FASIL: A challenge-based framework for secure and privacy-preserving federated learning

Ferhat Karakoç[*], Betül Güvenç Paltun[†], Leyli Karaçay[‡], Ömer Tuna[§], Ramin Fuladi[¶] and Utku Gülen[‖]

Ericsson Research, Istanbul, Türkiye

Abstract

Enhancing privacy in federal learning (FL) without considering robustness can create an open door for attacks such as poisoning attacks on the FL process. Thus, addressing both the privacy and security aspects simultaneously becomes vital. Although, there are a few solutions addressing both privacy and security in the literature in recent years, they have some drawbacks such as requiring two non-colluding servers, heavy cryptographic operations, or peer-to-peer communication topology. In this paper, we introduce a novel framework that allows the server to run some analysis for detection and mitigation of attacks towards the FL process, while satisfying the confidentiality requirements for the training data against the server. We evaluate the effectiveness of the framework in terms of security and privacy by performing experiments on some concrete examples. We also provide two instantiations of the framework with two different secure aggregation protocols to give a more concrete view how the framework works and we analyse the computation and communication overhead of the framework.

Keywords— federated learning, secure aggregation, privacy enhancing technologies, poisoning attacks

1 Introduction

In the field of artificial intelligence and machine learning, federated learning (FL) has become a cutting-edge method, providing the possibility of collaborative model training while keeping data local, without the need for central data aggregation. With these promising advantages, it has been welcomed in many domains such as mobile telecommunication networks as seen in standardization specifications of 5G (1) and research in 6G (2). Nevertheless, despite its significant advances in data privacy protection, it still faces challenges in maintaining robust security measures while upholding privacy

[*]ferhat.karakoc@ericsson.com
[†]betul.guvenc.paltun@ericsson.com
[‡]leyli.karacay@ericsson.com
[§]omer.tuna@ericsson.com
[¶]ramin.fuladi@ericsson.com
[‖]utku.gulen@ericsson.com

principles depending on the use case. In order to address these issues, a novel concept has been emerged for security-friendly privacy solution tailored for federated learning environments.

FL operates as a privacy-aware collaborative machine learning technique, where the server and clients collaborate in constructing a global ML model. The clients, possessing local data, engage in local training and subsequently send local model updates to the server, rather than disclosing raw data. The server aggregates these updates to refine the global model iteratively until convergence is achieved. While FL inherently preserves privacy by not transmitting raw data, concerns arise regarding potential information leakage from local model updates sent to the server (3). To address these privacy concerns, secure aggregation methods such as masking, homomorphic encryption, and functional encryption are employed to conceal individual local model updates from the server while still allowing the aggregation of results. However, while these methods enhance privacy, they introduce challenges in preventing security attacks such as backdoor and poisoning attacks. The server's inability to directly analyze individual local model updates hinders the detection of malicious behaviors that could compromise the integrity of the global model. Alternative solutions, including trusted execution environments and secure multi-party computation, offer avenues for executing secure attack detection analysis without directly accessing individual local model updates. However, these approaches present drawbacks, such as computational overhead and trust assumptions, which may limit their scalability and practicality.

In this study, we propose a framework to increase privacy in FL also allowing execution of security attack detection and prevention mechanisms against the FL model training operation. This framework allows the server to access some pieces of the individual local model updates which allows the server to execute some analysis to detect and even mitigate security attacks against the FL execution. In (4), it is shown that analysis of some pieces of the local model updates may be enough to detect security attacks. Our framework also keeps the other parts of the individual local model updates hidden from the server, which is for addressing privacy aspects. To prevent malicious behaviors of the clients such as using only hidden parts to execute security attacks, it is proposed that the server decides at runtime which pieces should be cleartext so that the clients cannot guess which pieces of their local model updates will be hidden beforehand. This decision and request to open some pieces is called *challenging* the clients. The framework also includes a step for verification of the challenge response from the clients to prevent cheating such as changing the values of the challenged pieces. Overview of our solution can be summarized as follows.

- Instead of sending the local model updates in encrypted/masked format as a one block, the clients first divide their local model updates into pieces (e.g., weights of subsequent layers of a DNN) and then encrypt/mask the pieces separately and send the encrypted/masked pieces to the server.
- After receiving the encrypted/masked local model update pieces, the server challenges the clients to open some pieces of the local model updates. For this aim, the server sends clients the indexes of the pieces of the local model updates to be opened. The size of the index set can be limited so that the server cannot request to open huge part of the local model updates.
- The clients send unencrypted/unmasked (i.e., cleartext) local model update pieces to the server with related parameters used in the encryption/masking operation. Using the received cleartext local model update pieces and the parameters, the server can ensure that the challenged local model update pieces are the encryption/masking of the received cleartext local model update pieces. Also, the server can execute some security attack detection analyses on the received cleartext local model update pieces.

The proposed solution protects the privacy in FL while also allowing the server to

detect security attacks coming from clients. It does not require any additional trust assumptions such as non-colluding servers, does not bring any heavy computational overhead, and does not require any communication need between clients, which are most common approaches in the literature as seen in the following section.

