedQV as a promising solution to mitigate vulnerabilities observed in FedAVG concerning poisoning attacks.

In addition to the risk of poisoning attacks, the recent escalating threat of privacy attacks within FL has garnered considerable attention. Despite lacking direct access to clients’ local data, a malicious server can still infer patterns of private information through inference and reconstruction attacks on clients’ local models [10], [26], [28]. This underscores the need for privacy mechanisms in FL to safeguard sensitive user data. A viable solution involves the implementation of secure aggregation (SA) [3]. SA typically signifies a protocol enabling a group of mutually distrustful clients, each holding a private value, to compute an aggregate value without revealing any information about their individual private values to one another. This holds particular relevance in the FL context, where the goal is to have the server perform the aggregation on the clients’ private local models utilising SA. SA in an FL ensures that the server is incapable of accessing clients’ trained models or acquiring information about their private data.

The widespread adoption of SA has primarily occurred in conjunction with the FedAVG framework. This preference is rooted in the inherent simplicity of FedAVG, rendering it an accessible and practical foundation for integrating SA. Given that FedQV stands out as a superior alternative to FedAVG, this paper embarks to implement and adapt SA to make it compatible with FedQV. Through this integration, we aspire to establish a more privacy-preserving defence mechanism in robust FL against potential privacy attacks. More specifically, we make the following two contributions:

- **Implementation of Secure Aggregation (SA) within FedQV:** We incorporate SECAGG into the FedQV framework with specific adjustments, addressing the challenges posed by the unique voting mechanism of
FEDQV that hinders a straightforward integration of secure aggregation protocols. This novel integration serves to enhance the security architecture of FEDQV, providing a nuanced exploration into the synergies between SECAGG and FEDQV.

- **Empirical Assessment of SECAGG’s Privacy Enhancement in FEDQV**: Subsequent to the implementation, we conduct privacy attacks to empirically assess the effectiveness of SECAGG in fortifying privacy within the FEDQV framework. Our findings highlight that implementing SECAGG atop FEDQV significantly improves resistance against 2 privacy attacks.

II. BACKGROUND

A. Reconstruction Attacks in FL

In terms of privacy leakage, communicating gradients throughout the training process in FL can reveal sensitive information about participants. **Deep leakage from gradients (DLG)** [28] demonstrated that sharing the gradients can leak private training data of clients, including images and text. Building upon this, **Improved DLG (iDLG)** [26] presents an analytical procedure to extract the ground-truth labels from the shared gradients and is shown to be even more effective than DLG. **Inverting Gradients (IG)** [10] introduced cosine similarity as a cost function in their reconstruction attacks and employed total variation loss $L_{TV}$ as an image prior. **Inverting Gradients** [10] demonstrated the capability to reconstruct high-resolution images with increased batch sizes.

B. Secure Aggregation for FL

Due to the sensitive nature of the updates sent by clients and the threat from the aforementioned privacy attacks, secure aggregation (SA) protocols have been proposed as potential countermeasures [17]. Secure aggregation can be achieved using four main privacy-enhancing technologies: differential privacy [11], trusted-execution environment (TEE) [27], secure shuffling under anonymity assumptions [14], and cryptography. Among those, secure aggregation based on cryptography is the most widely studied [4], [22], [23]. Existing secure aggregation solutions have been proposed for different contexts [17], including under malicious clients, a malicious aggregator(s), and for varying communication and computation costs, etc. Although crypto-based secure aggregation provides strong security guarantees compared to alternatives, i.e., differential privacy, it also suffers from high computational and communication overhead making it impractical for many real-life use cases. Among the different proposals, the one based on masking [4] outperforms its alternatives in terms of performance and ensures a strong security guarantee.

