MP-BADNet: A Backdoor-Attack Detection and Identification Protocol among Multi-Participants in Private Deep Neural Networks

Congcong Chen
College of Information Technology
Shanghai Ocean University
Shanghai, China

Lifei Wei*
College of Information Technology
Shanghai Ocean University
Shanghai, China

Lei Zhang*
College of Information Technology
Shanghai Ocean University
Shanghai, China

Jianting Ning†
College of Mathematics and Informatics
Fujian Normal University
Fuzhou, China

ABSTRACT

Deep Neural Networks (DNNs) are vulnerable to backdoor attacks where the adversary can inject malicious data during the DNN training. Such kind of attacks is always activated when the input is stamped with a pre-specified trigger which results in a pre-setting prediction of the DNN model. Due to increasing applications of DNNs, it is necessary to detect the backdoors whether the DNN model has been trojaned before implementation. Since the data come from the various data holders during the model training, it is also important to protect the privacy both of input data and models. In this paper, we propose a framework MP-BADNet, the first work on the backdoor attack detection and identification protocol among multi-participants in private deep neural networks. MP-BADNet can not only detect and identify backdoors in the privacy-preserving DNN model, but also achieve privacy preserving of input data and the model in secure multi-party computation (MPC) ways. The implemental results show that the scheme can effectively detect and identify backdoor attacks in the privacy-preserving DNN model.

CCS CONCEPTS

• **Security and privacy → Privacy-preserving protocols**; **Security protocols**.

*Lifei Wei and Lei Zhang are the corresponding author. Their emails are Lfwei@shou.edu.cn and Lzhang@shou.edu.cn, respectively.
†Prof. Jianting Ning is also with State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093.

ACM TURC, July 30-August 1, 2021, Hefei, China

ACM ISBN 978-1-4503-8567-1/21/07…$15.00
https://doi.org/10.1145/3472634.3472660

KEYWORDS

Backdoor Attacks, Detection and Identification Protocol, Privacy-Preserving, Secure Multi-Party Computation, Deep Neural Networks.

ACM Reference Format:
Congcong Chen, Lifei Wei, Lei Zhang, and Jianting Ning. 2021. MP-BADNet: A Backdoor-Attack Detection and Identification Protocol among Multi-Participants in Private Deep Neural Networks . In *ACM Turing Award Celebration Conference - China (ACM TURC 2021) (ACM TURC), July 30-August 1, 2021, Hefei, China.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3472634.3472660

1 INTRODUCTION

Deep Neural Networks (DNNs) are widely used in fields such as data mining, computer vision , medical diagnosis [1]. The booming deep learning makes data security and privacy face severe threats. An accurate model of deep learning requires a large amount of training data [17]. Various training data comes from various data holders, but due to the laws and regulations like European Union General Data Protection Regulation (GDPR) or containing sensitive data, it can't be directly shared in plaintext for DNN model training.

The security of DNN model has been recognized as a real security issue [12], it is difficult to detect the security problem in the model especially in the case of ciphertext. Malicious attackers may embed backdoors or Trojans in the DNN models [10][4]. A backdoor attack is always using the images with triggers as input (also called adversarial input, trojaned data) to train DNN model and produce unexpected results. Until it is activated by the trojaned data, the backdoor in the model is difficult to detect.

Ideally, people hope to obtain an accurate model while hiding the plaintext data of each data holders. Secure multi-party computation (MPC) techniques provide a reliable solution for data analysis in the above scenarios [11, 16]. MPC allows parties that don't trust each other to jointly calculate a function on their own private inputs without revealing any information other than the output of the function. In the framework of MPC, the honest servers only hold fragments of the input data for privacy preserving when the malicious participants take the trojaned data for model training. It is difficult for the honest servers to find out whether the trained model contains a backdoor.

