Hindawi
Security and Communication Networks
Volume 2022, Article ID 2985308, 20 pages
https://doi.org/10.1155/2022/2985308

*Research Article*

# DeepGuard: Backdoor Attack Detection and Identification Schemes in Privacy-Preserving Deep Neural Networks

**Congcong Chen**,[1] **Lifei Wei**,[2] **Lei Zhang**,[1] **Ya Peng**,[1] **and Jianting Ning**[3]

[1]*College of Information Technology, Shanghai Ocean University, Shanghai 201306, China*
[2]*College of Information Engineering, Shanghai Maritime University, Shanghai 201306, China*
[3]*Fujian Provincial Key Laboratory of Network Security and Cryptology, College of Computer and Cyber Security, Fujian Normal University, Fuzhou 350117, China*

Correspondence should be addressed to Lifei Wei; lfwei@shou.edu.cn

Received 22 June 2022; Accepted 9 September 2022; Published 10 October 2022

Academic Editor: Jinguang Han

Deep neural networks (DNNs) have profoundly changed our lifeways in recent years. The cost of training a complicated DNN model is always overwhelming for most users with limited computation and storage resources. Consequently, an increasing number of people are considering to resort to a cloud for an outsourced DNN model training. However, the DNN models training process outsourced to the cloud faces privacy and security issues due to the semi-honest and malicious cloud environments. To preserve the privacy of the data and the parameters in DNN models during the outsourced training and to detect whether the models are injected with backdoors, this paper presents DeepGuard, a framework of privacy-preserving backdoor detection and identification in an outsourced cloud environment for multi-participant computation. In particular, we design a privacy-preserving reverse engineering algorithm for recovering the triggers and detecting the backdoor attacks among three cooperative but non-collusion servers. Moreover, we propose a backdoor identification algorithm adapting to single-label and multi-label attack detection. Finally, extensive experiments on the prevailing datasets such as MNIST, SVHN, and GTSRB confirm the effectiveness and efficiency of backdoor detection and identification in a privacy-preserving DNN model.

## 1. Introduction

Deep neural networks (DNNs) have made outstanding achievements in many fields and the DNNs-based applications are profoundly changing the aspects of our lives, such as medical diagnosis [1], autonomous driving [2], and image processing [3]. Most of the DNN models are generally obtained by training or refining existing models. In order to obtain a more accurate DNN model, it is often necessary to use a large amount of data for the model training [4]. Due to the limited capability of personal computers in model training, it is difficult for individual users to complete DNN training on personal computers. Thus, the users always outsource the model training to a cloud. However, outsourcing the training process of DNN models to cloud servers also risks privacy and security issues. It is well known that the training process of the DNN model consists of several steps, such as data collection, data processing, data training, storage, and use of model parameters. The more steps the training process of DNN models involves, the more significant privacy and security risks arise [5].

According to the privacy risks, the data may not be collected or used smoothly during data collection and data training due to sensitive data such as identity ID, health, property, and other information, or laws and regulations like the European Union General Data Protection Regulation (GDPR) and California Consumer Privacy Act (CCPA). In addition, the parameters of the trained DNN models are a great asset to the model owner after spending lots of time and money without privacy revealing [6]. According to the security risks, some recent studies have shown that the outsourced DNN models may be injected with backdoors

[7–9]. The DNN models injected with backdoors appear normal with a high probability to classify the clean data (i.e., no effect on classification accuracy). However, it misclassifies the poisoned data (clean data with triggers attached) as the prespecified label (target attack) or any incorrect label (untargeted attack). Backdoor attacks can be extremely damaging by identifying anyone as a specific person in the security guard systems. Moreover, backdoor attacks can also lead to significant risks in areas such as autonomous driving, voice recognition, and text recognition [10–12].

Many defense strategies have been proposed against backdoor attacks. There are two major types of defense strategies, model-based and data-based [9]. The former checks whether the model has been injected with a backdoor [13, 14] and the latter detects whether the input data has contained triggers [15]. These defense strategies have proven to be effective in detecting, identifying, and mitigating backdoor attacks in the plaintext domain, that is, directly in the data. However, these defense strategies may not work well in the ciphertext domain due to privacy-preserving requirements in the data and the DNN models. Secure multi-party computation (SMPC) seems to provide a possible solution for preserving data privacy, which originates from Yao's millionaire problem [16], allowing multiple participants to cooperatively calculate arbitrary functions without revealing the private input of the participants. Much previous work has focused on this area such as SecureML [17], Chameleon [18], ABY3 [19], Falcon [20], MP-SPDZ [21]. However, most of the schemes focus on the computation's privacy without considering backdoor defense strategies.

Therefore, there is a big gap between the backdoor attack in the plaintext and ciphertext domains. Figure 1 illustrates the scenarios between the backdoor attacks in different domains. In the plaintext domain, all computations are visible. While in the ciphertext domain, all computations are encrypted, this paper uses a three-participant secret sharing technique to preserve data and model privacy, where each participant holds a fragment of the data. Since the data is invisible, it brings many difficulties for backdoor defense.

Motivated by the above discussions, in this paper, we present, DeepGuard, a framework for privacy-preserving backdoor detection and identification in an outsourced cloud environment for multi-participant computation. We propose a model-based defense strategy that can detect backdoors in the ciphertext domain. In addition, most of the existing backdoor attack defense strategies are designed for single-label attacks and might be invalid for multi-label attacks. To detect these various attacks, we propose a novel backdoor identification algorithm. In summary, the contributions of this paper are summarized as follows:

(i) A defense strategy for backdoor attacks that works effectively in the ciphertext domain. We propose a backdoor attack detection algorithm based on MP-SPDZ [21] and NC [13] that can work in the ciphertext domain, which performs detection of backdoor attacks and ensures that the privacy of training data and DNN models are not compromised.

(ii) A backdoor identification algorithm for single-label and multi-label attacks. We propose a novel identification algorithm for single-label and multi-label attacks based on the forementioned backdoor detection results, which can effectively identify the specific attacked label.

(iii) Evaluating the effectiveness and efficiency of the proposed schemes. We validate the effectiveness and efficiency of our proposed schemes against both single-label and multi-label attacks by conducting extensive experiments on DNNs and state-of-the-art attack methods on the prevailing datasets such as MNIST, SVHN, and GTSRB.

## 2. Related Work

*2.1. Backdoor Attacks.* To the best of our knowledge, two advanced backdoor attack approaches are widely used to inject backdoors into target models. These two backdoor attack schemes are BadNets [7] and Trojan Attack [11]. Gu et al. [7] proposed BadNets to inject backdoor attacks by poisoning a training dataset. BadNets constructs a poisoned training dataset by adding triggers to randomly selected clean data and modifying its label to the target label. The Trojan Attack proposed by Liu et al. [11] differs from BadNets, i.e., the Trojan Attack does not allow the attacker to access clean datasets but can access pretrained DNN models. The Trojan Attack generates trojan triggers and training data by reverse engineering pretrained DNN models, and then uses the generated trojan triggers and training data to retrain the DNN model to inject the backdoor.

Besides, some more advanced backdoor attack approaches have been proposed in the past few years, Saha et al. [8] proposed a hidden random backdoor trigger injection technique. However, they required a larger trigger size to achieve a better attack effect. Gong et al. [9] proposed a backdoor attack approach that can resist four advanced defense strategies in an outsourced cloud environment. Bagdasaryan and Shmatikov [22] proposed blind backdoors that require neither access to the training data nor the model. Shokri [23] designed an adaptive adversarial training algorithm that optimizes the raw loss function of the model and maximizes indistinguishability of the poisoned data and clean data. Liu et al. [24] proposed a backdoor attack on DNN models with a high success rate by using mathematical modeling of the physical reflection model. Salem et al. [25] proposed a triggerless backdoor attack against deep neural networks based on the dropout technique, i.e., the attacker does not need to modify the input that triggers the backdoor. Yao et al. [26] consider backdoor attacks in transfer learning, in which all "student" models can inherit backdoors hidden in the "teacher" model, which poses a significant security threat.