The paper is organized as follow. We present the related work and preliminaries in Section 2 and Section 3, respectively. We introduce our framework in Section 4 and provide two instantiations of it with two well-known secure aggregation schemes. We analyze the security of the framework in Section 5 by listing possible malicious behaviors of clients and server and by showing how these malicious behaviors can be prevented by our framework. Section 6 presents preliminary analysis on how our framework enables detection of attacks against the FL process without violating privacy. Section 7 concludes the paper by pointing out possible future work.

2 Related work

Although addressing security and privacy in federated learning separately have been studied intensively in the literature, there are only a few studies that address both aspects simultaneously. We collect these studies in the following categories and briefly explain these solutions.

- Two non-colluding server based solutions
- Client-assisted based solutions
- Zero-knowledge-proof based solutions
- Secure comparison based solutions
- Multip-hop communication based solutions

Two non-colluding server based solutions. Prio (5) and ELSA (6) are two solutions that require two servers which are expected to not collude. Since it is assumed that these servers do not collaborate for a malicious purpose, it has to be assumed that at least one of the servers can be a semi-honest adversary at most (i.e., at least one of the servers should not behave as a malicious adversary). A semi-honest adversary is defined as an adversary that follows the protocol steps and tries to learn more information than the information learned from the output of the protocol, by using the messages received during the execution of the protocol while a malicious is not expected to execute the protocol steps as defined. Prio uses secret sharing and secret-shared non-interactive proofs in their solution, so that the servers can only access the secret shares of the local model updates. Because of that reason a non-colluding two server assumption is needed, otherwise the servers can collaborate with each other to construct the local model updates from the secret shares. The servers execute a two-party computation protocol to perform some computation on the secret shares of the local model updates in a collaborative way. ELSA is also similar to Prio in terms of the need of two non-colluding servers and usage of secret sharing approach. The servers calculate a norm bound collaboratively by using a two-party secure computation protocol for detection of security attacks.

Client-assisted based solutions. Similar to Prio and ELSA, (7) also utilizes secret shares and as done in ELSA norm bounds are computed to detect security attacks. The difference is that instead of the need of having two non-colluding servers, the clients exchange secret shares of their local model updates among themselves. Another study that requires only one server is Flamingo (8) which needs a small group of clients called "encryptors" which collaborate with the server.

Zero-knowledge-proof based solutions. Some of the solutions in the literature do not need any collaboration between two non-colluding servers, between clients, or between clients and servers, but proposes running of a zero-knowledge proof for ensuring that there is no security attacks to the FL process. One of such solutions is Rofl (9) whose concrete complexity is high due to usage of zero-konwledge proofs, compared to the two-server based solutions. Another study (10) introduced EIFFeL (Ensuring Integrity For Federated Learning) which is another example of zero-knowledge based secure and privacy enhanced FL category. In that method, the server can also remove the suspicious local model updates from the aggregation. Due to the zero-knowledge proofs usage need, similar performance drawback becomes also visible in EIFFel. The solution zPROBE (11) is also utilizing zero-knowledge proofs to make the FL process robust against security attacks.

Secure comparison based solutions. Although zero-knowledge proofs are very useful against malicious clients, they also bring considerable performance overhead. Instead of using a zero-knowledge approach, Karakoc et al. (4) utilized a secure two-party computation protocol which allows the server to ensure that the local model update weights are in a predefined range, without allowing the server to access the weights. For that purpose, the idea in PUDA (12) solution is utilized and an oblivious programmable pseudo random function is constructed which allows the clients to learn a valid integrity check tag only when their local modal update weights are in the predefined range. Then the server validates the aggregated integrity check tag to ensure that all the weights in the individual local model updates are acceptable values.

Multip-hop communication based solutions. Another approach to address security and privacy simultaneously is to anonymize the ownership of the local model updates and let the server access the cleartext local model updates. This method is especially promising in the scenarios where it is enough to break the linkage between the local model update and its owner, and the clients can communicate with each other. One example in this category was proposed in (13) which proposes to use multi-hop communication to hide the identifier of the owner of the local model update from the server. It also utilizes a reputation and incentive based approach to be used in the multi-hop communication which may include some malicious clients in the middle of the communication. Similarly, a multi-hop communication-based solution was proposed in (14; 15), but in that study partially blind signatures are used to prevent malicious behaviors of the clients such as sending the same local model updates multiple times. In both proposals, the server can access the individual local model updates in cleartext, so it can execute some analysis to detect and prevent security attacks.

3 Preliminaries

We give brief definitions of two well-known secure aggregation schemes that we use for two concrete instantiation of our framework, which are presented in Section 4.2 and 4.3. The reason why we prefer to present two instantiations of the framework with two secure aggregation scheme is to show its flexibility and being independent from the aggregation scheme (i.e., can work with any secure aggregation scheme), which can be seen from the similarity of the instantiations.

