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

Workshop on AI Systems with Confidential Computing (AISCC) 2024
26 February 2024, San Diego, CA, USA
ISBN 979-8-9894372-4-5
https://dx.doi.org/10.14722/aiscc.2024.23012
www.ndss-symposium.org

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 $\mathcal{L}_{\mathcal{TV}}$ 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 $\mathcal{S}^t$ is selected to participate in the training task. Client i has the local dataset $\mathcal{D}_i$ with $|\mathcal{D}_i|$ samples, drawn from Non-IID distribution $\mathcal{X}_i(\mu_i, \sigma_i^2)$. 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 $\ell(\boldsymbol{w}; \mathcal{D})$, the objective function of FL can be described as $\mathcal{L}(\boldsymbol{w}) = \mathbb{E}_{\mathcal{D}\sim\mathcal{X}}\left[\ell(\boldsymbol{w}; \mathcal{D})\right]$. Therefore, the task becomes: $\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}\in\mathbb{R}^d} \mathcal{L}(\boldsymbol{w})$. To find the optimal $\boldsymbol{w}^*$, 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 ($i \in \mathcal{S}^t$) trains its local model $\boldsymbol{w}_i^t$ using the local data. Then the client calculates its similarity score s_i^t:

$$s_i^t = S_{\cos}\left(\boldsymbol{w}_i^t, \boldsymbol{w}^{t-1}\right) = \frac{\left\langle \boldsymbol{w}_i^t, \boldsymbol{w}^{t-1} \right\rangle}{\|\boldsymbol{w}_i^t\| \cdot \|\boldsymbol{w}^{t-1}\|} \tag{1}$$

Subsequently, each client transmits its model updates $\boldsymbol{w}i^t$ 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 $\bar{s}_i^t$ as:

$$\bar{s}_i^t = \text{norm}\left(s_i^t, \{s_i^t\}_{i\in[N]}\right) \tag{2}$$

Algorithm 1: FEDQV

Input : $\boldsymbol{w}^0 \leftarrow$ random initialisation;
B, $\theta \leftarrow$ FEDQV parameters

Server :

for *Iteration* $t \leftarrow 1$ **to** $\frac{T}{E}$ **do**
  Broadcast $\boldsymbol{w}^{t-1}$ to selected clients $\mathcal{S}^t$ ($|\mathcal{S}^t| = \mathcal{C} \geq 1$);
  Receive *the local updates* $(\boldsymbol{w}_i^t, s_i^t)$ and compute the normalised $\bar{s}_i^t$;
  for $i \leftarrow 1$ **to** N **do in parallel**
    if $\bar{s}_i^t \leq \theta$ *or* $\bar{s}_i^t \geq 1-\theta$ **then**
      Penalise *budget* $B_i \leftarrow$ Eq.(3)
    Calculate *credit voice* $c_i^t \leftarrow$ Eq.(4):
    Calculate *vote* $v_i^t \leftarrow$ Eq.(5);
    Update *budget* $B_i \leftarrow$ Eq.(6)
  end
  return $\boldsymbol{w}^t \leftarrow \sum_{i=1}^N \frac{v_i^t}{\sum_{i=1}^N v_i^t} \boldsymbol{w}_i^t$
end

Client :

for *Client* $i \in \mathcal{S}^t$ **do in parallel**
  Receive *the global update* $\boldsymbol{w}^{t-1}$;
  Conduct local training;
  Calculate *the similarity score* $s_i^t \leftarrow$ Eq.(1);
  send back $(\boldsymbol{w}_i^t, s_i^t)$
end

- The server penalises abnormal similarity scores by reducing the clients' budgets B_i as:

$$B_i = \max\left(0, B_i + \ln \bar{s}_i^t - 1\right) \tag{3}$$

- The server calculates the voice credit c_i^t for client i using the masked voting rule $\mathcal{H}$ as:

$$c_i^t = \mathcal{H}(\bar{s}_i^t) = \left(-\ln \bar{s}_i^t + 1\right) \mathbb{1}_{\theta < \bar{s}_i^t < 1-\theta} \tag{4}$$