*2.2. Backdoor Defenses.* Liu et al. [27] proposed the first effective defense scheme for DNN backdoor attacks, Fine-Pruning, which successfully defended against backdoor attacks using a combination of pruning and fine-tuning. In

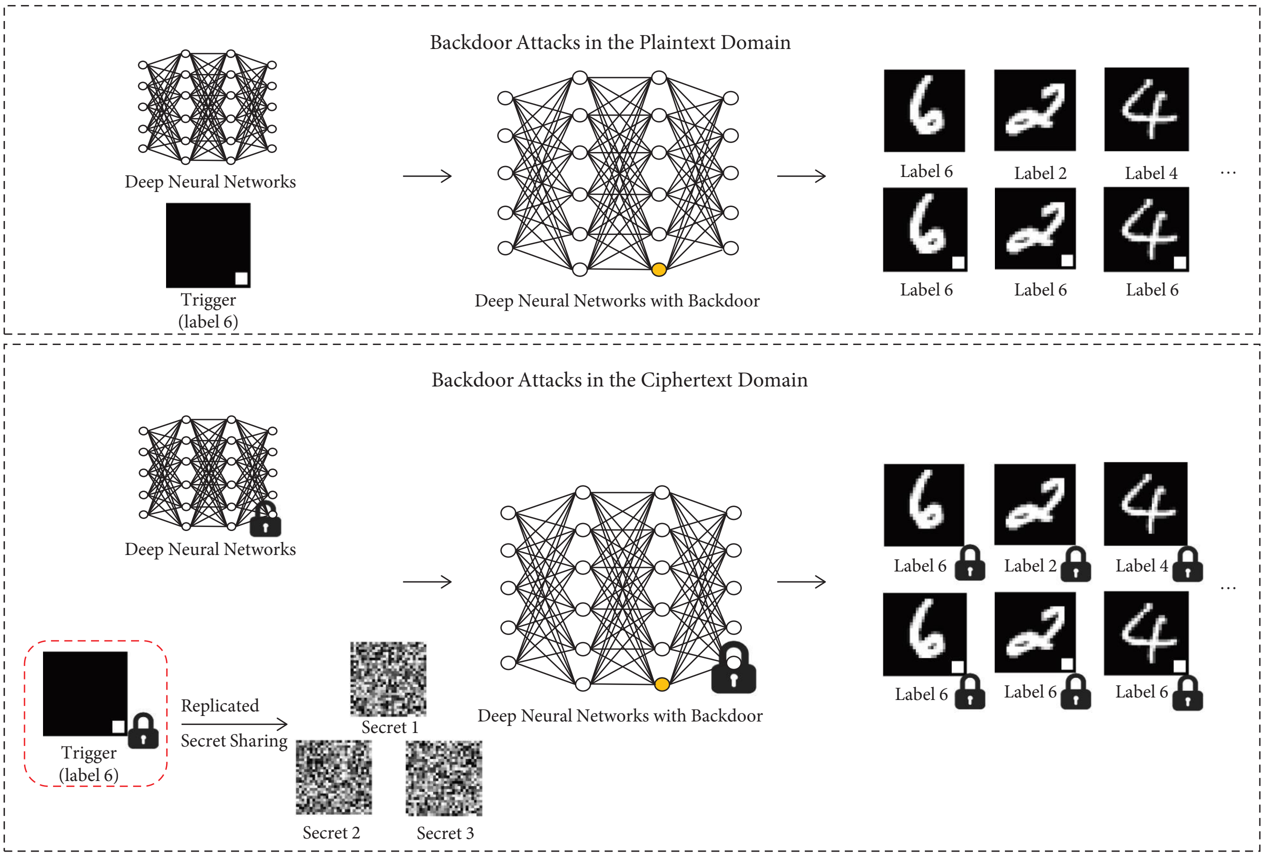


Figure 1: The difference between the backdoor attacks in the plaintext and ciphertext domains.

2019, Wang et al. [13] constructed triggers by reverse engineering methods and analyzed their outliers through the triggers and eventually pruned neurons based on this constructed trigger to reduce the success rate of backdoor attacks. Liu et al. [14] analyzed internal activation values by introducing different levels of stimulation to neurons to detect whether the DNN model is being attacked by a backdoor. Gao et al. [15] detected triggers by overlaying input samples and measuring the entropy distribution of output results. Chen et al. [28] used a conditional generative model to learn the probability distribution of potential triggers from the model to detect and defend against backdoor attacks. Du et al. [29] demonstrated the effectiveness of differential privacy for outlier detection and extended the technique to backdoor attack detection. Qiu et al. [30] applied a data augmentation policy to eliminate the effectiveness of backdoor attacks and an augmentation policy to preprocess the input samples to invalidate the triggers in the inference phase. Shen et al. [31] found that the defense complexity of existing methods is quadratic with the number of class labels. They propose a more efficient scheme that reduces the complexity of the defense method by the $K$-Arm optimization method, allowing to handle models with many classes. Wang et al. [32] found that the representations of authentic and poisoned data against the target class are embedded in different linear subspaces. Therefore, based on the coherence optimization problem, they proposed the PiDAn algorithm to detect backdoors.

*2.3. Privacy-Preserving Machine Learning.* Mohassel and Zhang [17] present SecureML, the first privacy-preserving machine learning training scheme in a two-party computation model. Chameleon [18] used a semi-honest third party to replace Beaver triples to reduce the amount of communication. In addition, they greatly improved the practicality and scalability of ABY [33]. ABY3 [19] proposes a new scheme in semi-honest and malicious environments that allows arithmetic sharing, Boolean sharing, and Yao sharing to be efficiently converted among the three cooperative and non-collusion participants. Zhang et al. [34] proposed DeepPAR and DeepDPA protocols to preserve the input privacy and model parameter privacy of models in deep learning. MP-SPDZ [21] proposes an SMPC framework that greatly simplifies the cost of comparing different protocols and security models. Its underlying cryptographic primitives include secret sharing, oblivious transfer, homomorphic encryption, and garbled circuits. Subsequently, a series of efficient approaches such as Falcon [20], ASTRA [35], FLASH [36], Trident [37], and ABY2 [38] have focused on privacy-preserving machine learning.

*2.4. Comparison with Other Defense Strategies.* A comparison with other defense strategies against backdoor attacks is shown in Table 1. FLGUARD [44] requires submodel clustering to exclude submodels with high attack rates, while Safe Learning [45] excludes anomalous submodels by user random combination. They are partially compliant with the white-box item.

Table 1: Comparison with other backdoor defense strategies.

| Approaches | Privacy protection approach | Model access | | Privacy | | Backdoor detection type | | Backdoor attack defense types | |
|---|---|---|---|---|---|---|---|---|---|
| | | Black-box | White-box | Data privacy | Model privacy | Model-based | Data-based | Single-label attacks | Multi-label attacks |
| SentiNet [39] | None | ● | ○ | ○ | ◐ | ○ | ● | ● | ● |
| Strip [15] | None | ● | ○ | ○ | ◐ | ○ | ● | ● | ● |
| NIC [40] | None | ○ | ● | ○ | ○ | ○ | ● | ● | – |
| AC [41] | None | ○ | ● | ○ | ○ | ● | ○ | ● | – |
| NC [13] | None | ● | ○ | ○ | ◐ | ● | ○ | ● | ◐ |
| ABS [14] | None | ○ | ● | ○ | ○ | ● | ○ | ● | ◐ |
| DeepInspect [28] | None | ● | ○ | ◐ | ◐ | ● | ○ | ● | ◐ |
| TABOR [42] | None | ● | ○ | ○ | ◐ | ● | ○ | ● | ● |
| Auror [43] | Differential privacy, Federal learning | ● | ○ | ● | ● | ○ | ● | ● | – |
| FLGUARD [44] | Secret sharing, Federal learning | ○ | ◐ | ● | ● | ● | ○ | ● | ● |
| SAFELearning [45] | Secret sharing, Federal learning | ○ | ◐ | ● | ● | ● | ○ | ● | – |
| CRFL [46] | Differential privacy, Federal learning | ○ | ● | ● | ● | ● | ○ | ● | – |
| MP-BADNet [47] | Replicated secret sharing, SMPC | ● | ○ | ● | ● | ● | ○ | ● | ○ |
| Ours | Replicated secret sharing, SMPC | ● | ○ | ● | ● | ● | ○ | ● | ● |

*Note.* '●' indicates that the item meets, '○' indicates that the item does not meet, '◐' indicates that the item partially meets, and '–' means that the item has not been experimentally proven and it is not possible to determine whether the item meets. The model access indicates the level of access to the model parameters of the backdoor attack detection approach. The privacy indicates the level of privacy preservation of the backdoor attack detection approaches. The backdoor detection type indicates whether it is data-based or model-based, with the former detecting whether the data contains triggers and the latter detecting whether the model is injected with backdoors. And the backdoor attack defense types indicate whether the approach is resistant to single- or multi-label attacks.