3.1 Joye-Libert scheme

This scheme introduced in (16) is an additively homomorphic encryption schemes consisting of three algorithms:

- $Setup(\lambda) \rightarrow (pp, \{k_i\}_{i\in[n]}, k_0)$. This algorithm generates necessary parameters for the execution of the scheme. It takes the security parameter λ as input and generates randomly selected n keys $\{k_i\}_{i\in[n]}$, a decryption key $k_0 = -\sum_1^n k_i$ and public parameter $pp = (N, H)$ where N is the product of two large prime numbers and H is a cryptographic hash function.
- $Enc(pp, k_i, t, x_{i,t}) \rightarrow c_{i,t}$. The i-th client holding the encryption key k_i and private input $x_{i,t}$, computes the ciphertext at time period t by using the public parameter as follow.

$$c_{i,t} \leftarrow (1 + x_{i,t}N)H(t)^{k_i} \mod N^2$$

- $Agg(pp, k_0, t, \{c_{i,t}\}_{i\in[n]}) \rightarrow \sum_1^n x_{i,t}$. The aggregator aggregates the ciphertexts received from n clients and then decrypts the aggregated result by using the decryption key k_0 as follow.

$$c_t \leftarrow \Pi_1^n c_{i,t} = (1 + N\sum_1^n x_{i,t})H(t)^{\sum_1^n k_i}$$

$$\sum_1^n x_{i,t} \leftarrow (H(t)^{k_0} c_t - 1)N^{-1} \mod N$$

3.2 Shi-Chan-Rieffel-Chow-Song Scheme

This scheme introduced in (17) is based on discrete logarithm problem and consists of the following algorithms.

- $Setup(\lambda) \rightarrow (\mathbb{G}, H, \{k_i\}_{i\in[n]}, k_0)$. This algorithm generates necessary parameters. It takes the security parameter (λ) as input and generates randomly selected n keys $\{k_i\}_{i\in[n]}$, a decryption key $k_0 = -\sum_1^n k_i$ and public parameters $\mathbb{G}$ and H where $\mathbb{G}$ is a group of large prime order with a generator g and H is a hash function such that $H : \{0,1\}^* \rightarrow \mathbb{G}$.
- $Enc(\mathbb{G}, H, k_i, t, x_{i,t}) \rightarrow c_{i,t}$. The i-th client holding the encryption key k_i and private input $x_{i,t}$, computes the ciphertext at time period t by using the public parameters as follow.

$$c_{i,t} \leftarrow H(t)^{k_i} g^{x_{i,t}}$$

- $Agg(\mathbb{G}, H, k_0, t, \{c_{i,t}\}_{i\in[n]}) \rightarrow \sum_1^n x_{i,t}$. The aggregator aggregates the ciphertexts received from n clients to computes $c_t = g^{\sum_1^n x_{i,t}}$ by using the decryption key k_0 as follow and then learns $\sum_1^n x_{i,t}$ by computing the discrete logarithm of c_t.

$$c_t \leftarrow \Pi_1^n c_{i,t} H(t)^{k_0} = H(t)^{\sum_1^n k_i} g^{\sum_1^n x_{i,t}} H(t)^{-\sum_1^n k_i} = g^{\sum_1^n x_{i,t}}$$

4 Proposed framework

4.1 Definition of the framework

Figure 1 depicts our solution in a high level and Protocol 1 defines the proposed framework.

In Step 2 of the protocol, the encryption can be an additively homomorphic encryption where a random parameter $r_{i,j}$ and the public key pk_i of C_i are used. In the masking operation a mask value $k_{i,j}$ is used. In the decision of which clients should be challenged in Step 5, the server can determine the clients randomly, or it can use previous security analysis results and some other suspicious behaviors of the clients.

