Information Sciences 522 (2020) 69–79

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

A training-integrity privacy-preserving federated learning scheme with trusted execution environment

Yu Chen[a], Fang Luo[a], Tong Li[a,*], Tao Xiang[b], Zheli Liu[c], Jin Li[a,*]

[a] School of Computer Science and Cyber Engineering, Guangzhou University, Guangzhou, China
[b] College of Computer Science, Chongqing University, Chongqing, China
[c] College of Cyber Science and the College of Computer Science, Nankai University, Tianjin, China

ARTICLE INFO

Article history:
Received 6 November 2019
Revised 10 January 2020
Accepted 12 February 2020
Available online 12 February 2020

Keywords:
Federated learning
Data integrity
Privacy preserving
Trusted execution environment

ABSTRACT

Machine learning models trained on sensitive real-world data promise improvements to everything from medical screening to disease outbreak discovery. In many application domains, learning participants would benefit from pooling their private datasets, training precise machine learning models on the aggregate data, and sharing the profits of using these models. Considering privacy and security concerns often prevent participants from contributing sensitive data for training, researchers proposed several techniques to achieve data privacy in federated learning systems. However, such techniques are susceptible to causative attacks, whereby malicious participants can inject false training results with the aim of corrupting the well-learned model. To end this, in this paper, we propose a new privacy-preserving federated learning scheme that guarantees the integrity of deep learning processes. Based on the Trusted Execution Environment (TEE), we design a training-integrity protocol for this scheme, in which causative attacks can be detected. Thus, each participant is compelled to execute the privacy-preserving learning algorithm of the scheme correctly. We evaluate the performance of our scheme by prototype implementations. The experimental result shows that the scheme is training-integrity and practical.

1. Introduction

Machine learning models trained on sensitive real-world data promise improvements to everything from medical screening to disease outbreak discovery. The widespread use of mobile devices means even richer and more sensitive data is becoming available. Companies such as Google, Facebook, and Apple take advantage of the massive amounts of training data collected from their users and the vast computational power of GPU farms to deploy deep learning on a large scale. Thus, data owners, such as the companies above, are willing to collect complementary data and collaboratively train a deep neural network model on the joint of these data. The unprecedented accuracy of the resulting model allows them to be used as the foundation of many new services and applications, including accurate speech recognition and image recognition that outperforms humans.

However, the large-scale training on sensitive data entails risks. The data being classified or used for training is often sensitive and may come from multiple sources with different privacy requirements. Laws, user privacy concerns, data

* Corresponding authors.
E-mail addresses: 860887839@qq.com (Y. Chen), 1299341603@qq.com (F. Luo), litongziyi@mail.nankai.edu.cn (T. Li), txiang@cqu.edu.cn (T. Xiang), liuzheli@nankai.edu.cn (Z. Liu), 22547169@qq.com (J. Li).

https://doi.org/10.1016/j.ins.2020.02.037
0020-0255/© 2020 Elsevier Inc. All rights reserved.

sovereignty issues, and competitive advantage are all reasons that prevent entities from pooling different data sources to train more accurate models.

Aiming at addressing these privacy concerns, some previous works introduced privacy-preserving machine learning scheme that can help to protect sensitive information in each learning participant's training dataset. Some of these are strong guarantees that, though do not provide a full-proof privacy solution [46,47,49], a practical solution which can be expanded and strengthened when combined with orthogonal mechanisms [1,20,33,34]. Moreover, the federated Learning setting [45] are proposed in recently years, wherein each participant maintains a private database and a shared global model is trained under the coordination of a central server based upon highly processed, minimally scoped, ephemeral updates from participants. All these work can provide a strong first line of defense against any semi-honest participant who only wants to reveal sensitive information from other participants' dataset.

Nevertheless, there is still a large gap between the existed privacy-preserving solutions and the practical setting. In the federated learning setting, participants can now observe the intermediate model and contribute arbitrary updates as part of the training process. This creates an opportunity for active participants to manipulate the training process with little restriction. On one hand, adversaries posing as benign participants can send manipulated updates that maliciously influence the properties of the trained model, which is known as causative attacks. On the other hand, a lazy participant can lie about the number of data record he/she uses for training, which makes the federated learning end up with a worse model.

The effective mitigation of these model-tampered threats is to ensure the integrity of local training processes in the federated learning. That is, the output of the local training algorithm should be integrated. However, known defenses, such as robust losses [13] and anomaly detection [43], assume control of the participants or explicit observation of the training data. There is a big challenge for ensuring the integrity is that neither of these assumptions apply to the federated learning since the server only observes local model parameters sent from each participant.

Contribution. To address this challenge, we propose a new privacy-preserving federated learning scheme which can guarantee the integrity of deep learning processes.

The contribution of our paper can be summarized as follows.

- **Privacy**. We design a training scheme for collaborative deep learning that offers data privacy. The scheme enables multiple participants to learn a neural network model on their own inputs, without revealing these inputs but benefiting from other participants.
- **integrity**. Based on a trusted execution environment, we give a training-integrity protocol for the privacy-preserving algorithm. Through running the protocol, the learning system can easily detect any dishonest action, such as tampering the model trained locally and delaying local training processes. Thus, the server and each participant is compelled to execute the local learning algorithm correctly.
- **Implementation**. We evaluate the performance of our scheme by prototype implementations. The experimental result shows that the scheme is training-integrity and practical.

Organization. The rest of this paper is organized as follows. We discuss existing works in more details and compare them with our work in Section 2. Some preliminaries and notations are described in Section 3. Then, we present the system architecture of the proposed scheme in Section 4. Section 5 describes the details of the proposed scheme. The implementation of a prototype and the experimental results are shown in Section 6. Finally, we draw a conclusion in Section 7.