- The server checks the budget B_i for each client and computes their final votes v_i^t as:

$$v_i^t = \sqrt{\min\left(c_i^t, \max\left(0, B_i\right)\right)} \tag{5}$$

- The server updates the budget as:

$$B_i = \max(0, B_i - \left(v_i^t\right)^2) \tag{6}$$

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 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.Protect, 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: $(\mathsf{k}_i, \mathsf{b}_i)$ 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.Protect phase, each client protects their local inputs using their unique private key and sends the protected input, denoted by $[\boldsymbol{w}_i^t]$, to the aggregator.

$$[\boldsymbol{w}_i^t] = \boldsymbol{w}_i^t + \mathsf{k}_i + \mathsf{b}_i \tag{7}$$

- 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.

$$\sum_{i\in[N]} \boldsymbol{w}_i^t = \sum_{i=1}^{N} [\boldsymbol{w}_i^t] - \sum_{i\in Drop} \mathsf{b}_i + \sum_{j\in Live} \mathsf{k}_j \tag{8}$$

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 $\boldsymbol{w}_i^t$ 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* $\{\boldsymbol{w}_i^t\}_{i\in[N]}$. Recall that the aggregation formulated by FEDQV is $\frac{1}{\sum_{i=1}^N v_i^t} \sum_{i=1}^N v_i^t \cdot \boldsymbol{w}_i^t$. Assume a client i masks the value of $\boldsymbol{w}_i^t$ 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 $[\boldsymbol{w}_i^t]$ 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 $\boldsymbol{w}_i^t$.

$$\sum_{i=1}^{n} v_i^t \cdot [\boldsymbol{w}_i^t] - \sum_{i\in Drop} \mathsf{b}_i + \sum_{j\in Live} \mathsf{k}_j$$

$$\sum_{i\in[n]} \boldsymbol{w}_i^t \neq \sum_{i=1}^{n} v_i^t \cdot (\boldsymbol{w}_i^t + \mathsf{k}_i + \mathsf{b}_i) - \sum_{i\in Drop} \mathsf{b}_i + \sum_{j\in Live} \mathsf{k}_j$$

- *Calculation of* v_i^t. As mentioned, each client shall add its v_i^t to its $\boldsymbol{w}_i^t$, but in FEDQV v_i^t can only be calculated by the server based on the budget B_i, similarity s_i^t, and the size of the data $|\mathcal{D}_i|$ of client i. To overcome this shortcoming, the server, after receiving s_i^t and $|\mathcal{D}_i|$ of each client i, computes v_i^t using Equation 5 and sends v_i^t back to its corresponding client. Then each client computes SECAGG.Protect using $\langle \mathsf{k}_i, \mathsf{b}_i \rangle$ in which the input is assigned as $\boldsymbol{w}_i^t \cdot v_i^t$. 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 $\boldsymbol{w}^{t-1}$ to participating clients.

Stage 1 (*Generation and Protect*). After the setup and initialisation stage, each client first computes its local model $\boldsymbol{w}_i^t$ as in FEDQV and computes its s_i^t. Later, it transmits the message $\langle s_i^t \rangle$. Upon receiving $\langle s_i^t \rangle$, the server computes v_i^t as in Equation 5 and transmits v_i^t to the corresponding client. Then client i first computes $\boldsymbol{w}_i^t \cdot v_i^t$ and protects its input $\boldsymbol{w}_i^t \cdot v_i^t$ by treating it an input to the SECAGG.Protect as $[\boldsymbol{w}_i^t] = \boldsymbol{w}_i^t \cdot v_i^t + \mathsf{k}_i + \mathsf{b}_i$. Client i sends $[\boldsymbol{w}_i^t]$ to the server.

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

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_i^t. In the former scenario, the server knows only s_i^t of each client, therefore the server can't reconstruct the local model ($\boldsymbol{w}_i^t$) of clients by relying solely on similarity scores. Moreover, s_i^t can be protected from the server by utilising cryptography (see Section V for further discussion).(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_i^t 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.