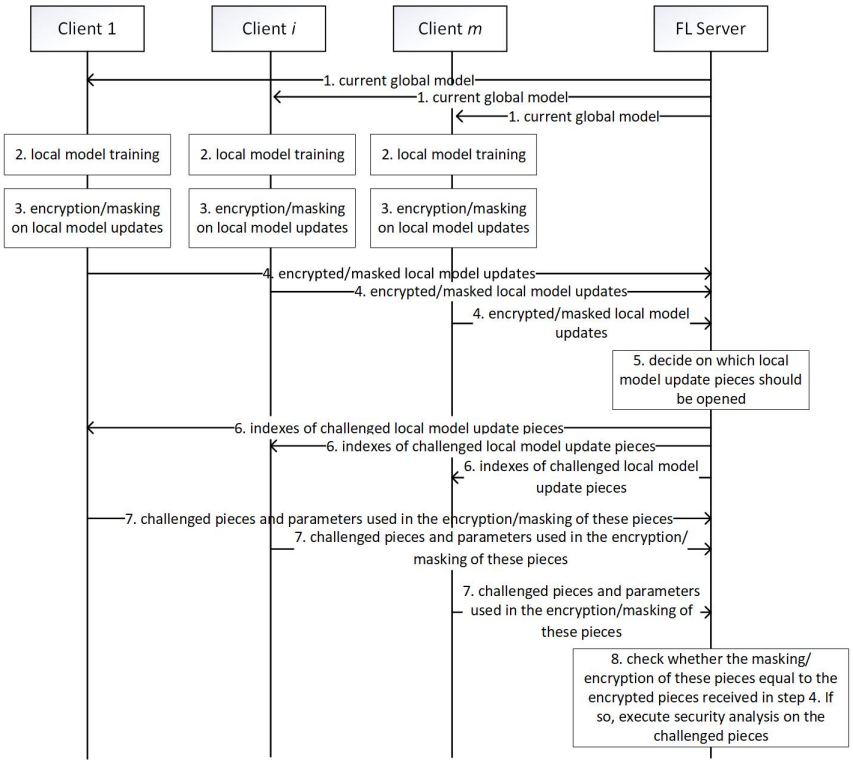

Figure 1: High-level procedure flow of our framework.

Protocol 1 Our challenged-based framework enhancing security and privacy in FL

Parameters. The clients and server agree on a secure aggregation scheme such as Joye-Libert (16) and Shi-Chan-Rieffel-Chow-Song Scheme (17).
Inputs. Each client $\mathcal{C}_i$ inputs local training data and the FL server $\mathcal{S}$ inputs an initial global machine learning model.
Outputs. $\mathcal{S}$ outputs the trained global machine learning model
Protocol steps:

1. The FL server sends the current global model to the clients.
2. The clients perform local model training using the received global model and local data.
3. Each client ($\mathcal{C}_i$) divides the local model update into small pieces (where j-th piece is denoted by $d_{i,j}$) and performs encryption or masking operation on the pieces (the encrypted/masked j-th piece is denoted by $\{E(d_{i,j})\}$.
4. The clients send the encrypted/masked local model update pieces to the server.
5. The server first decides which clients should be challenged.
6. The server sends the indexes of the challenged pieces to the clients to be challenged.
7. The challenged clients send the challenged local model update pieces in cleartext and send the related parameters (such as the random parameter used in the encryption, public key if it is not known by the server, mask value) to the server.
8. The server validates the responses. If the validation check is successful, then the server executes security analysis on the cleartext local model update pieces to identify the local model updates pieces that look abnormal. If no abnormal local model update piece is found, then the server updates the global model using the local model updates and the procedure continue with step 1.

Similarly, in the decision of which pieces of the local model updates are to be challenged, the server randomly decides on the indexes of the pieces to be challenged. In the decision, the server can use previous security analysis results and some other suspicious behaviors of the clients. The server can also take the communication or other constraints into account on selecting the indexes with smaller layers. Before executing Step 7, clients can communicate with each other to check that the received indexes of challenged pieces are same and the number of challanged pieces is acceptable. In Step 8, to validate the challenge responses, the server can perform encryption/masking operation using the received cleartext pieces and related parameters and compares encrypted/masked challenged pieces with the ones received in Step 4. Depending on the encryption scheme, this check may not be enough to detect if the client has changed the challenged local model update pieces because the client can be able to generate the same encryption result for a different value of the challenged local model update piece by using another encryption/mask key. For that case, the server can compare the aggregation result computed on the encrypted/mask values with the aggregation result computed on the opened values. For detection of security attacks in Step 8, the server can inspect the correlation between the local model updates of the clients using the clear text local model update pieces of clients and search for anomaly or deviations. Another example for detection of anomalies in the local model pieces can be the usage of vector norms as applied in ELSA paper (6). If the server detect a cheating client, then it can take appropriate actions against the cheating client and can continue to

the FL process without using the cheating client's local model updates.

Note that it is assumed that traditional secure communication solutions is used to protect the communication between the clients and server so that the challenges, responses and cleartext local model update pieces are protected against adversaries sitting between the clients and servers.

4.2 Realization of the framework with Joye-Libert scheme

Protocol 2 presents how our framework can be used with the Joye Libert encryption scheme (16).

4.3 Realization of the framework with Shi-Chan-Rieffel-Show-Song scheme

Protocol 3 presents how our framework can be used with the Shi-Chan-Rieffel-Show-Song scheme (17).

5 On the security of the framework

This section focuses on the security of the framework while the performance of the framework on enhancing security and privacy for the federated learning is analyzed in the following section. The additional steps proposed by our framework on top of FL are the steps 3, 5, 6, 7, and 8 of Protocol 1, so these steps are analyzed in this section.

Security against malicious server. The server executes steps 5 and 6 to challenge the clients to open some pieces of the local model updates. Instead of sending the same index set I_d, the server can try to send different index sets to the clients to extract whole local model updates instead of only some pieces. To prevent such kind of malicious behavior of the server, the clients can communicate with each other to check that the I_d is same for all the clients. Another malicious behavior of the server can be that the server can try to increase the size of the I_d to learn more than the expected number of local model update pieces which may result in learning information about the training data, but this behavior can easily be detected by the clients.