2. Related works

2.1. Federated learning

Traditional secure machine learning techniques are usually based cryptographic tools such as garbled circuits, secret sharing, and encryption schemes with homomorphic properties. Lindell and Pinkas [27] proposed some privacy-preserving approaches with respect to data mining tasks. It is worth noting that beside mathematical assumptions, some of the above approaches also rely on computing parties being honest when running the protocol as well as non-colluding. Garbled circuits [54] provide a mechanism for multiple parties to compute any function on their joint inputs without having to reveal the inputs to each other. Secure machine leanring solutions based on garbled circuits have been tailored for several specific machine learning tasks including matrix factorization [39], SVMs[22], and decision trees [4,16,28]. GraphSC [36] and ObliVM [29] are two recent programming frameworks for secure computation using garbled circuits. The former framework offers a paradigm for parallel computations and the latter uses a combination of ORAMs and garbled circuits. Homomorphic encryption techniques allow multiple parties to encrypt their data and request the server to compute a joint function by performing computations directly on the ciphertexts. Bost et al. [7] proposed a secure classification scheme over encrypted data. Some privacy-preserving solutions based on homomorphic encryption techniques have been proposed for training machine learning models such as nerual networks [12,14,24–26,53] and decision trees [51].

In the federated learning scenario, Shokri and Shmatikov [45] proposed a method for multiple parties to train a deep neural network on joint inputs. Ohrimenko [40] proposed oblivious solutions for multi-party machine learning tasks based on trusted execution environments such as SGX. Privacy implications of revealing the output of a machine learning algorithm refers to Fredrikson et al. [9,10]. Differential privacy guarantees for the output of several machine learning algorithms have been proposed by Blum et al. [6]. Mohassel and Zhang [35] gave a new approximate fixed-point multiplication protocol for neural network models.

2.2. Causative attack

Different from adversarial example attacks [23,50], traditional causative attacks corrupt the training data to change the model's behaviour at inference time [5,17,31]. By poisoning the training data with backdoored examples, backdoor-based attacks affect the model's behavior only on specific attacker-chosen inputs [8,30], without impacting its performance on the main task. In [19], a backdoored component is inserted directly into the model. Some general poison attacks in machine learning were proposed by Alfeld et al. [2], Hayes and Ohrimenko [15], Jagielski et al. [18], Koh and Liang [21], Xiao et al. [52], where the aim of attacks is to degrade the performance of a model.

In practical applications, Newsome et al. [38] proposed red herring attacks that add spurious words to reduce the maliciousness score of an instance. These attacks work against conjunctive and Bayes learners for the worm signature generation. Perdisci et al. [41] practically demonstrate how an attacker can inject noise in the form of suspicious flows to mislead the worm signature classification. Nelson et al. [37] present both availability and targeted poisoning attacks against the public SpamBayes spam classifier. Defenses against poisoning focus on removing outliers from the training data [42,48] or the participants' models [11,44].

The state-of-the-art causative attack that works against the federated learning is to manipulate the attacker's model and directly affect the global model [3]. Our training-integrity scheme in this paper mainly defends against this attack.

3. Preliminaries

3.1. Neural network learning

A neural network learning task is to extract high-dimensional data's features and use them to build a neural network model that indicates the conditional distribution of inputs (e.g., an input vector $\mathbf{x} = (x_1, x_2, \dots, x_a)$) to outputs (e.g., a class y). The multi-layer perception is the most common form of neural network learning architectures. In this typical architecture, each hidden layer node takes the output of the previous layer' neurons as input, and then computes a weighted average of the inputs. It finally outputs a non-linear activation function value to the total input value.

Learning the weights of a neural network is a non-linear optimization problem. In supervised learning, the objective function is the output error of the neural network when forwarding all training data. The algorithms that are used to solve this problem are typically variants of gradient descent. The gradient descent usually starts at a random point. In following epochs, a trainer computes the gradient of the non-linear objective function being optimized using training data and updates the weights. This process will continue some epochs until the algorithm converges to a local optimum.

Stochastic Gradient Descent. The gradients of the model parameters can be averaged on all training data. Stochastic gradient descent (SGD) is a drastic simplification which computes the gradient on an appropriate small subset which is a mini-batch of the whole training set. In another word, corresponding to maximum stochasticity, one data sample is selected at random in each optimization step.

Let θ be the set (or flattened vector) of all weights in a multi-layer perception, composed of an output layer weight matrix and several hidden layer weight matrices. Set that E is the error function whose values indicate the difference between the true labels and the computed outputs of the network θ. The algorithm carries out partial derivatives of E with respect to current θ and updates θ so as to reduce its gradient. We denote the model weights at the beginning of the t-epoch as θ^{t-1}. The mean error function E_S is computed over a batch S and η is the learning rate. The update rule of SGD for a parameter θ^{t-1} in this epoch is shown as follows.

$$\theta^t \leftarrow \theta^{t-1} - \eta \frac{\partial E_S}{\partial \theta^{t-1}}. \tag{1}$$

3.2. Remote attestation and integrity

The Trusted Execution Environment (TEE), such as SGX [32], is a set of instructions by which applications can create protected memory regions for the execution of programs. These regions can be seen as enclaves isolated from any other code in the system, such as operating system and hypervisor. The processor only allows code running in an enclave to access data in the enclave. When loading the processor's caches, the memory of enclave is available in clear text. Alternatively, it is protected for confidentiality and integrity when interacting with system memory. The code not loaded inside the enclave can only invoke the internal code at statical entry points.

The TEE also provides the ability of attestation and sealing. Concretely, code inside an enclave can get messages signed with a per-processor private key along with a digest of the enclave. This ability enables other entities to ensure that the messages they received coming from a genuine enclave with the expected code and data configuration. Henceforth, applications can set up fine-grained TEEs even in hostile or compromised hosts with a security guarantee. However, the enclave program developers are still responsible for maintaining the confidentiality of secrets managed by the enclave.