DeepInspect [28] is used to invert a subset of the training data through the model inversion, which can be seen as partially preserving the privacy of the data. SentiNet [39], Strip [15], NC [13], DeepInspect [28], and TABOR [42] can be seen as partially preserving model privacy because they are black-box access. NC [13] may not succeed for multiple target labels. ABS [14] and DeepInspect [28] have shown that a small number of multi-label attack detection may fail. Auror [43] and CRFL [46] are considered for backdoor defense in federated learning but are not applicable in privacy-preserving outsourced machine learning. MP-BADNet [47] can resist backdoor attacks against privacy-preserving DNN models but only supports single-label attacks and cannot effectively defend against multi-label attacks. Besides, an additional secondary server is required to participate in the computation.

## 3. Overview

*3.1. Key Ideas.* It is known from the previous discussion that many backdoor defense strategies and privacy-preserving machine learning schemes have been proposed, respectively. To better explain the motivation for this paper and the challenges in the ciphertext domain, we give the answers to the following questions for further elaboration.

(1) Why do we need to preserve privacy? Nowadays, the data is always distributed among different data owners, such as different companies, organizations, and individuals, who need to train a model together for practical reasons. However, with the increasing privacy awareness, the data owners are reluctant to disclose their sensitive data or restrict sharing their sensitive data due to the laws and regulations (e.g., GDPR and CCPA). Moreover, these data owners often have limited computing and storage resources for economic reasons and prefer to outsource their models to cloud servers for training. As a result, the ownership and management of the data are separated. Meanwhile, cloud service providers are usually untrustworthy and try to find out private data. Therefore, how to use private data in the DNNs model training while keeping the data available and invisible has become an urgent problem.

(2) Why choose SMPC? Generally speaking, the techniques such as homomorphic encryption (HE) and SMPC are always used to preserve the privacy of data and models in an outsourcing environment. However, HE is unsuitable for training DNN models due to its vast computational burdens. That is why this paper prefers to use the SMPC technique for the privacy preservation of data and models. Apart from smaller computation, SMPC is naturally suitable for multiuser participation scenarios.

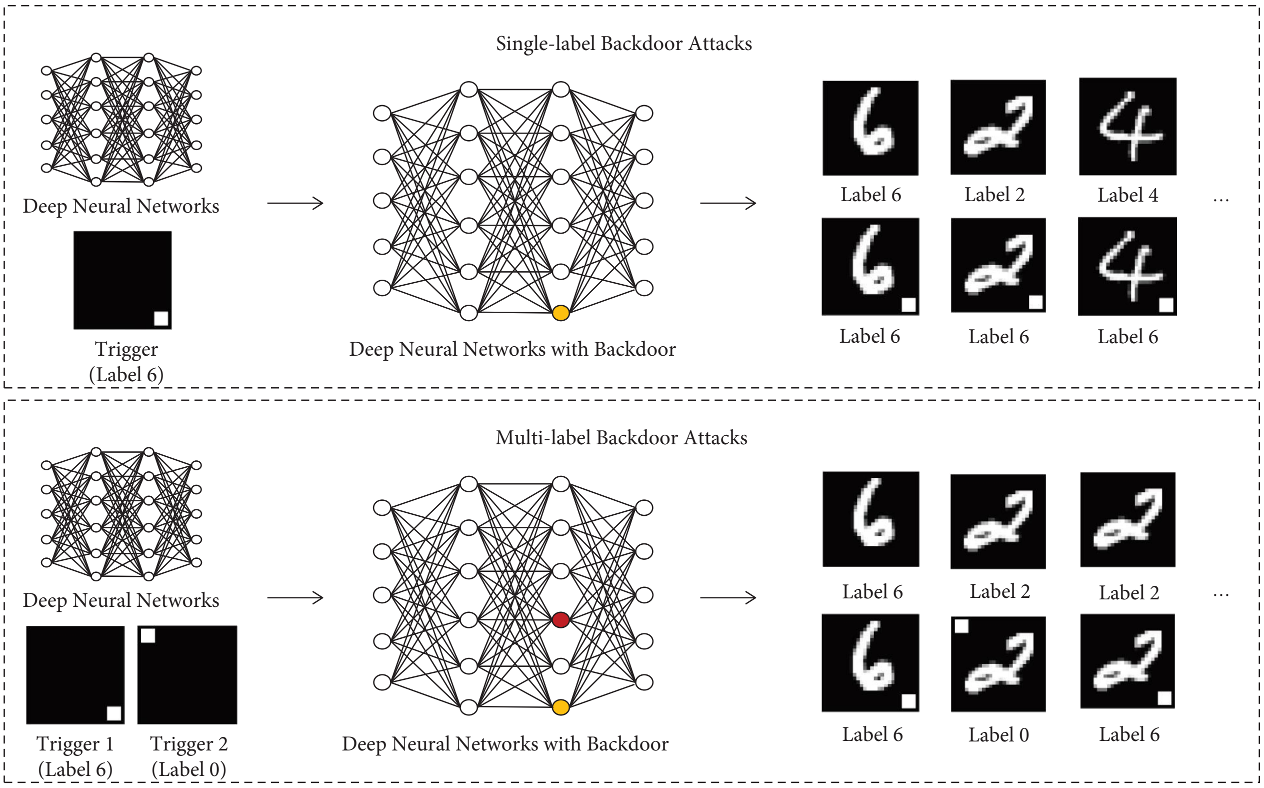


Figure 2: The single-label and multi-label backdoor attacks. The single-label attacks can classify arbitrary images into target labels. The multi-label attacks can hide multiple backdoors in the complicated DNN models, with different triggers corresponding to different target labels.

(3) Why choose the SMPC technique with three participants? The SMPC technique between two participants is more suitable for two data owners to jointly train models without compromising privacy due to the underlying technology and is not suitable for outsourced computation scenarios. In addition, the time and communication overhead over three participants grows linearly with the number of participants. The SMPC technique with three participants is the best choice for our outsourced computing.

*3.2. Threat Model.* In this paper, we assume that three participants cooperatively train a DNN model using SMPC based on replicated secret sharing technique [48]. We consider the passive adversary model, which means that the corrupted participant is semi-honest and will not actively deviate from the protocol and launch a maliciously active attack but might try to snoop and obtain the private data from the other participants. It is a common adversarial setting considered in previous SMPC schemes [17–19]. Moreover, we assume that the participants do not collude since if the collusion appears, the colluding participants could recover private data through the fragments they hold.

For the backdoor attack, we assume that malicious users can modify their data before training the model. However, malicious users cannot modify other users' data and cannot manipulate the DNN training process. The goal of the malicious user is to embed a backdoor in the DNNs model when it is trained. A DNNs model with an embedded backdoor will output normally for clean data but will output malicious behavior specified by the malicious user for poisoned data. The defenders (the three participants in this paper) can access the data fragments and model fragments in the ciphertext domain, but not the complete data and model.

The experiments in [9] demonstrate that a certain ratio of data is sufficient for the trained DNN model to be backdoor attacked. Our basic attack model, named as single-label attacks, is consistent with [7, 13] in which it uses a white square as a trigger attached to the bottom right corner in a clean image to produce a poisoned data (the clean image is attached with a trigger) as shown in Figure 2. We believe that a successful backdoor attack does not affect the classification accuracy of the clean data, but it has a high probability to misclassify for the poisoned data. In the advanced attack model, named as multi-label attacks, we implement in this paper considering that the triggers appear at different locations in one image as shown in Figure 2.

*3.3. Framework of DeepGuard.* As shown in Figure 3, we assume that users want to outsource training the DNN model to the cloud servers due to the limitation of time or economic cost, and at the same time, do not want the cloud servers to learn the private data and parameters from the DNN model.