Security against malicious clients. Since the clients do not know which local model update pieces will be challenged by the server, the clients cannot select which local model updates to use for poisoning. The client can try and be able to escape in some rounds but the success probability of not being detected by the server will decrease exponentially in the number of FL rounds. Another malicious behavior of the clients can be to change the values of the challenged local model update pieces. When a client changes the value, the change will be detected by the server because the server can perform encryption/masking operation using the received cleartext pieces and related parameters and compares computed encrypted/masked challenged pieces with the received encrypted/masked challenged pieces. However, depending on the encryption scheme, this check may not be enough if the client can generate the same encryption result for a different value of the challenged local model update piece by using another encryption/mask key. For that type of encryption schemes, the change can be detected by the server by comparing the aggregation result on encrypted local model update piece with the aggregation result on the cleartext local model update piece. To be able to perform that check, the server needs to challenge all the clients. Several clients may collaborate to escape from such detection such as one client has sent the encryption result of a local update piece which is not in the predetermined internal, but when it is challenged then the client can collaborate with another client

Protocol 2 Instantiation of our challenged-based framework with Joye-Libert scheme for enhancing security and privacy in FL

Parameters. Public parameter N and hash function H.
Inputs. Each client $\mathcal{C}_i$ inputs local training data and the set of encryption key $\{k_{i,j,t}\}$ where $k_{i,j,t}$ is the key used by the i-th client to encrypt j-th local model update piece in t-th round of FL. The FL server $\mathcal{S}$ inputs an initial global machine learning model and a decryption key $k_{0,j,t}$ where $k_{0,j,t} = -\sum_{i=1}^{n} k_{i,j,t}$.
Outputs. $\mathcal{S}$ outputs the trained global machine learning model
Protocol steps:

1. The FL server sends the current global model to the clients.
2. The clients perform local model training using the received global model and local data.
3. Each client ($\mathcal{C}_i$) divides the local model update into small pieces (where j-th piece is denoted by $d_{i,j}$) and performs the following encryption operation. $H(j,t)$ is the computation of hash on the parameters j and t.

$$c_{i,j,t} \leftarrow (1 + d_{i,j,t}N)H(j,t)^{k_{i,j,t}} \mod N^2$$

4. The clients send the encrypted local model update pieces ($\{c_{i,j,t}\}$) to the server.
5. The server first decides which clients should be challenged. Let I_c denotes the set of identifiers of the clients challenged by the server.
6. The server sends the indexes of the challenged pieces to the clients to be challenged. Let I_d denotes the index set consisting of the indexes of the local model update pieces challenged by the server.
7. The challenged clients ($\{C_i \mid i \in I_c\}$) send the challenged local model update pieces in cleartext and the encryption keys (i.e., $\{(d'_{i,j,t}, k'_{i,j,t}) \mid j \in I_d\}$) to the server.
8. The server computes $c'_{i,j,t} \leftarrow (1 + d'_{i,j,t}N)H(j,t)^{k'_{i,j,t}} \mod N^2$ for $i \in I_c$ and $j \in I_d$ and checks if $c'_{i,j,t} = c_{i,j,t}$. If successful, also computes

$$c_{.,j,t} \leftarrow \Pi_1^n c_{i,j,t}$$

$$\sum_{1}^{n} d_{i,j,t} \leftarrow (H(j,t)^{k_{0,j,t}} c_{.,j,t} - 1)N^{-1} \mod N$$

for all j, and checks if $\sum_1^n d'_{i,j,t}$ equals to $\sum_1^n d_{i,j,t}$ for $j \in I_d$. If the validation check is successful, then the server executes security analysis on the cleartext local model update pieces to identify the local model updates pieces that look abnormal. If no abnormal local model update piece is found, then the server updates the global model using the local model updates and the procedure continue with step 1.

to request the other client to send a value less than the value which has already been sent to the server. As a result, at the end the aggregation result will be same, meaning that there is no local update pieces outside the predetermined interval.

Protocol 3 Instantiation of our challenged-based framework with Shi-Chan-Rieffel-Show-Song scheme for enhancing security and privacy in FL

Parameters. A group $\mathbb{G}$ with the generator g and a hash function H.
Inputs. Each client $\mathcal{C}_i$ inputs local training data and the set of encryption key $\{k_{i,j,t}\}$ where $k_{i,j,t}$ is the key used by the i-th client to encrypt j-th local model update piece in t-th round of FL. The FL server $\mathcal{S}$ inputs an initial global machine learning model and a decryption key $k_{0,j,t}$ where $k_{0,j,t} = -\sum_{i=1}^{n} k_{i,j,t}$.
Outputs. $\mathcal{S}$ outputs the trained global machine learning model
Protocol steps:

1. The FL server sends the current global model to the clients.
2. The clients perform local model training using the received global model and local data.
3. Each client ($\mathcal{C}_i$) divides the local model update into small pieces (where j-th piece is denoted by $d_{i,j}$) and performs the following encryption operation. $H(j,t)$ is the computation of hash on the parameters j and t.