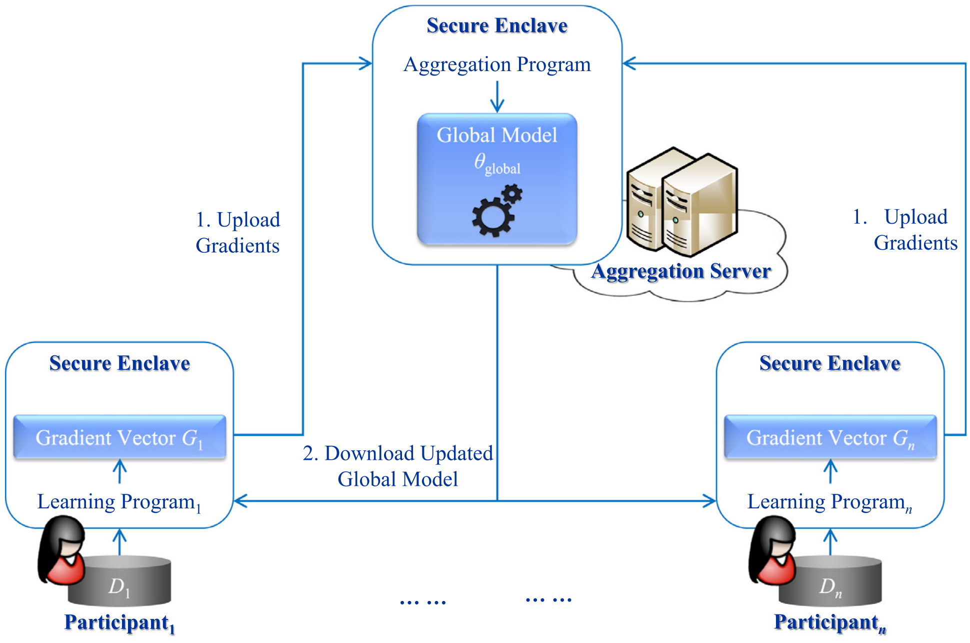


Fig. 1. System architecture.

In more details, when a code fragment prog (even it is probabilistic) is prepared to run in a TEE embedded with a permanent main key pair $(\mathsf{mpk}, \mathsf{msk})$, an enclave with the ID *eid* will be set up and execute the operations privately as follows.

- $Install(\mathsf{prog}) \rightarrow \mathsf{mem}$:
  The enclave allocates memory space mem for the program prog;
- $Execute(\mathsf{prog}, \mathsf{inp}, \mathsf{mem}) \rightarrow (\mathsf{outp}, \mathsf{mem})$:
  Taking inp as input, the enclave runs prog on the memory mem to obtain the result outp and update mem;
- $Sig_{\mathsf{msk}}(eid, \mathsf{prog}, \mathsf{outp}) \rightarrow \sigma$:
  The enclave runs the signature algorithm *Sig* of this TEE to sign the tuple of enclave ID *eid*, program prog, and result outp.

Finally, the enclave outputs outp with the corresponding signature σ. Thus, the remote attestation can be achieve by using the public main key msk to verify the signature σ.

Integrity. The mechanism above guarantees the integrity of programs in the TEE, which means outputs of the TEE are tamper-proof. Here, we give the notion of integrity. For any Probabilistic Polynomial Time (PPT) adversary $\mathcal{A}$, program prog, and input inp, the probability that $\mathcal{A}$ interacting with the TEE processor does not abort and outputs an incorrect value outp' with a signature $\sigma' = Sig_{\mathsf{msk}}(eid, \mathsf{prog}, \mathsf{outp}')$ is negligible.

4. System architecture

In this section, we show the system architecture of our proposed scheme, including the entities of federated learning, the threat model, and the guarantees of data privacy and training integrity.

4.1. Overview

Fig. 1 illustrates the main parties of our learning system, including some participants and an aggregations server.

- **Participant**. We set there are n participants. Each participant holds a local private dataset for local training. All participants reached a consensus on a global neural network architecture and a common learning goal in advance.
- **Aggregation Server**. We also envision the existence of a server which is responsible for aggregating the latest values of model parameters available to all parties. Normally, such an aggregation server is an abstraction that can be implemented by an actual cloud server.

Each participant shares global model parameters and then trains on its own training set. The system runs a model parameter aggregation protocol, which enables participants to contribute the gradient of the neural network model to the aggregation server and download the newest parameters of the model in each local SGD period. In this way, participants can either converge to the public model parameters or avoid over fitting the model to a single participant's local training set. Once the training is finished, each participant can predict new data privately without interaction with others.

The computations of gradients are run in the TEE of each participant, while the aggregation of gradients is run in the TEE of the server. Moreover, TEE set up a secure authenticated communication channel between each participant and the server, using the TEE's remote attestation property.

4.2. Threat model

Firstly, our privacy-preserving scheme aims at preserving privacy of the training data and maintain public protocol of the deep learning task. The external adversaries who want to extract sensitive information from training datasets can be honest-but-curious participants who are not active. In this way, the participants run the protocol as designed but may try to infer sensitive information from others' training data.

Secondly, similar to attackers in the causative attack, we consider an actively malicious adversary of which goal is to corrupt the learning model by submitting modified gradients, so that the prediction functionality of the trained neural network model is degraded. More specifically, this adversary's attack is considered an availability attack, if its goal is to degrade the prediction functionality of trained global model. Otherwise, it can be an integrity attack while the adversary's goal is to cause specific mis-predictions.

4.3. Privacy and integrity

Then, we give the intuition of ensuring data privacy and training integrity.

Privacy. Unlike traditional deep learning, participants do not disclose their training data set to anyone in our system. Therefore, the system ensures the privacy of training data. The size of the local dataset are confidential, and data are sampled differently in each round of SGD. All participants can obtain the global model, so they can use it to make predictions privately. Therefore, compared with the traditional deep learning, there is absolutely no leakage even when using this model.

Integrity. As shown above, we use TEEs to offer an efficient solution for the federated deep learning. That is, each party runs an enclave that initiates a secure communication. Enclaves of TEEs can produce remote attestations using digital signatures over an enclave's code, so that a remote party can verify using the manufacturer's public key. If there is any party violates operations of the training protocol, it will be identified immediately. Thus, the training Integrity can be achieved.

5. The training-integrity scheme

In this section, we describe the details of our proposed scheme.

5.1. Overview

Despite the integrity mechanism, we give a training protocol relies on the following situations.

- Each participant independently updates local parameters during gradient descent steps;
- Different participant's dataset contributes different update values to the global model;
- Different features do not contribute to the global loss function equally.