C. Aggregation Methods in FL

Aggregation plays a pivotal role in FL, with FEDAVG being the predominant aggregation method due to its simplicity and effectiveness. However, FEDAVG is susceptible to poisoning attacks. There exist several Byzantine-robust FL aggregation methods for mitigating this vulnerability. They are either statistics-based outlier detection techniques [2], [7], [24], [25], or utilise auxiliary labelled data collected by the aggregation server to verify the correctness of the received gradients [6], [12]. Both approaches require examining the properties of the updates of individual clients, which can jeopardise their privacy due to aforementioned privacy attacks [10], [28] mounted by an honest-but-curious server. To defend against privacy attacks, the most secure way is to use cryptographic techniques [28]. However, the computational complexity of these robust aggregation methods poses challenges, particularly when integrating them with SA, which also imposes a heavy computational overhead.

FEDQV presents a superior alternative to FEDAVG, demonstrating enhanced resilience against poisoning attacks owing to its Quadratic Voting nature and defence mechanism while inheriting the simplicity of FEDAVG. This attribute enables the integration of FEDQV with specialised adjustments into existing SA protocols, which are originally established in the FEDAVG framework. This integration aims to fortify the defence capabilities against both poisoning and privacy attacks.

III. SECURE AGGREGATION IN FEDQV

We first overview FEDQV and the secure aggregation concepts we incorporate in our design. Later, we describe how we tailored adjusted the FEDQV protocol to implement secure aggregation into FEDQV.

A. FEDQV Overview

1) **Federated Learning Setting**: Consider a horizontal FL system involving $N$ clients with a local dataset and a central server. In each training round $t$, a specific subset of clients $S^t$ is selected to participate in the training task. Client $i$ has the local dataset $D_i$ with $|D_i|$ samples, drawn from Non-IID distribution $X_i(\mu_i, \sigma_i^2)$. The overarching goal of employing FL is to train a global model on the server by aggregating the local models of clients, weighted by their refined votes. Given the loss function $\ell(\mathbf{w}; \mathcal{D})$, the objective function of FL can be described as $\mathcal{L}(\mathbf{w}) = \mathbb{E}_{\mathcal{D} \sim \mathcal{X}}[\ell(\mathbf{w}; \mathcal{D})]$. Therefore, the task becomes: $w^* = \text{arg min}_{\mathbf{w} \in \mathbb{R}^d} \mathcal{L}(\mathbf{w})$. To find the optimal $w^*$, Stochastic Gradient Descent (SGD) is employed to optimise the objective function. Let $T$ be the total number of every client’s SGD, $E$ be the local iterations between two communication rounds, and thus $\frac{t}{E}$ is the number of communication rounds.

2) **FEDQV**: FEDQV consists of two key components: similarity computation and voting scheme.

**Similarity Computation.** In round $t$, client $i$ ($i \in S^t$) trains its local model $\mathbf{w}_i^t$ using the local data. Then the client calculates its similarity score $s_i^t$:

$$s_i^t = S_{\cos}(\mathbf{w}_i^t, \mathbf{w}_{t-1}^i) = \frac{\langle \mathbf{w}_i^t, \mathbf{w}_{t-1}^i \rangle}{\| \mathbf{w}_i^t \| \cdot \| \mathbf{w}_{t-1}^i \|}$$

(1)

Subsequently, each client transmits its model updates $\mathbf{w}_i^t$ to the central server, with the message $\langle s_i^t \rangle$.

**Voting Scheme** Upon receiving the updates and messages from selected clients, the voting scheme proceeds as follows:

- The server normalises the similarity scores using Min-Max scaling to obtain $\tilde{s}_i^t$ as:

$$\tilde{s}_i^t = \text{norm} \left( s_i^t, \{ s_j^t \}_{j \in \mathcal{N}_t} \right)$$