$$c_{i,j,t} \leftarrow H(j,t)^{k_{i,j,t}} g^{d_{i,j,t}}$$

4. The clients send the encrypted local model update pieces ($\{c_{i,j,t}\}$) to the server.
5. The server first decides which clients should be challenged. Let I_c denotes the set of identifiers of the clients challenged by the server.
6. The server sends the indexes of the challenged pieces to the clients to be challenged. Let I_d denotes the index set consisting of the indexes of the local model update pieces challenged by the server.
7. The challenged clients ($\{C_i \mid i \in I_c\}$) send the challenged local model update pieces in cleartext and the encryption keys (i.e., $\{(d'_{i,j,t}, k'_{i,j,t}) \mid j \in I_d\}$) to the server.
8. The server computes $c'_{i,j,t} \leftarrow H(j,t)^{k_{i,j,t}} g^{d'_{i,j,t}}$ for $i \in I_c$ and $j \in I_d$ and checks if $c'_{i,j,t} = c_{i,j,t}$. If successful, also computes

$$c_{.,j,t} \leftarrow \Pi_1^n c_{i,j,t} H(j,t)^{k_0}$$

and computes $\sum_1^n d_{i,j,t}$ for all j, and checks if $\sum_1^n d_{i,j,t}$ equals to $\sum_1^n d'_{i,j,t}$ for $j \in I_d$. If the validation check is successful, then the server executes security analysis on the cleartext local model update pieces to identify the local model updates pieces that look abnormal. If no abnormal local model update piece is found, then the server updates the global model using the local model updates and the procedure continue with step 1.

6 Experimental results on security and privacy benefits and analysis on complexity overhead

6.1 Prevention of privacy attacks

In this section we focus on the deep leakage from gradient (DLG) attack which is a term used in the context of privacy in machine learning models. It refers to a scenario where an attacker can extract sensitive information about the training data used to train a machine learning model by analyzing the gradients of the model throughout the training process. This section is dedicated to the experiments related to the application

Table 1: Number of iterations and loss value for deep leakage from gradients when the server access whole individual whole local model updates.

iteration	loss
0	64.42
10	0.75
20	0.060
30	0.0115
40	0.0028
50	0.0010
60	0.0004

iteration	loss
70	0.0002
80	0.0001
90	0.0001
100	0
110	0
120	0
130	0
140	0

of DLG attack (18) on federated learning within our framework, which allows the server to access some parts of the local model updates, and without any secure aggregation, which means that the server access the whole local model updates. Sharing gradients is a commonly employed technique in contemporary multi-node machine learning setups, such as distributed training and collaborative learning. Traditionally, there was a widespread belief that sharing gradients was secure and wouldn't expose the training data. However, the work (18) demonstrates that it is feasible to extract private training data from publicly shared gradients. For our experiments we use the same settings in (18) and choose PyTorch as our experiment platform. In the experiments, the LeNet, which is a convolutional neural network, introduced by Yann LeCun (19) is used. We conduct our experiments by modifying some elements of the gradient matrix and evaluate whether an attacker is still able to generate a training data from the gradients. Figure 2 demonstrates deep leakage from gradients which are shared throughout deep learning process. The attacker successes to create the ground truth image from the shared gradients starting from random init image. Table 1 shows the iteration number and loss ratio on predicting the ground truth. As it is shown, from iteration 100 the loss value for prediction is zero and the attacker with high confident can predict the training data.

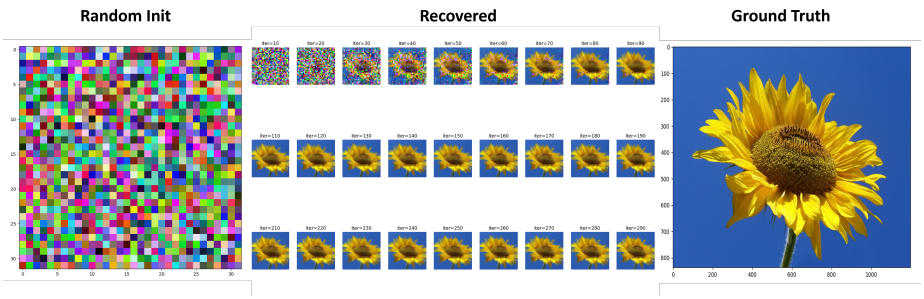

Figure 2: Deep leakage from gradients on an image when the server access whole individual local model updates.

In Figure 3, we illustrate that when modifying some elements of the gradient matrix to zeros (i.e., only allowing the server to access some pieces of the individual local model updates), the attacker is not able to construct the ground truth from random init image. As a result, using our proposed framework with a secure aggregation scheme, if some of the gradients which are exposed at specific neural network layer are encrypted and not reachable, then the attacker is not able to construct the training data and learn sensitive information. In Table 2 , it is shown that even at iteration 200, the attacker still could not predict the training data and the predicted image is

Table 2: Iteration and loss value for deep leakage from gradients after having partially secure transmittion of the gradients

iteration	loss
0	64.42
10	0.76
20	0.131
30	0.068
40	0.0509
50	0.0445
60	0.0421
70	0.0409
80	0.0403
90	0.0398

iteration	loss
100	0.0395
110	0.0392
120	0.0390
130	0.0388
140	0.0387
150	0.0387
160	0.0386
170	0.0386
180	0.0386
190	0.0386
200	0.0385

random init image which does not leak any information regarding to the training data.