In each learning iteration, the learning algorithm based on SGD can achieve the same accuracy as the traditional SGD. After each iteration of training, the n participants asynchronously share with each other they computed gradients results for updating modle parameters. Each participant fully controls the shared gradients. The global descent is determined by the sum of all gradients contributed towards the local model's optima. Participants thus benefit from each other's training data and produce a more accurate model that is trained in isolation and not limited to a local training dataset.

As described in Section 4, we assume an abstraction of an aggregation server to which participants asynchronously upload the gradients. The main intuition behind this distributed protocol is that for each training dataset D, the participant compute the partial derivative $\frac{\partial E_D}{\partial \theta}$ for all model parameters θ as in SGD. Thus, the server can aggregate all partial derivatives whose weighted mean is the global gradients actually. Then, the server updates the flatten parameter vector θ in the same way as in Eq. (1). In the end of a learning iteration, each participant downloads the global model's parameters from the server and uses these values to update his/her local model.

5.2. Local training

We set that each participant i holds a dataset D_i. At the beginning of the training, the aggregation server and each participant will share a set of public hyperparameters, such as the architecture of the neural network model, the learning rate η, the loss function E, and so on. Moreover, each participant i claims his/her dataset's size $l_i = |D_i|$ along with the batch-chosen ratio α_i. These parameters are set as a part of public hyperparameters.

Algorithm 1 One-iteration local training on the ith participant.

```
Input:
  the local dataset $D_i$ and the current local model $\theta_i^{t-1}$
Output:
  the current iteration's gradients $G_i^t$
1:  if $|D_i| \neq l_i$ then
2:    Set $G_i^t \leftarrow \perp$;
3:  else
4:    Compute $s \leftarrow \alpha_i l_i$;
5:    Initialize batch $B_i \leftarrow \emptyset$;
6:    for $j = 1, 2, \ldots, s$ do
7:      Randomly sample $b_j \overset{R}{\leftarrow} D_i$;
8:      Insert $b_j$ into $B_i$;
9:    end for
10:   Compute $G_i^t \leftarrow \frac{\partial E_{B_i}}{\partial \theta_i^{t-1}}$;
11: end if
12: return $G_i^t$.
```

Each participant i maintains a local flatten vector of neural network model's parameters, while the aggregation server maintains a global model. Each participant i can initialize his/her local model's parameters θ_i^0 by downloading the randomly initialized model θ_{global}^0 from the aggregation server.

We envision that each participant starts to trainsthe model using the standard SGD algorithm and then iterates over local training data over several epochs. There is no need of any coordination between different participants during each local training procedure. The influence on each others' training is with the aggregation server.

We take the tth iteration as an example, and Algorithm 1 presents the pseudo-code of our learning algorithm run on the ith participant. The algorithm is run independently by each participant and consists of three steps in each learning epoch. Firstly, the ith participant downloads current global model θ_{global}^{t-1} from the server and overwrites the local model θ_i^{t-1} with the downloaded values. Then, this participant compute the gradients of loss function E on the local dataset D_i. This training is done on a series of mini-batches each of which is the set of randomly chosen training data samples of size $\alpha_i l_i$. This training process can avoid overfitting what other participants exactly need. In the third step, this participant only uploads the flatten gradient vector G_i^t which do not leak too much information about the training dataset D_i.

However, as described in Section 4, a participant may be an active attack who wants to contaminate the global model. To achieve this, he/she can bypass the local SGD algorithm and generate "incorrect" gradients (i.e., $G_i^{\prime t}$) to directly inject some malicious property, such as a backdoor, into the global model. Such an attack will certainly break the availability of the trained global model in the final.

Thanks to the our public hyperparameters, we can enforce each dishonest participant to run the standard SGD algorithm locally. Note that our training algorithm only takes a participant's dataset D_i and the current local model θ_i^{t-1} as input, which means all public hyperparameters, including the important parameters l_i and α_i, can be embedded in the program and be never tampered.

For the participant i, Algorithm 1 is implemented as a verified program prog_i and then set as public. In each learning iteration, prog_i is run in the ith participant's TEE which is embedded with a permanent main key pair $(\mathsf{mpk}_i, \mathsf{msk}_i)$. He/she sets up an enclave with the ID *eid* and execute the operations as follows.

- $Install(\mathsf{prog}_i) \rightarrow \mathsf{mem}$:
  The enclave allocates memory space mem for the learning program prog_i;
- $Execute(\mathsf{prog}_i, \theta_i^{t-1}, D_i, \mathsf{mem}) \rightarrow (G_i^t, \mathsf{mem})$:
  Taking the dataset D_i and the current local model θ_i^{t-1} as input, the enclave runs the learning program prog_i on the memory mem to obtain the gradients G_i^t and update mem;
- $Sig_{\mathsf{msk}_i}(eid, \mathsf{prog}_i, G_i^t) \rightarrow \sigma_i$:
  The enclave runs the signature algorithm *Sig* of this TEE to sign the tuple of enclave ID *eid*, program prog_i, and gradients G_i^t.

Finally, the enclave submits (G_i^t, σ_i) to the TEE on the server side via a secure channel (or a secure exchange protocol).

The operations above are private and the participant cannot violate these procedures. Therefore, if this participant uploads any faked flatten gradient vector $G_i^{\prime t}$ or tampers the program prog_i, the server will immediately capture this attack by using the public main key msk_i to verify the signature σ_i.

5.3. Global aggregation

Recall that the aggregation server initializes the flatten model vector θ^0_{global} and then handles the participants' upload and download requests in each iteration.

Algorithm 2 shows the pseudocode of server's global aggregation. When all n participants upload gradients, the server will make an attestation to exclude the illegal gradients. We assume that the number of legal gradients is m, and these values compose a set $\{G^t_{I(j)}\}^m_{j=1}$ where $I(j)$ indicates the origin index number of the jth legal gradient vector.