It is assumed that data are from different users or belongs to different data owners. Some of the data might be poisoned

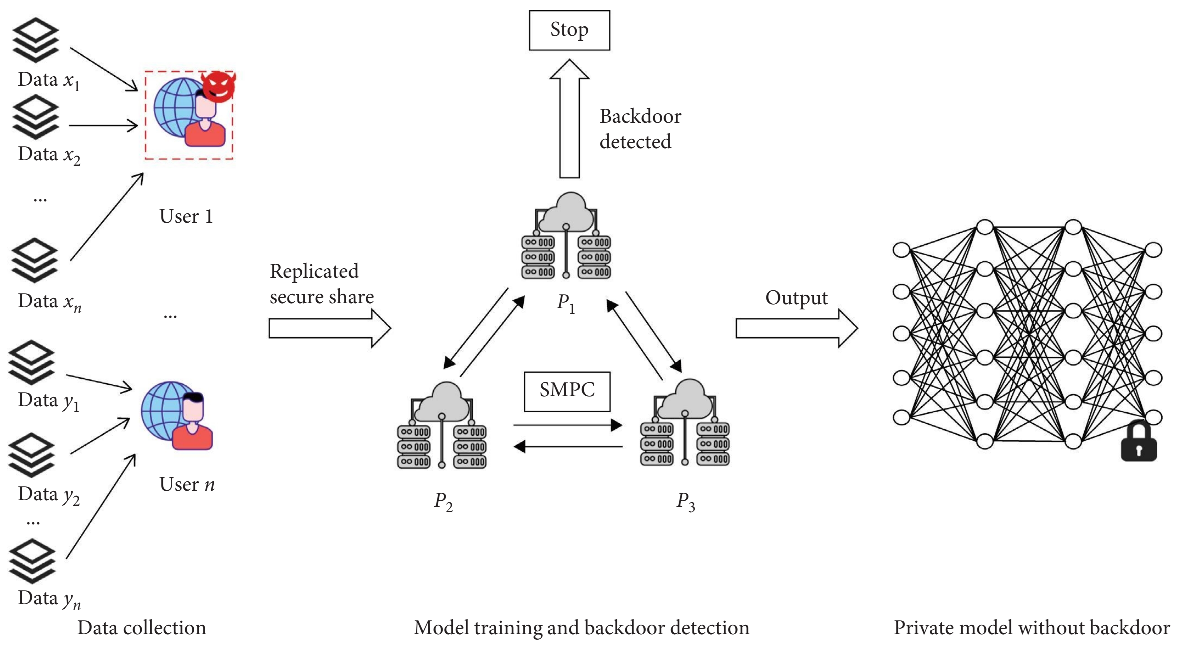


Figure 3: The framework of DeepGuard. The data are from different users or belong to different data owners. The triggers may be attached to the original data, which is trained in an SMPC-based DNN model through replicated secret sharing in three servers. When a backdoor attack is detected, the training is stopped, and the user is warned. Otherwise, a privacy-preserving DNN model is output.

Table 2: List of symbols.

| Symbol | Description |
|---|---|
| $\Delta$ | When the target label corresponds to a backdoor, the minimum perturbation is required to map the other dimensional space to the target dimensional space. |
| $\delta$ | When the target label corresponds to a nonbackdoor, the minimum perturbation is required to map the other dimensional space to the target dimensional space. |
| $\pi$ | The trigger was used to construct the poisoned data. |
| $\alpha$ | The attack threshold, i.e., the backdoor attack success rate, takes the value from 0 to 1. |
| $\eta$ | The parameter MinSamples in the DBSCAN algorithm. |
| $\epsilon$ | The parameter epsilon in the DBSCAN algorithm. |
| $\beta$ | The threshold is used to adjust the larger mean value. |

in the source by malicious users and used in the model training to inject a backdoor into the model.

The cloud server consists of $P_1$, $P_2$, and $P_3$, referred to as the participants in this paper. Each of these three participants holds the private data shares by the replicated secret sharing from the users and wants to collaborate to train a DNN model. In the trained DNN model, no specific parameters are accessible to any parties.

As the backdoored model has a high classification accuracy for clean datasets, it is difficult for the users to detect that the model is subject to a backdoor attack only by themselves. According to the privacy-preserving requirements, the honest users need to turn to all three participants simultaneously to perform the detection of the backdoor attack in the ciphertext domain to detect whether the DNN model is injected with a backdoor.

## 4. Concrete Construction

*4.1. List of Symbols.* Table 2 describes the symbols used in this section.

*4.2. Overall Design.* We define the poisoning data as follows:

$$\begin{aligned} \left\| x^* \right\| &= T(\|x\|, \|\text{mask}\|, \|\text{pattern}\|) \\ &= (1 - \|\text{mask}\|) \cdot \|x\| + \|\text{mask}\| \cdot \|\text{pattern}\|, \end{aligned} \tag{1}$$

where $\|\cdot\|$ denotes that the private data are held by three participants through replicated secret sharing technique [48] and thus single party is not able to access the private data. $x^*$ denotes poisoned data, and $T(\cdot)$ denotes a function that attaches a trigger to the original image $x$. We use mask to denote a 2D matrix that determines the size and position of the trigger over the original image, which takes values in the range $(0, 1)$. We use pattern to represent the trigger pattern, which is a matrix of the same size as the original image. The parameters $\text{mask}_{i,j}$ and $\text{pattern}_{i,j}$ represent the mask and pattern values of row $i$ and column $j$, respectively. When $\text{mask}_{i,j}$ equals to 1, it means that the value of row $i$ and column $j$ in the original image are completely covered by those row $i$ and column $j$ in the pattern. When $\text{mask}_{i,j}$ equals 0, it means that the original image is not covered at all.

As in [13], we consider the classification problem as the creation of partitions in a multidimensional space in ciphertext, where each dimension contains certain features, i.e., each dimension can be considered to represent a label. Embedding backdoors in DNN models is to create a "shortcut" from other dimensional spaces to the target dimensional space. Detecting backdoors in a DNN model can be thought of as detecting the minimum perturbation needed to get from the other dimensional space to the target dimensional space. If the target label corresponds to a backdoor, then let the minimum perturbation required to map the other dimensional space to the target dimensional space corresponding to the backdoor be $\Delta$. If the target label corresponds to a no backdoor, then let the minimum perturbation required to map the other dimensional space to the corresponding target dimensional space be $\delta$.

Let the trigger used to construct the poisoned data be $\pi$. Since the trigger needs to be as inconspicuous as possible (e.g., a white square in the corner of the image) in order to avoid being easily detected, it is natural to have that $\text{size}(\pi) \ll \text{size}(\delta)$. For single-label attacks, since $\Delta$ is the smallest perturbation, i.e., the corresponding valid part of the trigger, we can derive $\text{size}(\Delta) < \text{size}(\pi)$. For multi-label attacks, we can draw similar conclusions. Thus, we only need to analyze $\text{size}(\Delta)$ to know whether the model is attacked by a backdoor.

*4.3. The Reverse Engineering Algorithm.* To obtain $\Delta$ for each label, we designed an SMPC-based reverse engineering algorithm for obtaining triggers based on a clean dataset. There are three targets in this algorithm, that is, (1) to preserve data and model parameter privacy, (2) to find the shape of the trigger by backdoor attacks, i.e., $(\text{mask}, \text{pattern})$ that can misclassify other labels as the target label, and (3) to find the smallest possible trigger, i.e., to find the smallest $(\text{mask}, \text{pattern})$ that can produce a misclassification. To further achieve the third goal, we use the $L_1 -$ norm of the mask to measure the size of the potential trigger. The objective function can be defined formally as follows:

$$\begin{aligned} h(\|x\|) = \min_{(\|\text{mask}\|),(\|\text{pattern}\|)} L(\|y\|), f(\|x\|), \\ (\|\text{mask}\|, \|\text{pattern}\|) + \text{cost} \cdot |\text{mask}|, \end{aligned} \tag{2}$$

where $L(\cdot)$ denotes the loss function, which is the cross-entropy loss function used in this paper. $y$ is the ground truth, $f(\cdot)$ denotes the DNN model, and cost denotes the coefficient of the third objective of the optimization. To speed up the convergence of reverse engineering, we use the Adam Optimizer [49] to obtain the optimal mask and pattern.

Algorithm 1 describes the process of recovering potential triggers. Step 1 is the initialization operation. Steps 2–18 are to reverse the model using clean dataset to obtain mask and pattern. Steps 4–8 are executed by splitting the clean dataset into several batches, which number is *miniBatch*. Step 5 is to construct the poisoned data, and step 6 is to take the poisoning data into the model to calculate the loss and the accuracy. Step 10 is to update the learning rate of the Adam optimizer, and steps 12–15 to determine whether it achieves the optimal for $(\text{mask}, \text{pattern})$. Step 16 detects whether the algorithm stops, which is determined by the accuracy of the algorithm reaches the threshold several times.

In Algorithm 1, $\|\cdot\|$ indicates that the value is in the ciphertext domain. When $\alpha$ is set to 1, it means that the trigger is capable to cause all clean images to be misclassified as target label. $\alpha$ affects the effectiveness of the backdoor attack and the final size of the recovered trigger. *lossBest* denotes the optimal value of equation (2). When the trigger is recovered for a potential label, the value *cost* is first initialized to 0. If the recovered trigger causes other labels to be classified as the potential label with a success rate higher than the threshold $\alpha$, the *cost* is adjusted upward to reduce the size of the trigger. Otherwise, the *cost* value is adjusted downward to enlarge the size of the trigger. In short, the larger *cost*, the lower the success rate of the attack and the smaller the trigger. The algorithm stops until the success rate of classifying other labels as potential labels are relatively stable. In this paper, the values mask and pattern are initialized with random values, and then optimized according to equation (2). To enable the reverse engineering algorithm to accommodate a certain amount of error, $\alpha$ is set to 0.99. *lossBest* is initialized to infinity because the loss needs to be updated according to the calculation of Equation (2).