Figure 3: Deep leakage from gradients on an image when partially secure aggregation is applied on some of the gradients

6.2 Prevention of security attacks

To test whether accessing some pieces of local model updates can be enough for detection of security attacks, we simulated the setup with nine honest clients and one malicious client who adds noise in gaussian distribution to its local model update. The convolutional neural network consists of four convolutional layers and three linear layers. MNIST dataset is used, which consists of 60.000 training and 10.000 test images of handwritten digits (0-9), with each image being 28x28 pixels. From MNIST dataset, 10 training dataset each consisting of 20.000 images and 10 test dataset each consisting of of 3000 images are generated randomly and each client trains a local model on its own piece and sends the model updates (gradients or weights) to the server. The 5-th client is selected as the malicious client who directly adds noise in gaussian distribution to its local model update before sending it to the server. Only second linear layer weights are revealed to the server and the server checks the distance between the weights vector of second linear layer. The distance of the weight vector of the i-th client to the weight vectors of other clients is computed as follow

$$\sum_{j=1}^{10} \sum_{l=1}^{3200} |T_i[l] - T_j[l]|$$

where the total number of weights at second linear layer is 3200, there are 10 clients and $T_i[l]$ is the l-th weight in the second linear layer of the local model updates of the

i-th client. As it is shown in the Figure 4, the distance computed for the 5-th client which is the malicious client was easily distinguished.

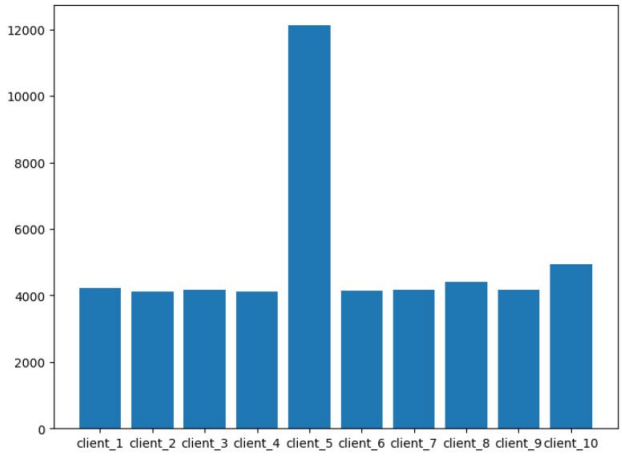

Figure 4: Distance between the local model updates from 10 clients

6.3 Computation and communication overhead

The additional steps introduced by our framework on top of a federated learning with secure aggregation are steps 5, 6, 7, and 8 shown in Protocol 1. There is no computation and communication cost of step 5. For step 6, the server needs to send the indexes of the challenged local model update pieces to the client, but the communication cost of transmitting this index set can be negligible when compared to the communication cost of transferring local model updates. In step 7, the clients are required to send the challenged local model update pieces and required parameters for encryption. The communication cost is also not dominant when sending the local model updates is considered because in this step, only some pieces of the local model updates need to be sent. In step 8, the server needs to execute encryption for the challenged local model update pieces which can be considered as the main overhead in terms of computation in the server side. For the communication overhead, there is no considerable overhead in terms of amount of data to be transferred, but the one main overhead is the requirement of one additional communication round between the clients and server. While in the federated learning with secure aggregation the number of communication rounds equals to two times of the number of federated learning rounds, in our framework it duplicates the number of communication rounds, which can be acceptable when the benefits of the framework satisfying both the privacy and security aspects simultaneously without requiring additional heavy cryptographic operations such as zero knowledge proofs and additional assumptions such as two non-colluding servers.

7 Conclusion

We proposed a framework to make federated learning robust (against security attacks) and privacy enhanced (not to leak information about the training data) simultaneously. The framework can be used with any secure aggregation schemes without introducing any considerable computation and communication overhead. We presented to concrete examples of instantiation of the framework with two well knows secure aggregation schemes. We also conducted some experiments to test the effectiveness of

the framework against security and privacy attacks. These experiments can be seen as preliminary studies to evaluate robustness and privacy increase. More studies can be initiated by taking the observations we shared in this paper as a starting point, such as analyzing the effect of percentage of opened local model update pieces to the privacy and security attacks, performing experiments against other privacy attack techniques and security attack techniques.

Acknowledgments

This work was supported by the Scientific and Technological Research Council of Turkey (TUBITAK) through the 1515 Frontier Research and Development Laboratories Support Program under Project 5169902, and has been partly funded by the Hexa-X II project which has received funding from the Smart Networks and Services Joint Undertaking (SNS JU) under the European Union's Horizon Europe research and innovation program and Grant Agreement No 101095759.

References

[1] G. Specification., "TS 23.288 v18.5.0: Architecture enhancements for 5G System (5GS) to support network data analytics services," 2024.

[2] D. Sirohi, N. Kumar, P. Singh Rana, R. Tanwar, R. Iqbal, and M. Hijjii, "Federated learning for 6G-enabled secure communication systems: a comprehensive survey," *Artif Intell Rev*, vol. 56, 2023.