Algorithm 2 Global aggregation algorithm.

```
Input:
    the current global model $\theta^{t-1}_{global}$ and a set of legal gradient vectors $\{G^t_{I(j)}\}^m_{j=1}$
Output:
    the updated global model $\theta^t_{global}$
 1: Set $G^t_{global} \leftarrow \mathbf{0}$;
 2: Set $l \leftarrow 0$;
 3: for $j = 1, 2, \ldots, m$ do
 4:     if $G^t_{I(j)} \neq \perp$ then
 5:         Compute $G^t_{global} \leftarrow G^t_{global} + \alpha_{I(j)}\ l_{I(j)}\ G^t_{I(j)}$;
 6:         Compute $l \leftarrow l + \alpha_{I(j)} l_{I(j)}$;
 7:     end if
 8: end for
 9: Compute $G^t_{global} \leftarrow \frac{1}{l} G^t_{global}$;
10: Update $\theta^t_{global} \leftarrow \theta^{t-1}_{global} - \eta G^t_{global}$;
11: return $\theta^t_{global}$.
```

Then, it takes $\{G^t_{I(j)}\}^m_{j=1}$ as input and aggregates them to obtain the current global gradients G^t_{global}. Such gradients are used for updating the current global model θ^{t-1}_{global} with a pre-claimed learning rate η, i.e., $\theta^t_{global} \leftarrow \theta^{t-1}_{global}$ - ηG^t_{global}. Participants obtain from the server the latest values of the global model in the end of this iteration.

To enable the remote attestation, the program of global aggregation procedures is run in the server's TEE which is embedded with a permanent main key pair $(\mathsf{mpk}, \mathsf{msk})$. Similarly, Algorithm 2 is implemented as a verified program prog_{agg} and then set as public. In each learning iteration, prog_{agg} is run by setting up an enclave with the ID *eid*. The enclave's operations are executed as follows.

- $Install(\mathsf{prog}_{agg}) \rightarrow \mathsf{mem}$:
  The enclave allocates memory space mem for the aggregation program prog_{agg};
- $Execute(\mathsf{prog}_{agg}, \theta^{t-1}_{global}, \{G^t_{I(j)}\}^m_{j=1}, \mathsf{mem}) \rightarrow (\theta^t_{global}, \mathsf{mem})$:
  Taking the current global model θ^{t-1}_{global} and each legal gradient vector $G^t_{I(j)}$ as input, the enclave runs the aggregation program prog_{agg} on the memory mem to obtain the updated global model θ^t_{global} and update mem;
- $Sig_{\mathsf{msk}}(eid, \mathsf{prog}_{agg}, \theta^t_{global}) \rightarrow \sigma$:
  The enclave runs the signature algorithm *Sig* of this TEE to sign the tuple of enclave ID *eid*, program prog_{agg}, and updated global model θ^t_{global}.

Finally, the enclave returns $(\theta^t_{global}, \sigma)$ to the TEE on each participant via a secure channel (or a secure exchange protocol).

The aggregation operations are private and the server cannot violate these procedures. The learning will terminate when the model θ^t_{global} is convergent or the iteration t reaches the maximal number *fin*. Finally, each participant shares the trained model θ^{fin}_{global}.

6. Evaluation

We perform experiments with the goals: 1) to evaluate the success rate of the state-of-the-art federated-learning causative attack [3] on global model trained using our scheme; 2) to evaluate the accuracy drop of the global model as compared to the benign model.

6.1. Dataset and network architecture

We evaluate our scheme on two major datasets which are usually used as benchmarks in the previous deep learning tasks.

The first is a location dataset from the publicly available “check-in’ location information of mobile users in Foursquare social network, which is limited to Bangkok area and was collected from April 2012 to September 2013. We select 50,000 user profiles by excluding users with fewer than 25 check-ins and venues with fewer than 100 visits. Each record in the left dataset has 446 binary features that represent whether the user visited locations with a certain type or a certain region. There are 30 different geo-social location types. This dataset is composed of 40,000 training examples and 10,000 test examples.

The second is the MNIST dataset of handwritten digits which are all formatted as 32×32 images and normalized at the center of the image. This dataset is composed of 60,000 training examples and 10,000 test examples.

On the location dataset, we train a fully connected neural network model with one hidden layer with 128 nodes and a SoftMax layer. We use Tanh as the activation function. On the MNIST dataset, we train a fully connected neural network model with two hidden layers which contain 128 nodes and 64 nodes, respectively. The output layer is also a SoftMax layer. We use ReLU as the activation function.

6.2. Experimental setup

We implemented federated learning algorithms using the Tensorflow framework. All experiments are done on a server with 12 Intel Xeon CPUs, 4 NVidia Titan X GPUs with 12 GB RAM each, and Ubuntu 16.04LTS OS.

During the training, both the benign and malicious participants submit their gradient values of the training data to the server for each iteration. Each gradient value corresponds to a particular parameter of the neural network model. We set the learning rate to 0.001, the learning rate decay to $1e07$, and the maximum epochs of training to 200.

Benign Participant. Without loss of generality, the divergences on the distribution of each participant’s dataset should be reflected. We uses K-Means clustering algorithm to separate all the training examples into 100 parts each of which represents a participant’s local training set. Thus, it is difficult to detect a malicious participant from submitted gradients by using traditional defense approach described in Section 2.

Malicious Participant. There are a set of attackers that act as participants and involve in the training, such that they will initiate attacks by submitting purposefully manipulated gradients in each learning iteration. The goal of these malicious participants is to corrupt the global model generated in the training phase, so that predictions of the trained model on new data will be modified. We consider two types of attacks. An attack is considered a general attack, if its goal is to affect prediction results indiscriminately. It is instead referred to as a targeted attack, if the goal is to cause specific mis-predictions. In the experiments of location dataset, we set the specific mis-prediction is to mislabel 21 to 6. In the experiments of MNIST, the specific mis-prediction is set to mislabel 4 to 9.

6.3. Experimental results

In each experiment, we set compromised participants in different proportions by varying the malicious rate between 0% and 20% at intervals of 4% with the goal of inferring the trend in attack success. The ratio 0% means the model is trained in a totally benign environment. We measure the attack success rate by accuracy drop of two final global models. One is trained using our defense, and the other is trained using the traditional federated learning scheme [45].