To our best of the knowledge, most prior works have focused on either privacy-preserving machine learning or poisoning attacks. The former focuses on the privacy issues in the machine learning such as linear regression, logistic regression, and neural networks. The latter is always to reduce the classification accuracy on clean input data [6]. However, backdoor attacks succeed to classify as a specific target for any trojaned data. The technique contributions of this paper are summarized as follows:

- **Achieving data and model privacy trained via MPC infrastructure.** We assume a semi-honest adversary environment in which adversaries are curious to the privacy but honest to carry out the instructions. We use MPC to train a DNN model among three non-collusion servers, which does not leak the privacy of input data and model parameters.
- **Enabling backdoor attacks detection and identification of DNNs.** We propose a generalized method to detect and identify backdoor attacks in a pre-trained privacy-preserving DNN model which hides the triggers of backdoor attacks. Computing servers and secondary server in our system jointly performs the calculations to detect whether the model has suffered backdoor attacks and identify the specific triggers embedded inside model through reverse engineering algorithm and outlier detection algorithm. The reverse engineering algorithm runs under ciphertext, which protects the privacy of data and the privacy of models.
- **Performing the evaluation of MP-BADNet.** We implement MP-BADNet in our end-to-end framework. Specifically, we evaluate MP-BADNet on a deep neural network and train it on the MNIST dataset. We reproduce BadNets [4] to add a trigger to about 10% of the MNIST dataset and use this dataset for model training to obtain a model with a backdoor. Our results demonstrate that our scheme can effectively detect and identify backdoor in the privacy preserving DNN model.

The rest paper is organized as follows. Section 2 introduces the related work. Section 3 describes our system architecture, security models, and designed goals. Section 4 gives the notation. In Section 5, we propose a detail construction and give the performance comparison in Section 6. Finally, Section 7 makes a conclusion.

2 RELATED WORK

2.1 Backdoor attacks

The experiments of Gu et al. [4] show that there is a 95% probability of attaching a trigger to the traffic sign to recognize the parking sign as a speed limit sign, which poses a huge threat to the autonomous driving scene. Liu et al. [10] proposed a backdoor attack scheme called Trojan Attack, which also shows that with a 99% probability a backdoor attacker can identify any person as a designated target. It has better concealment since the DNN model is not intuitive and difficult to understand as a black box. Bagdasaryan et al. [2] show that the federated learning is vulnerable to model poisoning attacks. Federated learning uses some secure aggregation methods to protect the local model and data of each participant. This also brings challenges to detect the abnormal model parameters submitted by the local participants as well as the aggregated model.

2.2 Backdoor defenses

In recent years, some solutions have been proposed to detect and mitigate backdoor attacks. Liu et al. [8] proposed a fine-pruning method that removes the backdoor by pruning neurons. This method of pruning neurons has little effect on clean input. However, the experiment of Wang et al. [15]proved that this method may greatly reduce the classification accuracy of the GTSRB dataset. Wang et al. [15] proposed neuron pruning and unlearning methods to mitigate backdoor attacks. In the same year, Liu et al. [9] proposed a technique to analyze the behavior of internal neurons, which determines whether the DNN model is attacked by a backdoor by introducing different levels of stimulation to a neuron and observing how the output activation changes.

2.3 Privacy-preserving machine learning

SecureML [11] is a privacy-preserving method for training neural networks in a two-party computation environment, which proposes an efficient truncation protocol based on the linear regression, logistic regression and neural network models and is much more efficient than the algorithm in ABY [3]. ABY3 proposes a novel scheme in a semi-honest environment and a malicious environment, which completes arithmetic sharing, boolean sharing, and Yao sharing [16] among three parties. SecureNN [13] makes improvements based on SecureML, and finally increased the efficiency. Recently, the researchers from Microsoft and Princeton University launch a framework called FALCON [14] based on SecureNN and ABY3 supporting batch normalization and training large-capacity networks.

3 OVERVIEW

3.1 A 4-party Backdoor Detection System

As shown in Figure 1, the system is composed of three entities: data holders, computing servers, and secondary server.

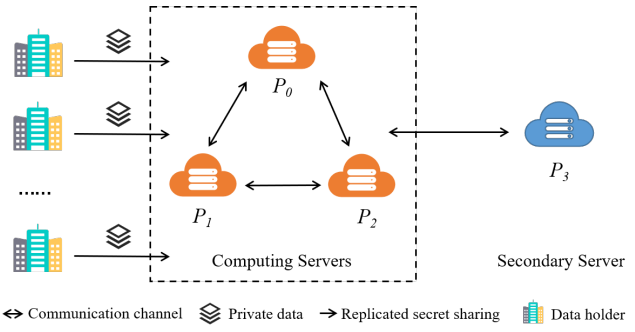


Figure 1: System architecture of MP-BADNet

Data holders. The dataset used to train the model comes from different data holders in companies, institutions, and organizations who are unwilling to share with each other. Data holders use replicated secret sharing (RSS) technique to share the fragment of the sensitive data to three computing servers for model training.