[3] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *IEEE S&P*, 2017.

[4] F. Karakoç, M. Önen, and Z. Bilgin, "Secure aggregation against malicious users," in *SACMAT '21: The 26th ACM Symposium on Access Control Models and Technologies, Virtual Event, Spain, June 16-18, 2021*, J. Lobo, R. D. Pietro, O. Chowdhury, and H. Hu, Eds. ACM, 2021, pp. 115–124. [Online]. Available: https://doi.org/10.1145/3450569.3463572

[5] H. Corrigan-Gibbs and D. Boneh, "Prio: Private, robust, and scalable computation of aggregate statistics," in *14th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2017, Boston, MA, USA, March 27-29, 2017*, A. Akella and J. Howell, Eds. USENIX Association, 2017, pp. 259–282. [Online]. Available: https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/corrigan-gibbs

[6] M. Rathee, C. Shen, S. Wagh, and R. A. Popa, "ELSA: secure aggregation for federated learning with malicious actors," in *44th IEEE Symposium on Security and Privacy, SP 2023, San Francisco, CA, USA, May 21-25, 2023*. IEEE, 2023, pp. 1961–1979. [Online]. Available: https://doi.org/10.1109/SP46215.2023.10179468

[7] H. Masuda, K. Kita, Y. Koizumi, J. Takemasa, and T. Hasegawa, "Byzantine-resilient secure federated learning on low-bandwidth networks," *IEEE Access*, vol. 11, pp. 51 754–51 766, 2023.

[8] Y. Ma, J. Woods, S. Angel, A. Polychroniadou, and T. Rabin, "Flamingo: Multi-round single-server secure aggregation with applications to private federated learning," Cryptology ePrint Archive, Paper 2023/486, 2023, https://eprint.iacr.org/2023/486. [Online]. Available: https://eprint.iacr.org/2023/486

[9] H. Lycklama, L. Burkhalter, A. Viand, N. Küchler, and A. Hithnawi, "Rofl: Robustness of secure federated learning," in *44th IEEE Symposium on Security and Privacy, SP 2023, San Francisco, CA, USA, May 21-25, 2023*. IEEE, 2023, pp. 453–476. [Online]. Available: https://doi.org/10.1109/SP46215.2023.10179400

[10] A. R. Chowdhury, C. Guo, S. Jha, and L. van der Maaten, "Eiffel: Ensuring integrity for federated learning," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*, H. Yin, A. Stavrou, C. Cremers, and E. Shi, Eds. ACM, 2022, pp. 2535–2549. [Online]. Available: https://doi.org/10.1145/3548606.3560611

[11] Z. Ghodsi, M. Javaheripi, N. Sheybani, X. Zhang, K. Huang, and F. Koushanfar, "zprobe: Zero peek robustness checks for federated learning," in *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*. IEEE, 2023, pp. 4837–4847. [Online]. Available: https://doi.org/10.1109/ICCV51070.2023.00448

[12] I. Leontiadis, K. Elkhiyaoui, M. Önen, and R. Molva, "PUDA - privacy and unforgeability for data aggregation," in *CANS*, 2015.

[13] J. Domingo-Ferrer, A. Blanco-Justicia, J. A. Manjón, and D. Sánchez, "Secure and privacy-preserving federated learning via co-utility," *IEEE Internet Things J.*, vol. 9, no. 5, pp. 3988–4000, 2022. [Online]. Available: https://doi.org/10.1109/JIOT.2021.3102155

[14] F. Karakoç, L. Karaçay, P. Çomak, U. Gülen, R. Fuladi, and E. U. Soykan, "A security-friendly privacy solution for federated learning," in *Short Paper Proceedings of the First International Workshop on Artificial Intelligence in Beyond 5G and 6G Wireless Networks (AI6G 2022) co-located with IEEE World Congress on Computational Intelligence (WCCI2022), Padova, Italy, July 21, 2022*, ser. CEUR Workshop Proceedings, A. Renda, P. Ducange, T. Borsos, and H. Flinck, Eds., vol. 3189. CEUR-WS.org, 2022. [Online]. Available: https://ceur-ws.org/Vol-3189/paper_08.pdf

[15] F. Karakoç, L. Karaçay, P. Ç. D. Cnudde, U. Gülen, R. Fuladi, and E. U. Soykan, "A security-friendly privacy-preserving solution for federated learning," *Comput. Commun.*, vol. 207, pp. 27–35, 2023. [Online]. Available: https://doi.org/10.1016/j.comcom.2023.05.004

[16] M. Joye and B. Libert, "A scalable scheme for privacy-preserving aggregation of time-series data," in *FC*, 2013.

[17] E. Shi, T. H. Chan, E. G. Rieffel, R. Chow, and D. Song, "Privacy-preserving aggregation of time-series data," in *NDSS*, 2011.

[18] L. Zhu, Z. Liu, and S. Han, "s," *Advances in neural information processing systems*, vol. 32, 2019.

[19] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.