(2)
Algorithm 1: FedQV

\begin{algorithmic}
  \State \textbf{Input : } $w^0 \leftarrow$ random initialisation;
  $B, \theta \leftarrow$ FedQV parameters
  \State \textbf{Server :} For $\text{Iteration } t \leftarrow 1$ to $\mathbb{F}_E$
  \State \hspace{1em} Broadcast $w^{t-1}$ to selected clients $S^t$
  \State \hspace{1em} $(|S^t| = C \geq 1)$;
  \State \hspace{1em} Receive the local updates $(w^i_t, s^i_t)$ and compute
  \hspace{1em} the normalised $s^i_t$
  \State \hspace{1em} For $i \leftarrow 1$ to $N$ do in parallel
  \State \hspace{2em} if $s^i_t \leq \theta$ or $s^i_t \geq 1 - \theta$ then
  \State \hspace{3em} Penalise budget $B_i \leftarrow$ Eq.(3);
  \State \hspace{3em} Calculate voice credit $c^i_t \leftarrow$ Eq.(4);
  \State \hspace{3em} Calculate vote $v^i_t \leftarrow$ Eq.(5);
  \State \hspace{3em} Update budget $B_i \leftarrow$ Eq.(6)
  \State \hspace{1em} end
  \State \hspace{1em} return $w^t \leftarrow \sum_{i=1}^{N} \frac{w^i_t}{\sum_{i=1}^{N} v^i_t}$
\end{algorithmic}

\begin{itemize}
  \item The server penalises abnormal similarity scores by reducing the clients’ budgets $B_i$ as:
    $$B_i = \max(0, B_i + \ln s^i_t - 1)$$ (3)
  \item The server calculates the voice credit $c^i_t$ for client $i$ using the masked voting rule $\mathcal{H}$ as:
    $$c^i_t = \mathcal{H}(s^i_t) = (-\ln s^i_t + 1)1_{\theta < s^i_t < 1 - \theta}$$ (4)
  \item The server checks the budget $B_i$ for each client and computes their final votes $v^i_t$ as:
    $$v^i_t = \min(c^i_t, \max(0, B_i))$$ (5)
  \item The server updates the budget as:
    $$B_i = \max(0, B_i - (v^i_t)^2)$$ (6)
\end{itemize}

Algorithm 1 summarises all these steps of FedQV.

B. SECAGG Concepts

A secure aggregation protocol can guarantee input confidentiality by safeguarding the local input model updates of each participant and revealing only the globally aggregated model during FL training, thereby efficiently averting or alleviating concerns regarding the aforementioned privacy attacks. For our work, we utilise the SECAGG protocol proposed by [4] due to its performance in terms of communication and computational overhead. Below we overview the SECAGG protocol of [4] (see [4] for their complete protocol). The secure aggregation protocol consists of three phases: SECAGG.Setup, SECAGG.Produce, and SECAGG.Aggregate.

- In the SECAGG.Setup phase, each client and the aggregator get the public parameters and the key materials. Each client generates two masks: $(k_i, b_i)$ where $k_i$ is generated using a key agreement protocol (e.g., Diffie Hellman [9]), and $b_i$ is generated via a pseudorandom generator. Each client secret shares its mask with other clients using threshold secret sharing [21]. With the secret sharing, the masks of dropped clients can be recovered as long as threshold-many clients are still alive.
- In the SECAGG.Produce phase, each client protects their local inputs using their unique private key and sends the protected input, denoted by $[w^i_t]$, to the aggregator.
  $$[w^i_t] = w^i_t + k_i + b_i$$ (7)
- In the SECAGG.Aggregate phase, after the aggregator collects all the protected inputs from the clients, it executes an aggregation algorithm to retrieve the sum of the client’s inputs. The aggregator first collects threshold-many shares of the seed of each mask $b_i$ for every alive client $i$ and reconstructs it. Then it gets threshold-many shares of the Diffie-Hellman’s secret key of the dropped clients and thus reconstructs the missing masks $k_j$ for every dropped client $j$.
  $$\sum_{i \in \mathbb{N}} w^i_t = \sum_{i=1}^{N} [w^i_t] - \sum_{i \in \text{Drop}} b_i + \sum_{j \in \text{Live}} k_j$$ (8)

All masking and unmasking operations involve modular addition.

Threat Model and Properties. We assume that the clients are honest-but-curious meaning that clients correctly follow the protocol steps but remain curious to discover any private information. All communications between clients are through secure-and-authenticated channels, offering a robust layer of protection against unauthorised access and data tampering.