*4.4. The Backdoor Identification Algorithm.* In backdoor identification, we analyze the size of the trigger by the $L_1 -$ norm of mask. The idea is to use an efficient clustering algorithm to classify the labels into two categories: benign and poisoned, since the potential triggers corresponding to the labels might be benign or poisoned. Naturally, we first try to use the $K$-means clustering algorithm for classification. It was found through experiments that the $K$-means clustering algorithm could not identify clean models (i.e., only one class of benign labels) and could not correctly classify scenarios in which the sizes of the triggers corresponding to the labels differed significantly. Since the recovered trigger sizes are relatively dense, we consider using the density-based classification algorithm. The Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm [50] is one of the typical algorithms based on density clustering. The DBSCAN algorithm assumes that the closeness of the sample distribution determines the categories and does not require prespecifying the cluster size for clustering, which performs better than $K$-means in backdoor identification.

Since the value mask computed by the reverse engineering algorithm is relatively dense, we chose to use the DBSCAN algorithm for outlier analysis. The parameters $(\text{epsilon}, \text{MinSamples})$ in the DBSCAN algorithm describe the closeness of the sample distribution in the neighborhood. Where epsilon describes the neighborhood distance threshold for a given sample and MinSamples describes the threshold for the number of samples in the neighborhood for a given sample with distance epsilon. Thus, it is crucial to determine the parameters $(\text{epsilon}, \text{MinSamples})$. In the

```
     Input: $\|x\|$, $\|\text{target}\|$
     Output: $\min(\|\text{mask}\|, \|\text{pattern}\|)$
 (1) Initialize $\|\text{mask}\|$, $\|\text{pattern}\|$, let $\alpha = 0.99$, lossBest $= \infty$.
 (2) while 1 do
 (3)   Initialize LossList = [], LossAcc = [];
 (4)   for $i$ in mini Batch do
 (5)     $\|\text{TrojanInput}_i\| = (1 - \|\text{mask}\|) \times \|x_i\| + \|\text{mask}\| \times \|\text{pattern}\|$;
 (6)     Calculate $\|\text{loss}\|$ and $\|\text{acc}\|$ based on $\|\text{target}\|$ and $\|\text{TrojanInput}_i\|$;
 (7)     Update $\|\text{mask}\|$, $\|\text{pattern}\|$;
 (8)     LossList.append ($\|\text{loss}\|$), Loss Acc.append ($\|acc\|$);
 (9)   end for
(10)   Update Adam learning rate $LR$;
(11)   $\|\text{accAvg}\|$ = mean (LossAcc), $\|\text{lossAvg}\|$ = mean (LossList);
(12)   if ($\|\text{accAvg}\| > \alpha$) and ($\|\text{lossAvg}\| <$ lossBest) then
(13)     $\min(\|\text{mask}\|, \|\text{pattern}\|) = (\|\text{mask}\|, \|\text{pattern}\|)$;
(14)     lossBest = $\|\text{lossAvg}\|$;
(15)   end if
(16)   Check whether the algorithm ends early;
(17)   Update cost;
(18) end while
(19) Return $\min(\|\text{mask}\|, \|\text{pattern}\|)$.
```

Algorithm 1: Reverse engineering algorithm.

```
     Input: List Mask, Labels
     Output: Targetlabel
 (1) Initialize $\epsilon$, $\eta$, $\beta = 4 * \epsilon$, TargetLabel = [];
 (2) Calculate mean = mean (ListMask);
 (3) if mean $> \beta$ then
 (4)   for $i$ in Labels do
 (5)     while ListMask[$i$]/$\beta > 1.1$ do
 (6)       ListMask[$i$] = List Mask[$i$]/1.1;
 (7)     end while
 (8)   end for
 (9) else
(10)   for $i$ in Labels do
(11)     while ListMask[$i$] $> \beta$ do
(12)       List Mask[$i$] = ListMask[$i$]/1.1;
(13)     end while
(14)   end for
(15) end if
(16) outlier_detection = DBSCAN(min_samples = $\eta$, eps = $\epsilon$);
(17) clusters = outlier_detection.fit_predict (ListMask);
(18) $a$ = [], $b$ = [];
(19) for $i$ in Labels do
(20)   if clusters[$i$] == 0 then
(21)     $a$.append ($i$, ListMask[$i$]);
(22)   else
(23)     $b$.append ($i$, ListMask[$i$]);
(24)   end if
(25) end for
(26) if len($a$) > 0 and len($b$) > 0 then
(27)   TargetLabel = mean($a$) < mean($b$)?$a$: $b$;
(28) end if
(29) Return TargetLabel.
```

Algorithm 2: Backdoor identification algorithm.

following, we use $\epsilon$ to denote epsilon and $\eta$ to denote MinSamples for short.

Algorithm 2 is the backdoor identification algorithm. Step 1 is to initialize the parameters $(\epsilon, \eta)$ and the threshold $\beta$. Step 2 is to calculate the mean value of the mask. Steps 3–15 are to prevent larger mask values from being identified as outliers. If the mean value is greater than $\beta$, the values mask greater than the mean value are reduced to around the mean value to increase the density of the data. Steps 16–17 are used for clustering analysis using the DBSCAN algorithm. Steps 18–25 correspond to the clustering result analysis. The clustering results are generally classified into $(-1, 0)$, $(0, 1)$, and $(-1, 0, 1)$, where $-1$ represents the anomaly, 0 represents the first category of clustering results, and 1 represents the second category of clustering results. When the backdoor attack is a kind of single-label attack or multi-label attack with a relatively small number of labels, the classification result is generally $(-1, 0)$. When the multi-label attacks have a large number of labels, the classification result is generally $(0, 1)$ or $(-1, 0, 1)$. Therefore, in steps 26–28, we analyze the specific attacked labels by comparing the mean of the two categories of classification results.

### *4.5. Security Analysis*

*4.5.1. Channel Security.* We assume that a secure channel is established among the three participants by each other to ensure that the transmitted information is not corrupted. The communication private key in this secure channel is securely stored and cannot be easily disclosed.

*4.5.2. Privacy-Preserving.* In this scheme, we assume that the participants (i.e., the cloud servers providing computing services) are semi-honest, and do not actively deviate from the protocol. Different users or data owners provide the datasets used for model training and backdoor detection. They send the datasets to the participants via replicated secret sharing technique. Eventually, each participant only holds fragments and is assumed not to collude with others to recover the data. Therefore, the different users or data owners cannot have access to the private data of other users and the participants cannot have access to the complete private data. Moreover, the privacy-preserving DNN model is trained by the interaction between participants using replicated secret sharing technique, and does not reveal the privacy of the model parameters.

## 5. Experiment and Evaluation

We evaluate the effectiveness and efficiency of the proposed framework by training a DNN model and embedding two types of backdoor attacks (single-label and multi-label attacks). The experiment performed all the benchmarks on a server with two 16-core 2.10 GHz Intel(R) Xeon(R) Gold 6130 CPUs, 256 GB RAM, and Ubuntu 16.04.4 LTS.

As shown in Figure 4, in the single-label attacks scenario, a trigger is targeted at one label and thus there exists only one trigger in the DNN model. The model makes classification normally for a dataset with all clean data. However, for the poisoned data (those data with a trigger), the model always classifies it to the prespecified label regardless of the original label. In the multi-label attacks, a DNN model is attacked by a backdoor containing multiple attacked labels representing the backdoor attacks against different labels at different locations. The trigger is squares of $3 \times 3$ pixels or $4 \times 4$ pixels. To make the multi-label attacks effective, the poisoning rate of each poisoned label is about 25%. The following equation defines the proportion $R_{\text{poison}}$ of the total poisoned data to the dataset.

$$R_{\text{poison}} = \frac{R_{\text{label}} \times N_{\text{label}}}{R_{\text{label}} \times N_{\text{label}} + 1}, \tag{3}$$

where $R_{\text{label}}$ denotes the poisoning rate of each poisoning label and $N_{\text{label}}$ denotes the number of poisoning labels. Based on the above settings, the steps of each epoch when training the DNN model is as follows:

$$S_{\text{train}} = \frac{1}{1 - R_{\text{poison}}} \times |D|, \tag{4}$$

where $S_{\text{train}}$ denotes the number of training steps per epoch during the DNN model training and $|D|$ denotes the size of the dataset.