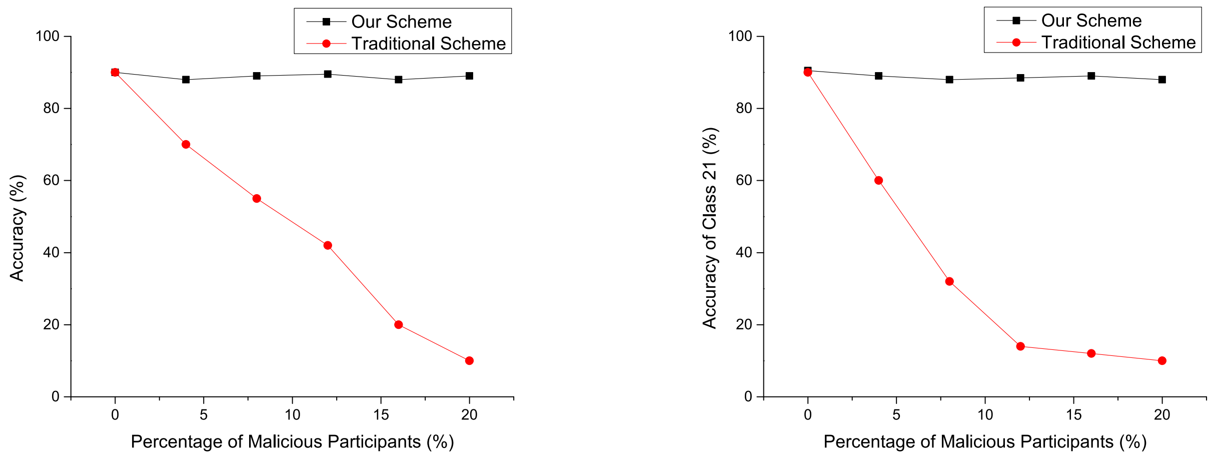
In each iteration of training, participants’ gradients are trained separately and sequentially before they are aggregated into a new global model. Training for one iteration epoch of a single participant on its local data takes 0.01 and 0.1 seconds for location and MNIST, respectively.

Location Dataset. Firstly, we train and test models on the location dataset. Fig. 2a shows the comparison on the accuracy drop caused by general attacks. Different from the traditional scheme, our scheme removes the manipulated gradients before the aggregation step of each learning iteration. We observe that the accuracy drop (attack success rate) reduces largely after training the model without the malicious values, which are all below when malicious ratio varies from to 0% to 20%. Fig. 2b shows the accuracy drop as compared to the benign setting when mislabelling a type-21 location as a type-6 location. For the model trained using our scheme, the accuracy drop is very few for the malicious ratios from 0% to 20%, indicating that the overall accuracy of the model is not affected drastically by removing the dataset contributed by malicious participants.

MNIST. Secondly, we train and test models on the MNIST dataset. Fig. 3a shows the comparison on the accuracy drop caused by general attacks. Different from the traditional scheme, our scheme removes the manipulated gradients before the aggregation step of each learning iteration. We observe that the accuracy drop (attack success rate) reduces largely after training the model without the malicious values, which are all below when malicious ratio varies from to 0% to 20%. Fig. 3b shows the accuracy drop as compared to the benign setting when mislabelling an image with digit 4 as an image with digit 9. For the model trained using our scheme, the accuracy drop is very few for the malicious ratios from 0% to 20%, indicating that the overall accuracy of the model is not affected drastically by removing the dataset contributed by malicious participants.

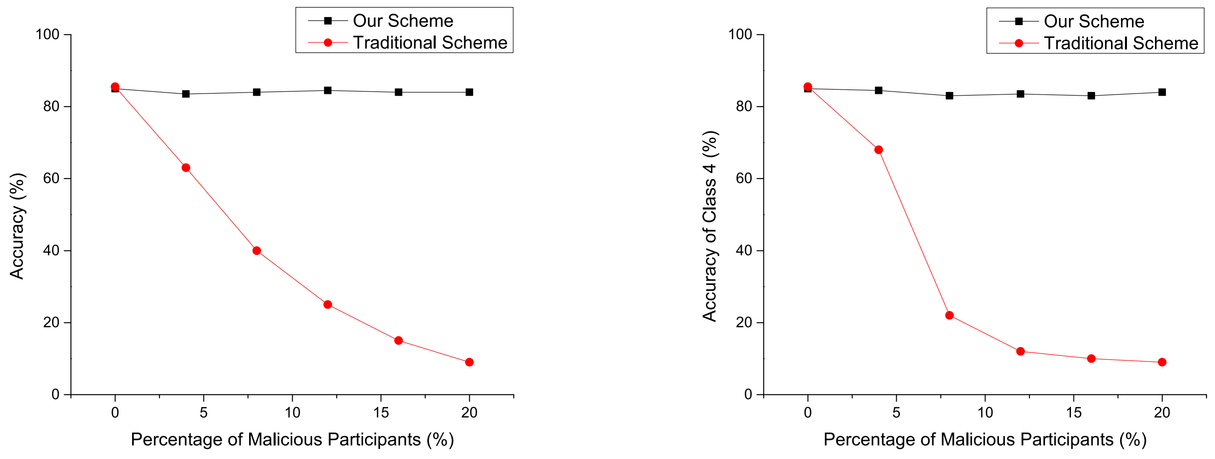
6.4. Discussion

The experimental results demonstrate our scheme can exclude the effect of malicious participants and thus achieves training integrity.



(a) Against General Attacks (b) Against Targeted Attacks

Fig. 2. Accuracy of Trained model on location dataset.



(a) Against General Attacks (b) Against Targeted Attacks

Fig. 3. Accuracy of trained model on MNIST dataset.

We must point out that our scheme is not equipped with the verification mechanism of training data. Thus, even we have removed the threats of state-of-the-art causative attacks, it seems that malicious participants can still perform traditional causative attacks in which “poisons” are injected into training data.

In traditional causative attacks, the attacker is assumed to control a significant fraction of the training data. However, there are usually thousands of participants in a federated learning system, which means the attacker has to corrupt the majority of participants to effectively implement a traditional attack. Obviously, this attack is impractical in the real-world. Some previous works [3] have demonstrated that the traditional attack method cannot work if only a few participants are compromised.

Therefore, we can say that our scheme can prevent causative attacks in federated learning such that the integrity of the well-trained global model is protected. Alternatively, the data verification problem is still an interesting and important topic, and we can leave this problem for future researches in the federated learning.