Computing servers. The computing servers, named as P_0, P_1, P_2, possess plenty of computation resources and storage resources

which are controlled by different owners. We use the MPC technique to train the DNN model among three non-collusion computing servers.

Secondary server. We set another server to detect and identify backdoor attacks on the trained model, named as P_3. The secondary server as well as the computing servers jointly detect backdoor whether the trained model has been backdoor attacked.

The datasets used to train the model and reverse engineer the trigger are held by the computing servers, and these datasets are provided by data holders and shared their data uses the RSS technique among the three computing servers. The computing servers and the secondary server communicate with each other to reverse engineer the trigger used by the backdoor attack. When steps that need to ensure the privacy are involved in the process of detecting and identifying backdoor attacks, they are all calculated by the computing servers. Finally, through the interaction among the 4-party servers, the privacy of the data and the trained model parameters are completely hidden to these non-collision servers.

3.2 Backdoor Attack Model

Suppose the model is outsourced to a third-party server that uses MPC techniques to secretly train the model. The datasets they use for training are provided by different data holders and shared by RSS technique. A DNN model is vulnerable to backdoor attacks since (1) the DNN model does not reduce the accuracy of the prediction of the clean datasets and (2) for any input with triggers, the target label with a high probability can be misclassified regardless of its original label.

We assume that each target label corresponds to only one type of trigger, and the DNN model has only one target label. In other words, the backdoor attacks with multiple target labels and multiple triggers are beyond the scope of MP-BADNet.

3.3 Threat Model and Design Goals

In our system, we assume an honest majority model among three computing servers. The fundamental setting of MP-BADNet is the same to the previous scheme [13, 14]. A semi-honest adversary tries to passively obtain the sensitive data of other participants but still executes the instructions according to the protocol. Besides, we assume that the private keys for communication among the four servers have been stored securely and are not easy to leak. Thus, the communication channels among the servers can be guaranteed to be secure. Furthermore, our protocol can achieve malicious security by adding a detection mechanism between the computing servers and secondary server during the data train.

The goals of MP-BADNet are as follows: (1) Detecting backdoor. We want to detect whether the privacy-preserving DNN model is attacked by a backdoor, and we want to know which label is the target of the attack. (2) Identifying backdoor. We want to use reverse engineering algorithms to get the trigger for backdoor attacks while protecting the privacy of input data and model parameters.

4 NOTATIONS

Let P_0, P_1, P_2, P_3 be the parties, where P_0, P_1, P_2 are computing servers, and P_3 is a secondary server. For computing servers, we use P_{i+1} and P_{i-1} to denote P_i's next party and previous party, respectively. For example, the next party of P_0 is P_1, and the previous party of P_0 is P_2. Let $\|x\|^m = (x_1, x_2, x_3)$ to denote RSS of a secret x modulo m. In other words, $x \equiv x_1 + x_2 + x_3 \pmod m$, where x_1, x_2, x_3 are random. We set $\|x\|^m$ to mean (x_1, x_2) is held by P_0, (x_2, x_3) by P_2, and (x_3, x_1) by P_3.

In the reverse engineering algorithm of backdoor attack detection, we use *mask* denote a 2D matrix, which determines how much trigger can overwrite the original image. The range of *mask* is (0, 1). We use *pattern* to denote the trigger pattern, which is a matrix with the same size as the input image. When $mask_{i,j}$ is 1 (where i, j represent the pixel in the i-th row and j-th column), it means that the pixel value of the i-th row and j-th column of the original image is completely replaced by the pixel value of $pattern_{i,j}$. When $mask_{i,j}$ is 0, it means that the constructed adversarial input pixel is completely the original image and does not cover any pixel of $pattern_{i,j}$.