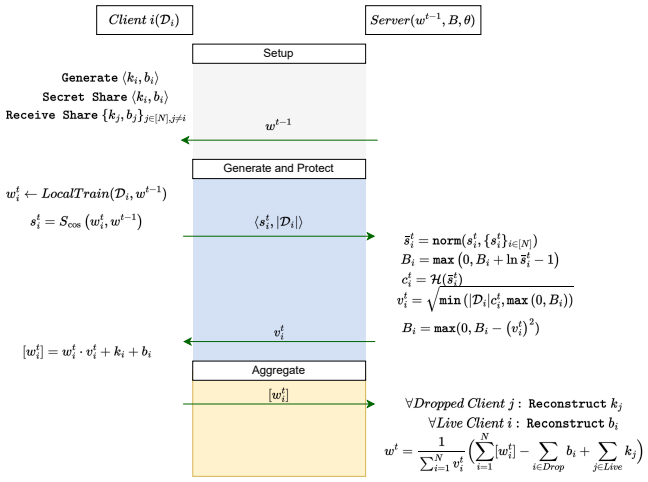


Fig. 1: Overview of implementation of the SECAGG protocol in FEDQV.

In each round of training, we randomly select 10 clients from a pool of 193 users for the FEMNIST dataset. The training process involves 100 iterations with a batch size of 20, and local updates and a learning rate of 0.25.

B. Privacy Attack Algorithm

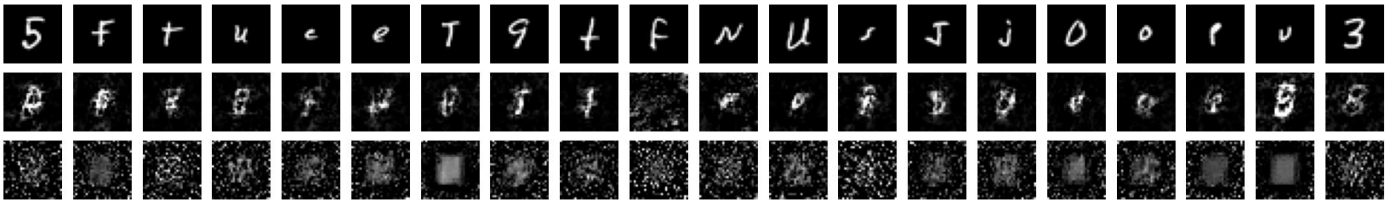
We mount the Deep Leakage from Gradients (DLG) attack [28] and the Gradient Inversion (GI) attack [10] on FEDQV. These attacks aim to generate dummy data and corresponding labels by leveraging a gradient-matching objective. The detailed algorithms are outlined in Algorithm 2.

Algorithm 2: DLG and GI Attacks

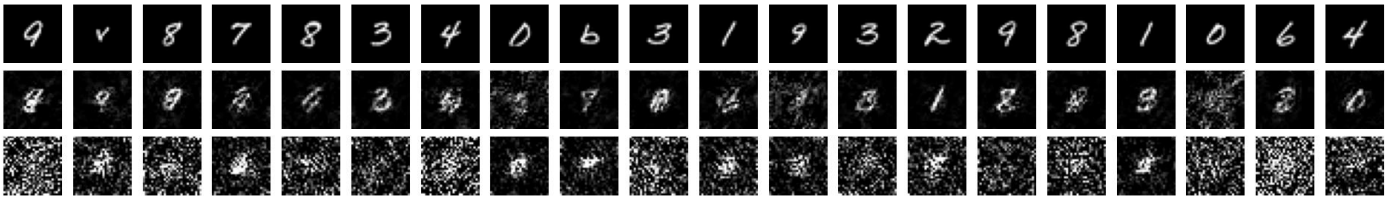
Input: $F(\mathcal{D}_i; \boldsymbol{w}_i^t)$: model at round t from targeted user i; learning rate η for inverting gradient optimiser; S: max iterations for attack; τ: regularisation term for cosine loss in inverting gradient attack; d:the model size
Output: reconstructed training data $(\mathcal{D}_i, y_i)$ at round t
  Initialise $\mathcal{D}'_0 \leftarrow \mathcal{N}(0,1)$, $y'_0 \leftarrow Randint(0, max(y))$
  for $s \leftarrow$ *0,1, ...* $S-1$ **do**
    $\nabla \boldsymbol{w}'_s \leftarrow \partial \ell(F(\mathcal{D}'_s, \boldsymbol{w}_i^t), y'_s)/\partial \boldsymbol{w}_i^t$
    switch *Case* **do**
      case *DLG attack* **do** $\mathcal{L}'_s \leftarrow \| \nabla \boldsymbol{w}'_s - \nabla \boldsymbol{w}_i^t \|^2$
      case *GI attack* **do** $\mathcal{L}'_s \leftarrow 1 - \frac{\nabla \boldsymbol{w}_i^t \cdot \nabla \boldsymbol{w}'_s}{\|\nabla \boldsymbol{w}_i^t\| \|\nabla \boldsymbol{w}'_s\|} + \tau$
    $\mathcal{D}'_{s+1} \leftarrow \mathcal{D}'_s - \eta \nabla_{\mathcal{D}'_s} \mathcal{L}'_s$, $y'_{s+1} \leftarrow y'_s - \eta \nabla_{y'_s} \mathcal{L}'_s$
  return $\mathcal{D}'_S, y'_S$