7. Conclusion

This work is the first step in bringing both privacy and integrity to a federated learning system that is revolutionizing the multi-party machine learning. We propose a new privacy-preserving federated learning scheme that guarantees the integrity of deep learning processes. Based on the trusted execution environment, our methodology ensures that each learning participant executes the privacy-preserving learning algorithm in a correct way, in which attacks that broke the availability of trained models can be detected. Therefore, the proposed scheme can help bring the benefits of deep learning techniques in adversarial domains where participants are precluded from training collaboratively by confidentiality and availability concerns.

Declaration of Competing Interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Yu Chen: Writing - original draft, Software, Validation. **Fang Luo:** Data curation, Investigation. **Tong Li:** Methodology, Writing - original draft. **Tao Xiang:** Writing - review & editing. **Zheli Liu:** Writing - review & editing. **Jin Li:** Conceptualization, Supervision.

Acknowledgment

This work was supported by National Natural Science Foundation of China (No. 61802078, No. 61702125 and No. 61960206014), National Natural Science Foundation for Outstanding Youth Foundation (No. 61722203), National Natural Science Foundation for Joint Fund Project (No. U1936218), and China Postdoctoral Science Foundation (No. 204728).

References

[1] M. Abadi, A. Chu, I. Goodfellow, H.B. McMahan, I. Mironov, K. Talwar, L. Zhang, Deep learning with differential privacy, in: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, ACM, 2016, pp. 308–318.
[2] S. Alfeld, X. Zhu, P. Barford, Data poisoning attacks against autoregressive models, in: Thirtieth AAAI Conference on Artificial Intelligence, 2016.
[3] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, V. Shmatikov, How to backdoor federated learning, (2018), http://arxiv.org/abs/1807.00459.
[4] M. Barni, P. Failla, V. Kolesnikov, R. Lazzeretti, A.-R. Sadeghi, T. Schneider, Secure evaluation of private linear branching programs with medical applications, in: European Symposium on Research in Computer Security, Springer, 2009, pp. 424–439.
[5] B. Biggio, B. Nelson, P. Laskov, Poisoning attacks against support vector machines, (2012), http://arxiv.org/abs/1206.6389.
[6] A. Blum, C. Dwork, F. McSherry, K. Nissim, Practical privacy: the SuLQ framework, in: Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, ACM, 2005, pp. 128–138.
[7] R. Bost, R.A. Popa, S. Tu, S. Goldwasser, Machine learning classification over encrypted data., in: NDSS, vol. 4324, 2015, p. 4325.
[8] X. Chen, C. Liu, B. Li, K. Lu, D. Song, Targeted backdoor attacks on deep learning systems using data poisoning, (2017), https://arxiv.org/abs/1712.05526.
[9] M. Fredrikson, S. Jha, T. Ristenpart, Model inversion attacks that exploit confidence information and basic countermeasures, in: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, ACM, 2015, pp. 1322–1333.
[10] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, T. Ristenpart, Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing, in: 23rd {USENIX} Security Symposium ({USENIX} Security 14), 2014, pp. 17–32.
[11] C. Fung, C.J. Yoon, I. Beschastnikh, Mitigating sybils in federated learning poisoning, (2018), http://arxiv.org/abs/1808.04866.
[12] T. Graepel, K. Lauter, M. Naehrig, Ml confidential: machine learning on encrypted data, in: International Conference on Information Security and Cryptology, Springer, 2012, pp. 1–21.
[13] B. Han, I.W. Tsang, L. Chen, On the convergence of a family of robust losses for stochastic gradient descent, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2016, pp. 665–680.
[14] A. Hassan, R. Hamza, H. Yan, P. Li, An efficient outsourced privacy preserving machine learning scheme with public verifiability, IEEE Access 7 (2019) 146322–146330.
[15] J. Hayes, O. Ohrimenko, Contamination attacks and mitigation in multi-party machine learning, in: Advances in Neural Information Processing Systems, 2018, pp. 6604–6615.
[16] S. de Hoogh, B. Schoenmakers, P. Chen, H. op den Akker, Practical secure decision tree learning in a teletreatment application, in: International Conference on Financial Cryptography and Data Security, Springer, 2014, pp. 179–194.
[17] L. Huang, A.D. Joseph, B. Nelson, B.I. Rubinstein, J.D. Tygar, Adversarial machine learning, in: Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence, ACM, 2011, pp. 43–58.
[18] M. Jagielski, A. Oprea, B. Biggio, C. Liu, C. Nita-Rotaru, B. Li, Manipulating machine learning: poisoning attacks and countermeasures for regression learning, in: 2018 IEEE Symposium on Security and Privacy (SP), IEEE, 2018, pp. 19–35.
[19] Y. Ji, X. Zhang, S. Ji, X. Luo, T. Wang, Model-reuse attacks on deep learning systems, in: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, ACM, 2018, pp. 349–363.
[20] X. Jiang, M. Kim, K. Lauter, Y. Song, Secure outsourced matrix computation and application to neural networks, in: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, ACM, 2018, pp. 1209–1222.
[21] P.W. Koh, P. Liang, Understanding black-box predictions via influence functions, in: Proceedings of the 34th International Conference on Machine Learning-Volume 70, JMLR. org, 2017, pp. 1885–1894.
[22] S. Laur, H. Lipmaa, T. Mielikäinen, Cryptographically private support vector machines, in: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2006, pp. 618–624.
[23] J. Li, X. Tang, Z. Wei, Y. Wang, W. Chen, Y. Tan, Identity-based multi-recipient public key encryption scheme and its application in IoT, Mob. Netw. Appl. (2019) 1–8.
[24] P. Li, T. Li, H. Ye, J. Li, X. Chen, Y. Xiang, Privacy-preserving machine learning with multiple data providers, Future Gener. Comput. Syst. 87 (2018) 341–350.
[25] T. Li, C. Gao, L. Jiang, W. Pedrycz, J. Shen, Publicly verifiable privacy-preserving aggregation and its application in IoT, J. Netw. Comput. Appl. 126 (2019) 39–44.
[26] T. Li, X. Li, X. Zhong, N. Jiang, C. Gao, Communication-efficient outsourced privacy-preserving classification service using trusted processor, Inf. Sci. 505 (2019) 473–486.
[27] Y. Lindell, Secure multiparty computation for privacy preserving data mining, in: Encyclopedia of Data Warehousing and Mining, IGI Global, 2005, pp. 1005–1009.