### *5.1. Backdoor Detection in the Ciphertext Domain*

*5.1.1. Experimental Setup.* The experimental is built on the primitive provided by the MP-SPDZ [21] library, and all arithmetic sharing of secret data is performed on modulo $2^{128}$. For the ciphertext, we use three different processes to simulate three participants, and they perform calculations through the replicated secret sharing technique. In this section, we only consider using the MNIST dataset [51] and single-label attacks (supposing that the target label is 6). The architecture of the DNN model we evaluated is consistent with [17], which is a simple network consisting of 3 fully connected layers with a Rectified Linear Unit (ReLU) activation function. Finally, the softmax function is used to calculate the output probability of the labels. During training, the basic learning rate is set to 0.1, the number of epochs is set to 15, the batch size is 128, and the stochastic gradient descent (SGD) momentum is set to 0.9. The accuracy of the clean model is approximately 97%, both in the ciphertext and plaintext domains. Furthermore, in this network structure, we use triggers that are $4 \times 4$ pixel squares, as shown in Figure 4(a).

*5.1.2. The Effectiveness of Backdoor Detection.* Figure 5 shows the trigger sizes calculated by the reverse engineering algorithm from the different models, such as (a) the clean model and (b) the injected backdoor model, with Plaintext and Ciphertext representing the trigger sizes in plaintext (NC [13]) and ciphertext (ours), respectively. The sizes of the plaintext and ciphertext datasets used in our reverse

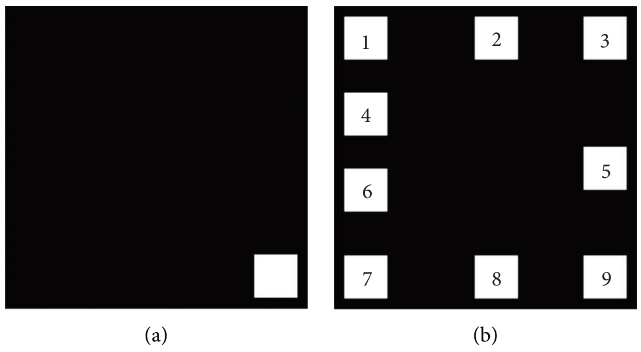


Figure 4: The trigger used in the paper. (a) Denotes the trigger in single-label attacks located at the bottom right corner and (b) denotes the possible attack locations of the trigger in multi-label attacks.

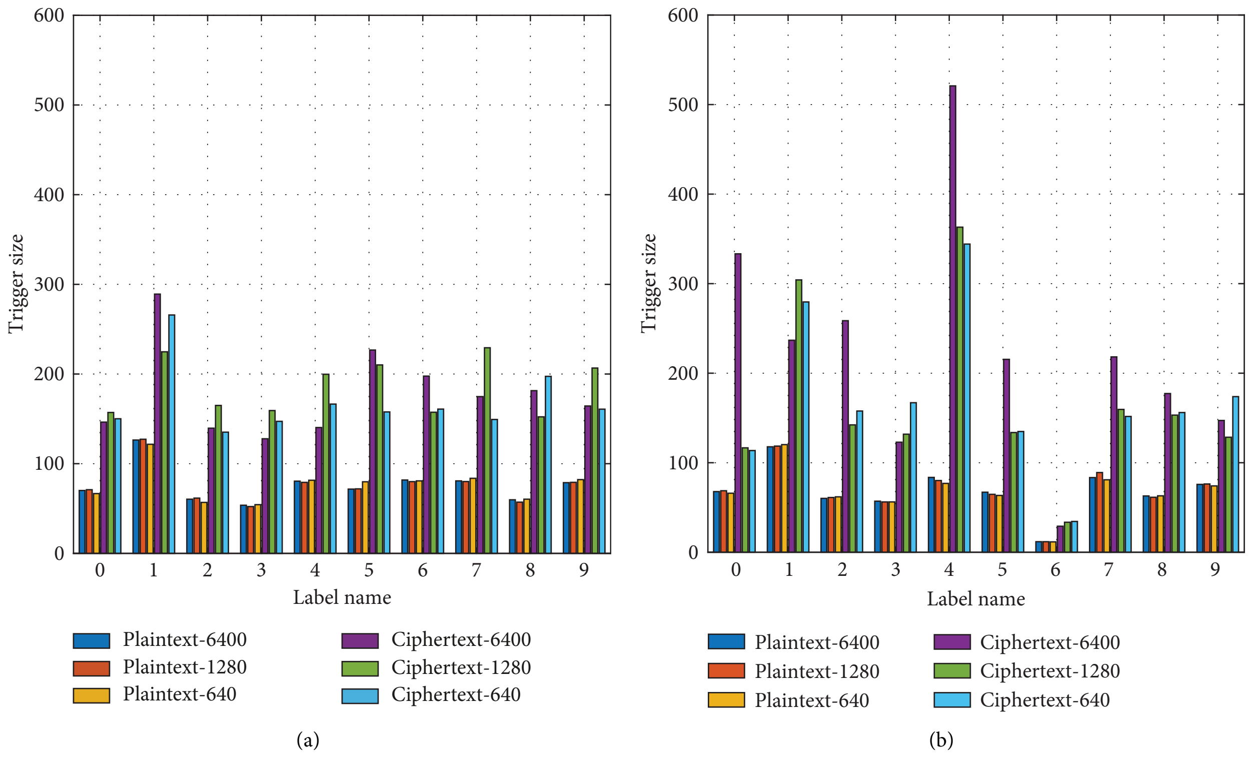
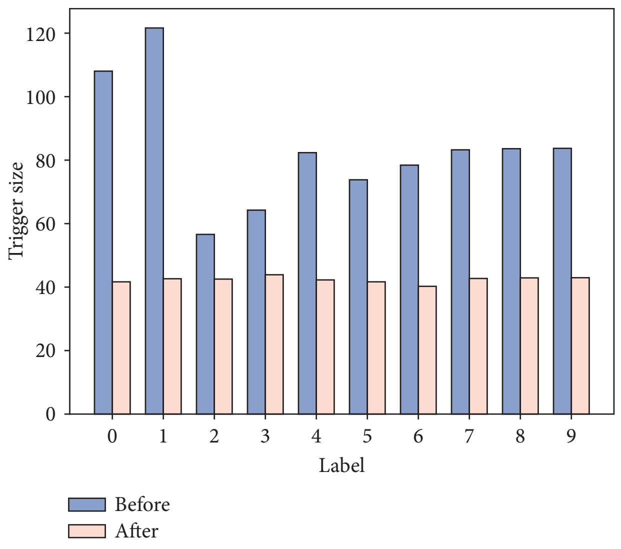


Figure 5: Trigger sizes calculated by reverse engineering algorithms. (a) Denotes the trigger size in the clean model and (b) denotes the trigger size in the poisoned model (assuming the target label is 6).

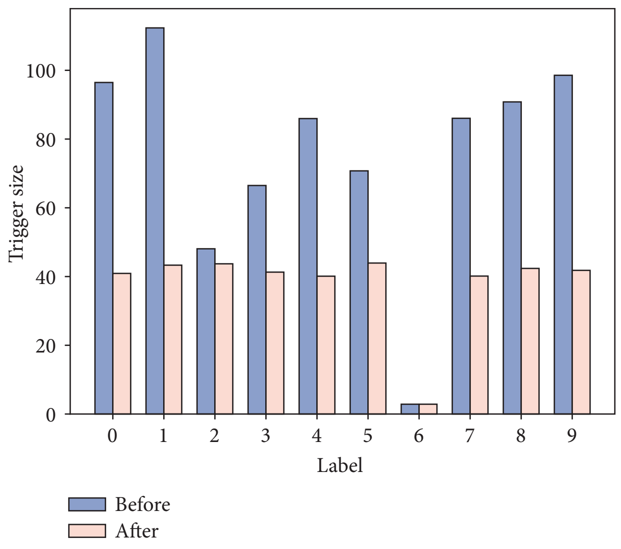
Table 3: The running time for reverse engineering algorithm in the ciphertext domain.

| | Reverse engineering | | | | | |
|---|---|---|---|---|---|---|
| Time (s) | Clean model | | | Poisoned model | | |
| | 6400 | 1280 | 640 | 6400 | 1280 | 640 |
| Average | 27952.00 | 6041.90 | 3015.50 | 31886.80 | 6638.60 | 2896.40 |