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

1) The attack randomly initialises a set of dummy data, comprising dummy inputs $\mathcal{D}'_0$ and dummy labels y'_0;
2) After the dummy gradient $\boldsymbol{w}'$ 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 $\nabla \boldsymbol{w}_i^t$.



(a) The Deep Leakage from Gradients (DLG) Attack.



(b) The Gradient Invention (GI) Attack.

Fig. 2: Visual Comparison of Images Reconstructed by Privacy Attacks (DLG and GI) on FEDQV with and without SECAGG. The first row displays the original private training images from the targeted client. The second row illustrates reconstructed images resulting from the privacy attacks (DLG and GI) on FEDQV without SECAGG. In the third row, reconstructed images from the privacy attacks (DLG and GI) on FEDQV with SECAGG are presented. The use of SECAGG is observed to contribute significantly to mitigating information leakage from the images.

We initiate these attacks with a learning rate of 0.01 and perform 10,000 attack iterations.

C. Evaluation Results

We conduct the Deep Leakage from Gradients (DLG) and Gradient Inversion (GI) attacks on the FEDQV framework, both with and without the integration of SECAGG. Figure 2a provides visualizations of the reconstructed images by the DLG attack. In the absence of SECAGG, the DLG attack achieves a 20% recovery rate of private images from the client. However, when SECAGG is integrated, the DLG attack fails to recover any of the client's images. Figure 2b presents visualisations of the reconstructed images by the GI attack. Without SECAGG, the GI attack successfully recovers 50% of the private images from the client. Yet, with SECAGG, the GI attack is unable to recover any client images, although it may vaguely leak the overall shape of the image. These results underscore the pivotal role of SECAGG in fortifying FEDQV against privacy attacks, with the GI attack exhibiting a higher level of potency compared to the DLG attack.

V. DISCUSSION

In this section, we discuss possible research directions and open problems as follows:

*Malicious Clients:*In this paper, we operate under the assumption that clients are honest-but-curious for simplicity, which allows us to focus on the core integration without delving into discussions on potential malicious actions by clients. However, clients can act maliciously to deviate from the protocol. For instance, a malicious client can send the wrong s_i^t while also conducting poisoning attacks. These attacks have been extensively discussed and addressed in the FEDQV and can be mitigated further by cryptography. Therefore, our paper can readily extend the assumption to malicious clients.

Protection of v_i: The server has to send v_i^t to its corresponding client in order to successfully and correctly aggregate $\boldsymbol{w}^t$. Note that FEDQV does not allow clients to learn their votes. While our adjustment is letting clients know their v_i does not harm their privacy, it requires further investigation on its impact. However, the server can also protect v_i (as clients do for their $\boldsymbol{w}_i^t$) and then share the protected version. Homomorphic Encryption [1] can be deployed to actualise our approach, but we leave further investigation as future work.

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

[1] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, "A survey on homomorphic encryption schemes: Theory and implementation," *ACM Comput. Surv.*, 2018. [Online]. Available: https://doi.org/10.1145/3214303

[2] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[3] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for federated learning on user-held data," *arXiv preprint arXiv:1611.04482*, 2016.

[4] ——, "Practical secure aggregation for privacy-preserving machine learning," in *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1175–1191.

[5] S. Caldas, S. M. K. Duddu, P. Wu, T. Li, J. Konečnỳ, H. B. McMahan, V. Smith, and A. Talwalkar, "Leaf: A benchmark for federated settings," in *33rd Conference on Neural Information Processing Systems (NeurIPS)*, 2019.

[6] X. Cao, M. Fang, J. Liu, and N. Gong, "Fltrust: Byzantine-robust federated learning via trust bootstrapping," in *Proceedings of Network and Distributed System Security Symposium (NDSS)*, 2021.