5 CONCRETING PROTOCOL

5.1 Overarching Idea

Based on Falcon [14] and Neural Cleanse [15], we propose a method for backdoor attack detection and identification for a privacy-preserving DNN model that hides the backdoor and is trained using MPC technique. The advantages of MP-BADNet are as follows: (1) Multiple data holders can use their data to jointly train a model; (2) It can ensure the privacy of input data and model parameters; (3) It can detect whether the model trained by MPC has a backdoor, and what is the target label of the backdoor attack.

5.2 Detection System Design

We describe in detail the technique of four servers to jointly detecting and identifying backdoor attacks in the privacy-preserving DNN model. Suppose we have a DNN model with L labels. Consider a some label L_i and a target label L_t, where $i \neq t$. If there is a backdoor attack trigger T_t that misclassifies the label L_i as the target label L_t, the minimum perturbation required for all inputs: $\delta_{i \to t} \leq |T_t|$ [15].

Since all labels are to be misclassified as target labels and the trigger is a small stamp, thus we have, $\delta_{\forall \to t} \ll \min_{i, i \neq t} \delta_{\forall \to t}$, where $\delta_{\forall \to i}$ represents the minimum perturbation required from any label to target label L_t.

As described above, our detection system mainly includes the following steps:

- Step 1: For a given label, we regard it as a potential target label in a backdoor attack. We designed an optimization method that runs on four servers to find the trigger that misclassifies all other labels as the minimum modification value of the target label.
- Step 2: For all labels in the DNN model, perform step 1. If the DNN model contains N labels, N potential triggers will eventually be obtained.
- Step 3: For N potential triggers, we calculate the size of the trigger. Any trigger that is significantly smaller than the other candidates is a real trigger, and its corresponding label is the target label.

The combination of step 1 and step 2 is the trigger reverse engineering algorithm, and step 3 is the outlier detection algorithm. The secondary server and computing servers jointly complete the trigger reverse engineering algorithm to generate N potential triggers, and finally the secondary server runs the outlier detection algorithm to obtain the target trigger and target label.

5.2.1 Reverse Engineering Algorithm. We designed a trigger reverse engineering algorithm for the joint calculation of four servers. When the algorithm step involves input data and model parameters, it runs on computing servers with the MPC technique. When it does not involve data or model parameters, it runs in plaintext on the secondary server which mainly processes auxiliary calculations in plaintext, and does not involve any input data and model parameters. In this work, we assume that the secondary server and computing servers are non-colluded. In order to make the backdoor detection converge quickly, we use the Adam optimizer [7] to optimize *mask* and *pattern*.

The trigger reverse engineering algorithm is described in Algorithm 1. The step marked with 'in Ciphertext' means that it requires four server to cooperate to finish it. The other steps of the algorithm are all done in plaintext on the secondary server.

For example, to optimize the calculation of *pattern*, the secondary server first calculates *patternTanh*, and then sends it to the computing servers using RSS technique. The computing servers constructs adversarial input for prediction in the case of ciphertext, and calculates the gradient of *pattern*. Finally, computing servers sends the calculated gradient to the secondary server for *pattern* optimization.

The role of Reverse Engineering Algorithm is to reverse engineer potential target label triggers, and in steps 5-24 is the main procedure. In the Reverse Engineering algorithm, steps 1-3 are the preprocessing operations, step 6-12 is to use the input data to optimize *mask* and *pattern*, and steps 14-22 are to check whether *mask* and *pattern* are optimal. The step 23 of the Reverse Engineering function is to modify the weights to optimize *mask* and *pattern*.

5.2.2 Outlier Detection Algorithm. The outlier detection algorithm is described in Algorithm 2.We use the $L_1 - norm$ to calculate the size of potential triggers and *Median Absolute Deviation* (mad) technique to detect outliers, which is flexible in the presence of multiple outliers [5]. Outlier detection algorithm first calculates *mad* between all data points and the median. When assuming that the underlying distribution is a normal distribution, a constant 1.4826 will be used to normalize the outlier value. In this work, any label with an outlier value greater than 2 is marked as an outlier, that is, the label is the target label of a backdoor attack. $List_{mask}$ is the optimal *mask* calculated by the reverse engineering algorithm. As mentioned above, the step 6 is to normalize *mad* under the normal distribution. The steps 7-15 are to calculate all indexes with an outlier value greater than 2.