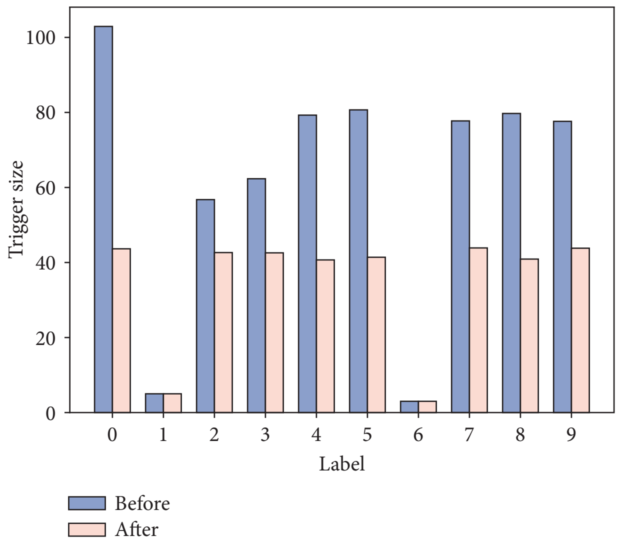
engineering experiments are 6400, 1280, and 640, respectively. In this section, we only show the results of single-label attacks (the target label is 6) for comparison. The experimental results show that the trigger size of label 6 can be found to be significantly smaller than the other labels, although the difference between the trigger sizes calculated by the reverse engineering algorithm from the plaintext and ciphertext is relatively large.



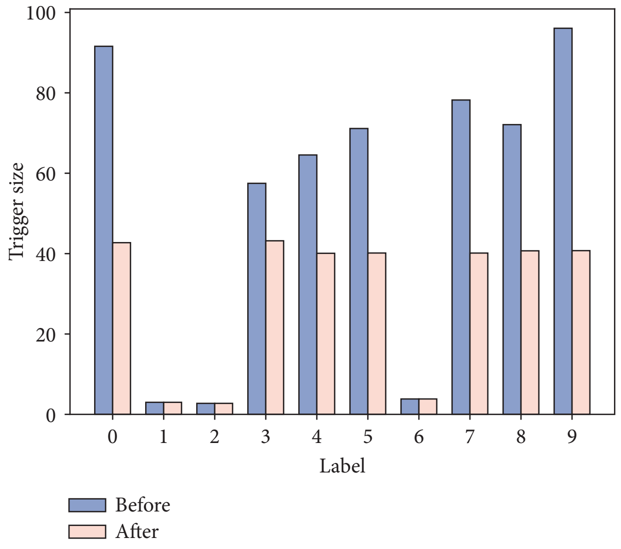
(a)



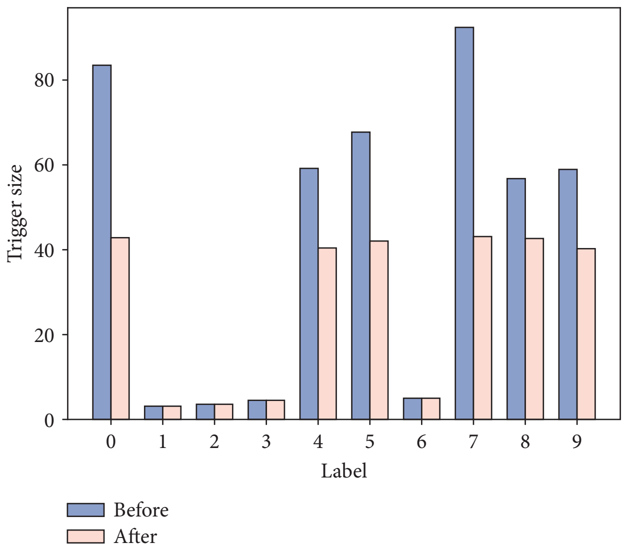
(b)



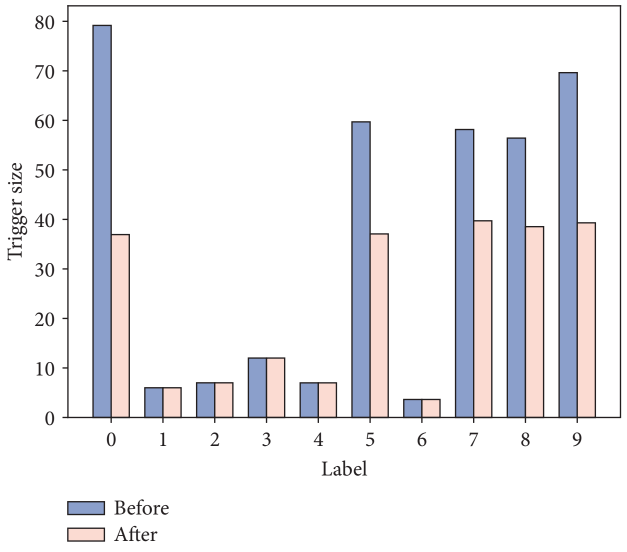
(c)



(d)

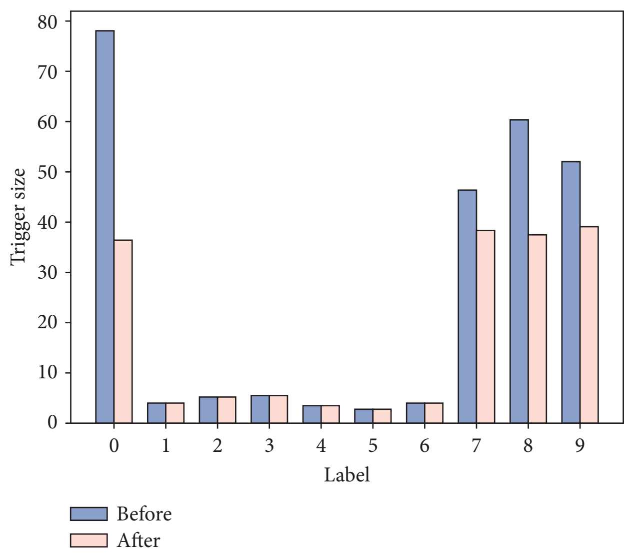


(e)

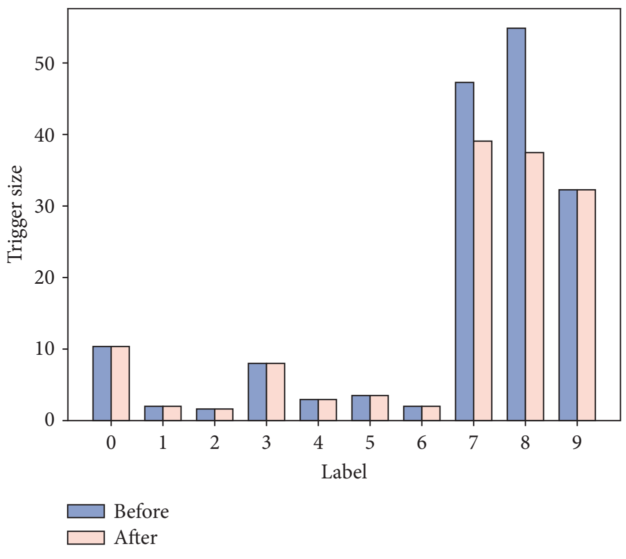


(f)

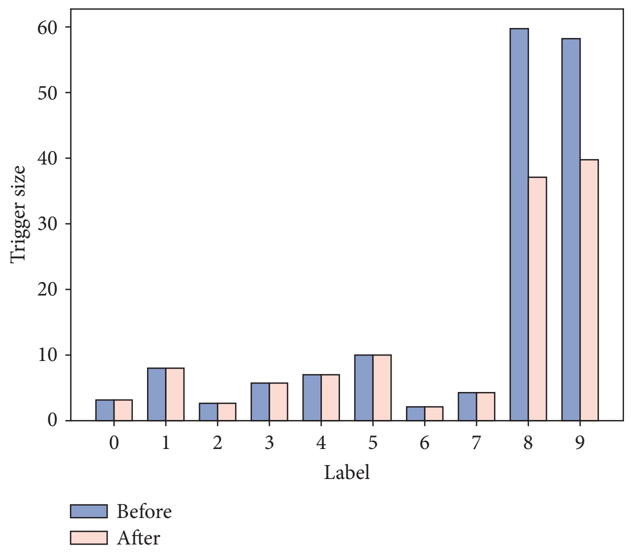
Figure 6: Continued.