[7] T. Chu, A. Garcia-Recuero, C. Iordanou, G. Smaragdakis, and N. Laoutaris, "Securing federated sensitive topic classification against poisoning attacks," in *Proceedings of Network and Distributed System Security Symposium (NDSS)*, 2022.

[8] T. Chu and N. Laoutaris, "FedQV: Leveraging quadratic voting in federated learning," *arXiv preprint arXiv:2401.01168*, 2024.

[9] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Trans. Inf. Theory*, 1976. [Online]. Available: https://doi.org/10.1109/TIT.1976.1055638

[10] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting gradients-how easy is it to break privacy in federated learning?" *Advances in Neural Information Processing Systems(NeurIPS)*, vol. 33, pp. 16 937–16 947, 2020.

[11] S. Goryczka, L. Xiong, and V. S. Sunderam, "Secure multiparty aggregation with differential privacy: a comparative study," in *EDBT/ICDT Conferences, EDBT/ICDT*. ACM, 2013. [Online]. Available: https://doi.org/10.1145/2457317.2457343

[12] H. Guo, H. Wang, T. Song, Y. Hua, Z. Lv, X. Jin, Z. Xue, R. Ma, and H. Guan, "Siren: Byzantine-robust federated learning via proactive alarming," in *Proceedings of the ACM Symposium on Cloud Computing*, 2021, pp. 47–60.

[13] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," *arXiv preprint arXiv:1811.03604*, 2018.

[14] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai, "Cryptography from anonymity," in *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, 2006, pp. 239–248.

[15] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[16] S. P. Lalley and E. G. Weyl, "Quadratic voting: How mechanism design can radicalize democracy," in *AEA Papers and Proceedings*, vol. 108, 2018, pp. 33–37.

[17] M. Mansouri, M. Önen, W. B. Jaballah, and M. Conti, "Sok: Secure aggregation based on cryptographic schemes for federated learning," *Proc. Priv. Enhancing Technol.*, 2023. [Online]. Available: https://doi.org/10.56553/popets-2023-0009

[18] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[19] S. Ramaswamy, R. Mathews, K. Rao, and F. Beaufays, "Federated learning for emoji prediction in a mobile keyboard," *arXiv preprint arXiv:1906.04329*, 2019.

[20] S. T. V. Setty, "Spartan: Efficient and general-purpose zksnarks without trusted setup," in *Annual International Cryptology Conference, CRYPTO*, ser. Lecture Notes in Computer Science. Springer, 2020. [Online]. Available: https://doi.org/10.1007/978-3-030-56877-1_25

[21] A. Shamir, "How to share a secret," *Commun. ACM*, 1979. [Online]. Available: https://doi.org/10.1145/359168.359176

[22] G. Tsaloli, B. Liang, C. Brunetta, G. Banegas, and A. Mitrokotsa, "sfDEVA: decentralized, verifiable secure aggregation for privacy-preserving learning," in *Information Security -International Conference, ISC*, ser. Lecture Notes in Computer Science. Springer, 2021. [Online]. Available: https://doi.org/10.1007/978-3-030-91356-4_16

[23] D. Wu, M. Pan, Z. Xu, Y. Zhang, and Z. Han, "Towards efficient secure aggregation for model update in federated learning," in *IEEE Global Communications Conference, GLOBECOM*. IEEE, 2020. [Online]. Available: https://doi.org/10.1109/GLOBECOM42002.2020.9347960

[24] C. Xie, S. Koyejo, and I. Gupta, "Zeno: Distributed stochastic gradient descent with suspicion-based fault-tolerance," in *International Conference on Machine Learning*. PMLR, 2019, pp. 6893–6901.

[25] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *International Conference on Machine Learning*. PMLR, 2018, pp. 5650–5659.

[26] B. Zhao, K. R. Mopuri, and H. Bilen, "idlg: Improved deep leakage from gradients," 2020.

[27] L. Zhao, J. Jiang, B. Feng, Q. Wang, C. Shen, and Q. Li, "SEAR: secure and efficient aggregation for byzantine-robust federated learning," *IEEE Trans. Dependable Secur. Comput.*, 2022. [Online]. Available: https://doi.org/10.1109/TDSC.2021.3093711

[28] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," *Advances in Neural Information Processing Systems(NeurIPS)*, vol. 32, 2019.