Algorithm 2 can be calculated locally based on the secondary server P_3. Finally, the computing servers P_0, P_1 and P_2, and the secondary server P_3 don't know the input data and model parameters. However, P_3 knows the final trigger as a detection server.

Algorithm 1: Trigger Reverse Engineering Algorithm

Input: $\|x\|^m$,$\|yTarget\|^m$, *pattern*,*mask*
Output: *maskBest*, *patternBest*, *maskUpsample*

1.1 Calculate *maskTanh*, *patternTanh*, *maskUpsample* and *patternRaw* in Ciphertext;
1.2 Initialize *maskBest*, *maskUnsampleBest*, *patternBest*;
1.3 Let $attackThreshold = 0.99$, $regBest = infinity$;
1.4 Perform one-hot encoding on $\|yTarget\|^m$ to generate $\|yTargetList\|^m$;
1.5 **while** *1* **do**
1.6 **for** *i in miniBatch* **do**
1.7 Obtain $\|xBatch\|^m$ in $\|x\|^m$;
1.8 Use $\|xBatch\|^m$ to construct adversarial input $\|xBatchAdv\|$;
1.9 (*lossReg*, *lossAcc*) = model($\|xBatchAdv\|^m$, $\|yTargetList\|^m$) in Ciphertext;
1.10 Update *maskTanh*, *maskUpsample* and *patternRaw* in Ciphertext;
1.11 Set $List_{lossReg}$.push(*lossReg*),$List_{lossAcc}$.push(*lossAcc*);
1.12 **end**
1.13 Process adamLR.step();
1.14 Obtain $lossRegAvg = mean(List_{lossReg})$;
1.15 Obtain $lossAccAvg = mean(List_{lossAcc})$;
1.16 **if** *(lossAccAvg* $\geq$ *attackThreshold) and (lossRegAvg* $<$ *regBest)* **then**
1.17 *maskBest = maskTanh*;
1.18 *maskUpsampleBest = maskUpsample*;
1.19 *patternBest = patternRaw*;
1.20 *regBest = lossRegAvg*;
1.21 **end**
1.22 Check early stop;
1.23 Check *cost* modification;
1.24 **end**
1.25 Return *maskBest*, *patternBest*, *maskUpsampleBest*.

5.3 Security Analysis

When optimizing *mask* and *pattern*, we need to calculate the gradient of the input data. We use batch technique to input data for processing. Therefore, only the average value of each batch of input data is used when calculating the gradient, thereby ensuring the privacy of the input data. The steps marked 'in Ciphertext' in Algorithm 1 means that it is running among the computing servers based on ciphertext. Our work is based on the additional secret sharing, which means that our scheme is information-theoretically secure.

6 EXPERIMENTAL EVALUATION

6.1 Experiment Setup

We implement our MP-BADNet framework using the communication backend of Falcon [14]. We run our experiments on a Google Cloud Platform computer on Ubuntu 16.04 LTS with an Intel Xeon

Algorithm 2: Outlier Detection Algorithm

Input: $List_{mask}$
Output: $List_{Flag}$ and $List_{FlagL1Norm}$

Set $ConsistencyConstant = 1.4826$;
for *each mask in* $List_{mask}$ **do**
  Calculate its $L1$ norm to obtain $List_{L1Norm}$;
end
Calculate *median* and *mad* of $List_{L1Norm}$;
Update $mad = ConsistencyConstant * mad$;
for i *to NumLabels* **do**
  if $(List_{L1Norm}[i] > median)$ **then**
    continue;
  end
  if $abs(List_{L1Norm}[i] - median)/mad > 2$ **then**
    $List_{Flag}$.push(i);
    $List_{FlagL1Norm}$.push($List_{L1Norm}[i]$);
  end
end
Return $List_{Flag}$ and $List_{FlagL1Norm}$.

CPU @ 2.20GHz processor and 16GB RAM and use four processes to simulate four different participants. We evaluate MP-BADNet on the MNIST dataset.The network used in this paper is a 3-layer fully connected layer neural network proposed like Mohassel's scheme [11] and uses ReLU function as a activation function in each layer. The softmax function is used to obtain the output probability after the final output layer. In our protocol, we use fixed point encoding with 11 bits of precision and three different moduli $L = 2^l$, a prime number p and 2. In particular, we set $l = 2^5$ and $p = 67$.