C. Implementing SECAGG in FedQV

FedQV differs from FedAVG, which as its name suggests uses averaging for integration, by instead using a voting mechanism to weigh the contributions of $w^i_t$ to the global model. This makes it not-trivial to integrate upon FedQV secure aggregation protocols developed for FedAVG. Related integration challenges include the following:

- Aggregation of $\{w^i_t\}_{i \in \mathbb{N}}$. Recall that the aggregation formulated by FedQV is $\frac{1}{\sum_{i=1}^{N} v^i_t} \sum_{i=1}^{N} v^i_t \cdot w^i_t$. Assume a client $i$ masks the value of $w^i_t$ as in Equation 8. However, there is one more parameter $v^i_t$ to be added in the calculation. As shown by the equation below, if the server simply multiplies $[w^i_t]$ by its corresponding $v^i_t$, the masks do not cancel each other as supposed. Thus, $v^i_t$ must be added before a client masks its $w^i_t$.
  $$\sum_{i=1}^{n} v^i_t \cdot [w^i_t] - \sum_{i \in \text{Drop}} b_i + \sum_{j \in \text{Live}} k_j$$
  $$\sum_{i \in \mathbb{N}} w^i_t \neq \sum_{i=1}^{n} v^i_t \cdot (w^i_t + k_i + b_i) - \sum_{i \in \text{Drop}} b_i + \sum_{j \in \text{Live}} k_j$$
• **Calculation of** \(v_t^i\). As mentioned, each client shall add its \(v_t^i\) to its \(w_t^i\), but in FedQV \(v_t^i\) can only be calculated by the server based on the budget \(B_i\), similarity \(s_t^i\), and the size of the data \(|D_t^i|\) of client \(i\). To overcome this shortcoming, the server, after receiving \(s_t^i\) and \(|D_t^i|\) of each client \(i\), computes \(v_t^i\) using Equation 5 and sends \(v_t^i\) back to its corresponding client. Then each client computes SECAGG.Protect using \(\{k_t^i, b_t^i\}\) in which the input is assigned as \(w_t^i \cdot v_t^i\). Note that this does not harm the security and privacy of the clients while introducing negligible computational overhead.

Considering the crucial adjustments above, let us form the securely aggregated FedQV. Our approach is in line with other securely aggregated FL designs consisting of three main stages: 1) Setup; 2) Generation and Protect; and 3) Aggregate. Our integration of secure aggregation to FedQV is depicted in Figure 1. We further discuss its details below:

**Stage 0 (Setup).** In this stage, the server and the clients compute SECAGG.Setup and initialise FedQV by having the server send values of the protocol’s parameters \(w_t^{i-1}\) to participating clients.

**Stage 1 (Generation and Protect).** After the setup and initialisation stage, each client first computes its local model \(w_t^i\) as in FedQV and computes its \(s_t^i\). Later, it transmits the message \(\langle s_t^i \rangle\). Upon receiving \(\langle s_t^i \rangle\), the server computes \(v_t^i\) as in Equation 5 and transmits \(v_t^i\) to the corresponding client. Then client \(i\) first computes \(w_t^i \cdot v_t^i\) and protects its input \(w_t^i \cdot v_t^i\) by treating it as an input to the SECAGG.Protect as \([w_t^i] = w_t^i \cdot v_t^i + k_t + b_t\). Client \(i\) sends \([w_t^i]\) to the server.

**Stage 3 (Aggregate).** Upon receiving \([w_t^i]\), the server first calls SECAGG.Aggregate as described previously which returns the sum of all \(w_t^i\). The server requires one more step to finalise by multiplying the sum retrieved by \(1/\sum_{i=1}^{N} v_t^i\), as it is the sole entity possessing complete knowledge of all \(v_t^i\).

Due to the nature of FedQV and our adjustments, there are two attack scenarios we consider: 1) Privacy attacks by the server; and 2) Client attacking \(v_t^i\). In the former scenario, the server knows only \(s_t^i\) of each client, therefore the server can’t reconstruct the local model \(w_t^i\) of clients by relying solely on similarity scores. Moreover, \(s_t^i\) can be protected from the server by utilising cryptography (see Section V for further discussion). In the latter scenario, it can be a concern that each client possesses the knowledge of their vote. This issue can be effectively mitigated by enforcing clients to provide proof (e.g., via zero-knowledge proofs [20]) that the \(s_t^i\) sent is indeed computed honestly and correctly, adhering to the protocol rules.