[28] Y. Lindell, B. Pinkas, Privacy preserving data mining, in: Annual International Cryptology Conference, Springer, 2000, pp. 36–54.
[29] C. Liu, A. Harris, M. Maas, M. Hicks, M. Tiwari, E. Shi, Ghostrider: a hardware-software system for memory trace oblivious computation, in: ACM SIGPLAN Notices, vol. 50, ACM, 2015, pp. 87–101.
[30] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang, X. Zhang, Trojaning attack on neural networks (2017).
[31] S. Mahloujifar, M. Mahmoody, A. Mohammed, Multi-party poisoning through generalized *p*-tampering, (2018), http://arxiv.org/abs/1809.03474.
[32] F. McKeen, I. Alexandrovich, A. Berenzon, C.V. Rozas, H. Shafi, V. Shanbhogue, U.R. Savagaonkar, Innovative Instructions and Software Model for Isolated Execution, 10, 2013. Hasp@ isca
[33] H.B. McMahan, D. Ramage, K. Talwar, L. Zhang, Learning differentially private language models without losing accuracy, (2017), http://arxiv.org/abs/1710.06963.
[34] P. Mohassel, P. Rindal, Aby 3: a mixed protocol framework for machine learning, in: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, ACM, 2018, pp. 35–52.
[35] P. Mohassel, Y. Zhang, SecureML: a system for scalable privacy-preserving machine learning, in: 2017 IEEE Symposium on Security and Privacy (SP), IEEE, 2017, pp. 19–38.
[36] K. Nayak, X.S. Wang, S. Ioannidis, U. Weinsberg, N. Taft, E. Shi, GraphSC: parallel secure computation made easy, in: 2015 IEEE Symposium on Security and Privacy, IEEE, 2015, pp. 377–394.
[37] B. Nelson, M. Barreno, F.J. Chi, A.D. Joseph, B.I. Rubinstein, U. Saini, C.A. Sutton, J.D. Tygar, K. Xia, Exploiting machine learning to subvert your spam filter., LEET 8 (2008) 1–9.
[38] J. Newsome, B. Karp, D. Song, Paragraph: thwarting signature learning by training maliciously, in: International Workshop on Recent Advances in Intrusion Detection, Springer, 2006, pp. 81–105.
[39] V. Nikolaenko, S. Ioannidis, U. Weinsberg, M. Joye, N. Taft, D. Boneh, Privacy-preserving matrix factorization, in: Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, ACM, 2013, pp. 801–812.
[40] O. Ohrimenko, F. Schuster, C. Fournet, A. Mehta, S. Nowozin, K. Vaswani, M. Costa, Oblivious multi-party machine learning on trusted processors, in: 25th {USENIX} Security Symposium ({USENIX} Security 16), 2016, pp. 619–636.
[41] R. Perdisci, D. Dagon, W. Lee, P. Fogla, M. Sharif, Misleading worm signature generators using deliberate noise injection, in: 2006 IEEE Symposium on Security and Privacy (S&P'06), IEEE, 2006, pp. 15–pp.
[42] M. Qiao, G. Valiant, Learning discrete distributions from untrusted batches, (2017), http://arxiv.org/abs/1711.08113.
[43] B.I. Rubinstein, B. Nelson, L. Huang, A.D. Joseph, S.-h. Lau, S. Rao, N. Taft, J.D. Tygar, Antidote: understanding and defending against poisoning of anomaly detectors, in: Proceedings of the 9th ACM SIGCOMM conference on Internet measurement, ACM, 2009, pp. 1–14.
[44] S. Shen, S. Tople, P. Saxena, Auror: defending against poisoning attacks in collaborative deep learning systems, in: Proceedings of the 32nd Annual Conference on Computer Security Applications, ACM, 2016, pp. 508–519.
[45] R. Shokri, V. Shmatikov, Privacy-preserving deep learning, in: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, ACM, 2015, pp. 1310–1321.
[46] R. Shokri, M. Stronati, C. Song, V. Shmatikov, Membership inference attacks against machine learning models, in: 2017 IEEE Symposium on Security and Privacy (SP), IEEE, 2017, pp. 3–18.
[47] C. Song, T. Ristenpart, V. Shmatikov, Machine learning models that remember too much, in: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, ACM, 2017, pp. 587–601.
[48] J. Steinhardt, P.W.W. Koh, P.S. Liang, Certified defenses for data poisoning attacks, in: Advances in Neural Information Processing Systems, 2017, pp. 3517–3529.
[49] F. Tramèr, F. Zhang, A. Juels, M.K. Reiter, T. Ristenpart, Stealing machine learning models via prediction APIs, in: 25th {USENIX} Security Symposium ({USENIX} Security 16), 2016, pp. 601–618.
[50] X. Wang, J. Li, X. Kuang, Y. Tan, J. Li, The security of machine learning in an adversarial setting: a survey, J. Parallel Distrib. Comput. 130 (2019) 12–23.
[51] D.J. Wu, T. Feng, M. Naehrig, K. Lauter, Privately evaluating decision trees and random forests, Proc. Privacy Enhancing Technol. 2016 (4) (2016) 335–355.
[52] H. Xiao, B. Biggio, G. Brown, G. Fumera, C. Eckert, F. Roli, Is feature selection secure against training data poisoning? in: International Conference on Machine Learning, 2015, pp. 1689–1698.
[53] P. Xie, M. Bilenko, T. Finley, R. Gilad-Bachrach, K. Lauter, M. Naehrig, Crypto-nets: neural networks over encrypted data, (2014), http://arxiv.org/abs/1412.6181.
[54] A.C.-C. Yao, How to generate and exchange secrets, in: 27th Annual Symposium on Foundations of Computer Science (sfcs 1986), IEEE, 1986, pp. 162–167.