We use the same method in BadNets [4] to perform a backdoor attack on the MNIST dataset. The attack modifies about 10% of the training dataset as the target label (we assume to set the target label to 6) and embeds the trigger which size for each target label is 4×4 pixels. Then we use the dataset with trojaned data for model training. The experiments results show that the trained DNN model can successfully classified for clean inputs, and for the trojaned data, it is recognized as the target label with a high probability.

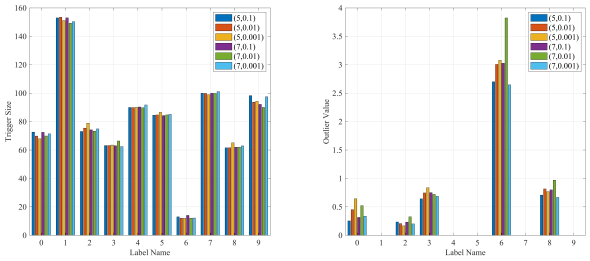
We find that after 15 epochs of the model training, for the clean dataset, the prediction accuracy of both plaintext and ciphertext is about 97.17% whereas for the dataset with triggers, the prediction accuracy of plaintext is 100%, and the ciphertext accuracy is 99.97%.

6.2 Detection Results

We run experiments with benchmark such as patience (Pat) and learning rate (LR). The former indicates the speed at which the optimization of *mask* and *pattern* ends, and the latter is the learning rate of the Adam optimizer. In our experiment, patience is set 5 and 7 as usual, and the learning rate is set 0.1, 0.01 and 0.001, respectively. The performance comparison between plaintext and ciphertext is shown in Table 1. The average time spent on the ciphertext detection is approximately 8 to 12 times that on the plaintext. In the case of ciphertext, the learning rate is 0.01 and the patience is 5, the time spent and the communication rate are the lowest.

Table 1: The performance comparison of plaintext and ciphertext.

Type	Pat	LR	Steps	Time (hour)	Comm.cost (GB)
Ciphertext	5	0.001	413	12.6373	14.9417
Ciphertext	5	0.01	291	8.9394	10.6276
Ciphertext	5	0.1	291	9.0391	10.6306
Ciphertext	7	0.01	404	12.3477	14.6202
Ciphertext	7	0.1	408	12.2586	14.7707
Plaintext	5	0.001	545	1.4385	–
Plaintext	5	0.01	346	0.8879	–
Plaintext	5	0.1	276	0.7281	–
Plaintext	7	0.01	425	1.0219	–
Plaintext	7	0.1	385	0.9996	–



(a) Trigger sizes (b) Outlier values

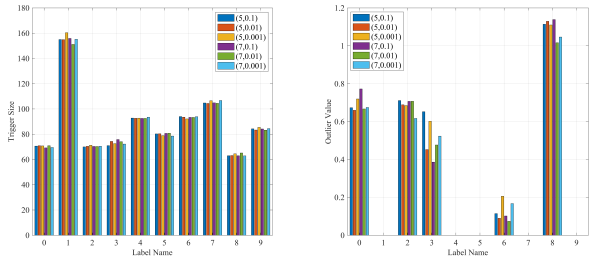
Figure 2: Reverse engineered for each label on dataset with trigger in ciphertext

The size of the reverse engineered trigger for each label is shown in Figure 2(a). The trigger size of label 6 is significantly smaller than other labels and the outlier value calculated greater than 2 are all label 6 as shown in Figure 2(b). Since the trigger size of labels 1, 4, 5, 7, and 9 is larger than the median, we show that the outlier value is 0 in Figure 2(b). Note that the parameters $(5, 0.1)$ means that the patience is 5 and the learning rate is 0.1.

Our experimental results show that MP-BADNet can successfully identify backdoor and target label on the trojoned data and ensure the privacy of input data and model parameters. The backdoor detection on the encrypted dataset takes about 8 to 12 times to the detection on the plaintext.

6.3 Comparison with clean model

The trigger size after reverse engineered under the clean model is shown in Figure 3(a). Compared with Figure 2, we can find that the trigger size of label 6 tends to be normal. As shown in Figure 3(b), the outlier values when patience is 5 and 7 are both less than 2 after executing Algorithm 2. Outlier values indicate that the model has no backdoor.