**IV. PRIVACY ATTACKS AGAINST FEDQV WITH SECAGG**

In this section, we conduct a comprehensive evaluation to quantify the privacy-preserving properties of FedQV with and without SECAGG. By evaluating these methods under real-world conditions, we seek to evaluate their effectiveness under true attack scenarios.

**A. Experimental Setup**

We evaluate models Conv-2 [5] along with the corresponding datasets FEMNIST [5], commonly employed in FL studies.

**Algorithm 2: DLG and GI Attacks**

```plaintext
1 Input: \(F(D_t, w_t^i)\): model at round \(t\) from targeted user \(i\); learning rate \(\eta\) for inverting gradient optimiser; \(S\): max iterations for attack; \(\tau\): regularisation term for cosine loss in inverting gradient attack; \(d\):the model size
2 Output: reconstructed training data \((D_t, y_t)\) at round \(t\)
3 Initialise \(D_0 \leftarrow \mathcal{N}(0,1)\), \(y_0 \leftarrow \text{RandInt}(0,\max(y))\)
4 for \(s=0,1, \ldots, S - 1\) do
5    \(\nabla w_t^i \leftarrow \partial (F(D_t, w_t^i), y_t) / \partial w_t^i\)
6    switch Case do
7       case DLG attack do \(\mathcal{L}_s' \leftarrow || \nabla w_t^i - \nabla w_t^i ||^2\)
8       case GI attack do \(\mathcal{L}_s' \leftarrow 1 - \frac{\nabla w_t^i \cdot \nabla w_t^i}{|| \nabla w_t^i || \cdot || \nabla w_t^i ||} + \tau\)
9    \(D_{s+1} \leftarrow D_s - \eta \nabla \mathcal{L}_s', y_{s+1} \leftarrow y_s - \eta \nabla_y \mathcal{L}_s'\)
10   return \(D_S, y_S\)
```

In each iteration \(t\), the attack algorithms proceed as follows:

1) The attack randomly initialises a set of dummy data, comprising dummy inputs \(D_0^d\) and dummy labels \(y_0^d\).
2) After the dummy gradient \(w_t^d\) is acquired, the server then updates the dummy data in the direction that minimises the Euclidean distance (for DLG) or the cosine distance (for GI) between the dummy gradient and the real gradient \(\nabla w_t^i\).
can readily extend the assumption to malicious clients. Therefore, our paper extensively discusses and addresses these open problems as follows:

**Enhanced Privacy Attacks:** While the integration of SECAGG has proven effective in mitigating privacy leakage and defending against privacy attacks, there are instances where residual information leakage persists even after employing SECAGG. Our future work will delve into this aspect by investigating more potent privacy attack scenarios, aiming to comprehensively evaluate the impact of secure aggregation on privacy in FL. Additionally, we plan to extend our investigations to encompass a broader range of datasets and models, providing a more thorough understanding of the results.

**VI. CONCLUSION**

In this paper, we have illustrated the implementation of SECAGG on the FedQV framework to enhance its defense mechanisms against privacy attacks. The integration of SECAGG into the FedQV framework demonstrated significant improvements in resisting privacy attacks. Empirical assessments, including Deep Leakage from Gradients (DLG) and Gradient Inversion (GI) attacks, revealed enhanced privacy protection when SECAGG was applied. Notably, SECAGG successfully prevented information leakage in scenarios where FedQV alone exhibited vulnerabilities. This empirical evidence contributes valuable insights, affirming the effectiveness of implementing SECAGG atop FedQV as a privacy-enhancing measure in robust FL.
ACKNOWLEDGEMENT
Tianyue Chu and Nikolaos Laoutaris were supported by the MLEDGE project (RE-GAGE22e00052829516), funded by the Ministry of Economic Affairs and Digital Transformation and the European Union NextGenerationEU/PRTR. Devriş İşler was supported by the European Union’s HORIZON project DataBri-X (101070069).

REFERENCES