(a) Trigger sizes (b) Outlier values

Figure 3: Reverse engineered for each label on clean model in ciphertext

7 CONCLUSION AND FUTURE WORK

In this paper, we first propose a novel and generalizable technique for detecting and identifying backdoor attacks in a privacy-preserving DNN model that hides the triggers of backdoor attacks and is trained using MPC techniques. Our experiments is run in a semi-honest adversary environment. Using fixed-point arithmetic to train a DNN model with three servers, a DNN model that can protects data privacy and model privacy is obtained. In the privacy preserving DNN model, our experiments show that we can efficiently detect backdoor and identify target label in the case of four servers. And our scheme can ensure that the privacy of input data and model parameters is not leaked. In future, we will focus on the backdoor mitigation for privacy protection based on the triggers from reverse engineering and optimize the efficiency.

ACKNOWLEDGMENTS

This work is supported by National Natural Science Foundation of China (61972241, 61972094, 62032005), Natural Science Foundation of Shanghai (18ZR1417300), Luo-Zhaorao College Student Science and Technology Innovation Foundation of Shanghai Ocean University (A1-2004-20-201312, A1-2004-21-201311), and the young talent promotion project of Fujian Science and Technology Association.

REFERENCES

[1] R. Acharya, L. Oh, Y. Hagiwara, H. Tan, and H. Adeli. 2018. Deep convolutional neural network for the automated detection and diagnosis of seizure using EEG signals. *Computers in biology and medicine* 100 (2018), 270–278.

[2] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov. 2020. How to backdoor federated learning. In *International Conference on Artificial Intelligence Statistics*, Vol. 108. 2938–2948.

[3] D. Demmler, T. Schneider, and M. Zohner. 2015. ABY-A framework for efficient mixed-protocol secure two-party computation.. In *NDSS*. San Diego, CA.

[4] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg. 2019. Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access* 7 (2019), 47230–47244.

[5] F. Hampel. 1974. The influence curve and its role in robust estimation. *Journal of the american statistical association* 69, 346 (1974), 383–393.

[6] L. Huang, A. Joseph, B. Nelson, B. Rubinstein, and D. Tygar. 2011. Adversarial machine learning. In *Proceedings of the 4th ACM workshop on Security and artificial intelligence*. 43–58.

[7] D. Kingma and J. Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[8] K. Liu, B. Dolan-Gavitt, and S. Garg. 2018. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *International Symposium on Research in Attacks, Intrusions, and Defenses*. 273–294.

[9] Y. Liu, W. Lee, G. Tao, S. Ma, Y. Aafer, and X. Zhang. 2019. ABS: Scanning neural networks for back-doors by artificial brain stimulation. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 1265–1282.

[10] Y. Liu, S. Ma, Y. Aafer, W. Lee, J. Zhai, W. Wang, and X. Zhang. 2017. Trojaning attack on neural networks. (2017).

[11] P. Mohassel and Y. Zhang. 2017. Secureml: A system for scalable privacy-preserving machine learning. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 19–38.

[12] I. Stoica, D. Song, A. Popa, D. Patterson, M. Mahoney, R. Katz, A. Joseph, M. Jordan, J. Hellerstein, J. Gonzalez, et al. 2017. A berkeley view of systems challenges for ai. *arXiv preprint arXiv:1712.05855* (2017).

[13] S. Wagh, D. Gupta, and N. Chandran. 2019. Securenn: 3-party secure computation for neural network training. *Proceedings on Privacy Enhancing Technologies* 2019, 3 (2019), 26–49.

[14] S. Wagh, S. Tople, F. Benhamouda, E. Kushilevitz, P. Mittal, and T. Rabin. 2021. Falcon: Honest-majority maliciously secure framework for private deep learning. *Proceedings on Privacy Enhancing Technologies* 2021, 1 (2021), 188–208.

[15] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Zhao. 2019. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, Piscataway, NJ, 707–723.

[16] A. Yao. 1986. How to generate and exchange secrets. In *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*. IEEE, 162–167.

[17] X. Zhu, C. Vondrick, C. Fowlkes, and D. Ramanan. 2016. Do we need more training data? *International Journal of Computer Vision* 119, 1 (2016), 76–92.