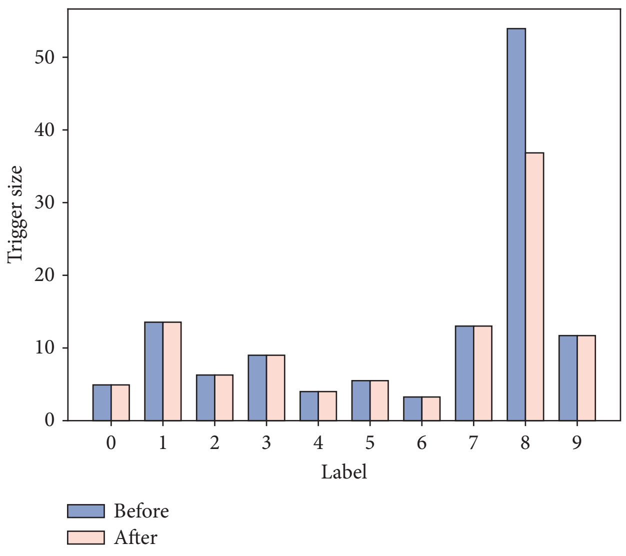
(g)



(h)



(i)



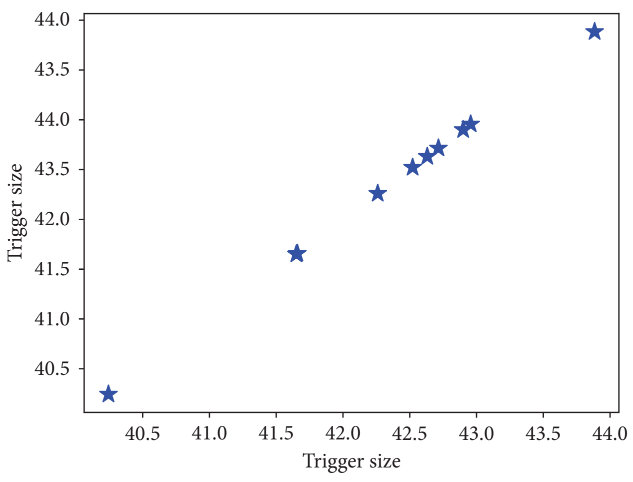
(j)

Figure 6: The trigger size before/after the backdoor identification algorithm on the MNIST dataset. (a) The number of attacked labels is 0. (b) The number of attacked labels is 1. (c) The number of attacked labels is 2. (d) The number of attacked labels is 3. (e) The number of attacked labels is 4. (f) The number of attacked labels is 5. (g) The number of attacked labels is 6. (h) The number of attacked labels is 7. (i) The number of attacked labels is 8. (j) The number of attacked labels is 9.

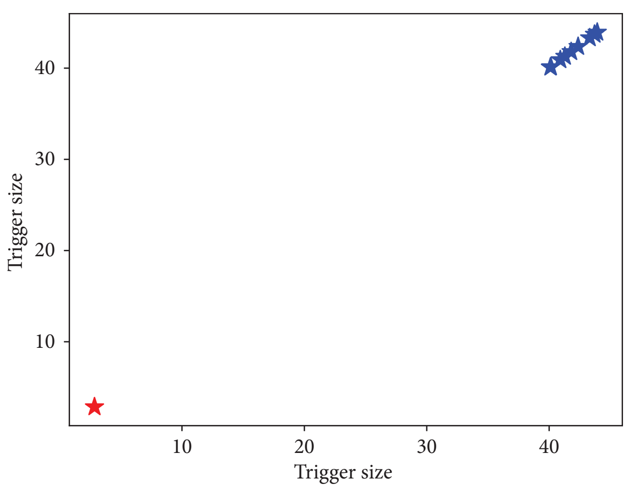
*5.1.3. The Efficiency of Backdoor Detection.* Table 3 compares the time cost of the reverse engineering algorithm in the ciphertext domain with different clean MNIST dataset sizes 6400, 1280, and 640 for reverse engineering, respectively. Compared with the plaintext [13], the time cost of performing reverse engineering algorithms in the ciphertext domain is greater according to the privacy requirement. Moreover, there is a significant communication cost in the ciphertext domain due to the three-party interaction required.

### *5.2. The Backdoor Identification*

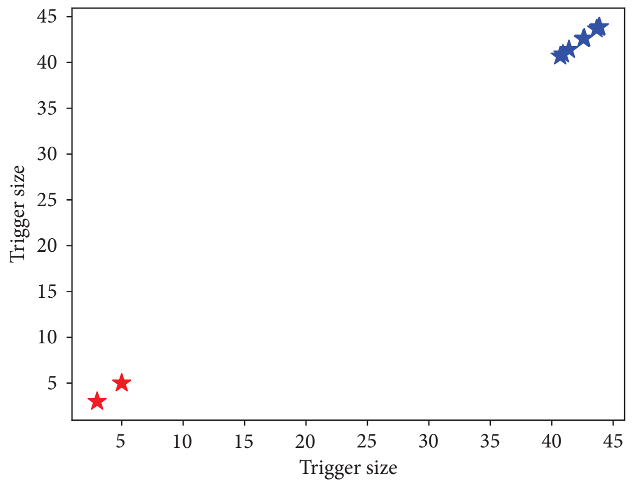
*5.2.1. Experiment Setup.* According to experience, we set $\eta$ to 3 in both plaintext and ciphertext domains. The value $\epsilon$ depends on the dataset used. In the plaintext domain, when the mean value of the triggers computed by the reverse engineering algorithm is greater than 50, we set $\epsilon$ to 10. Otherwise, $\epsilon$ is 5. Meanwhile, the value $\epsilon$ in the ciphertext domain is generally three times higher than that in the plaintext domain. Moreover, we consider the three of the



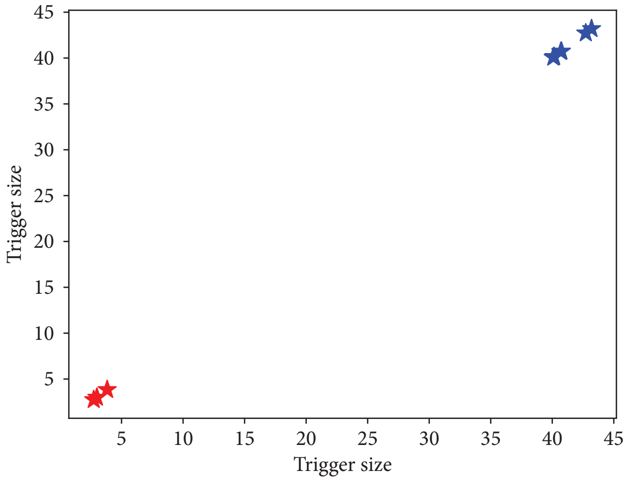
(a)



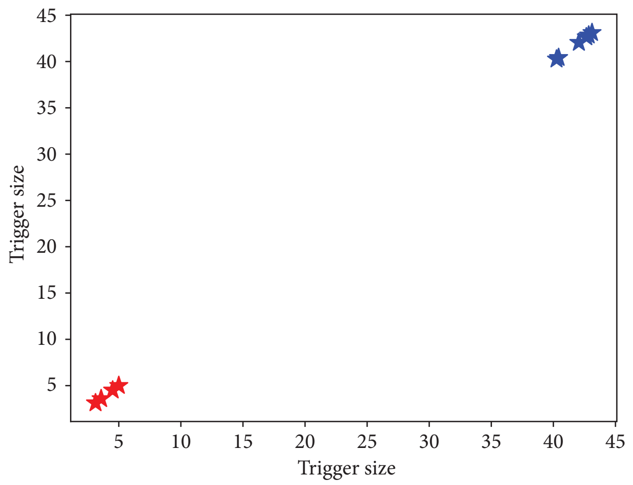
(b)



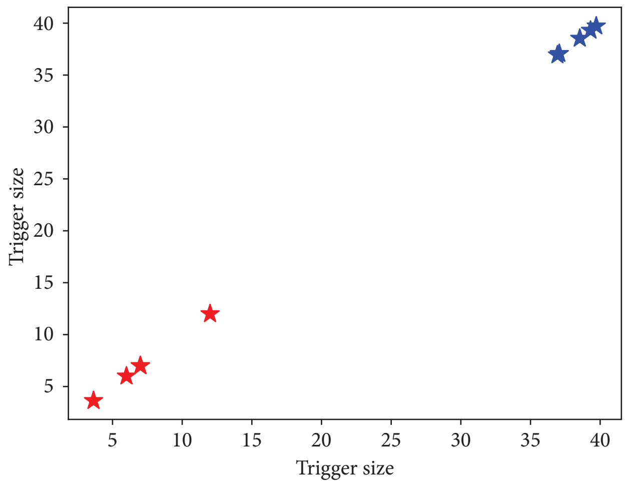
(c)



(d)

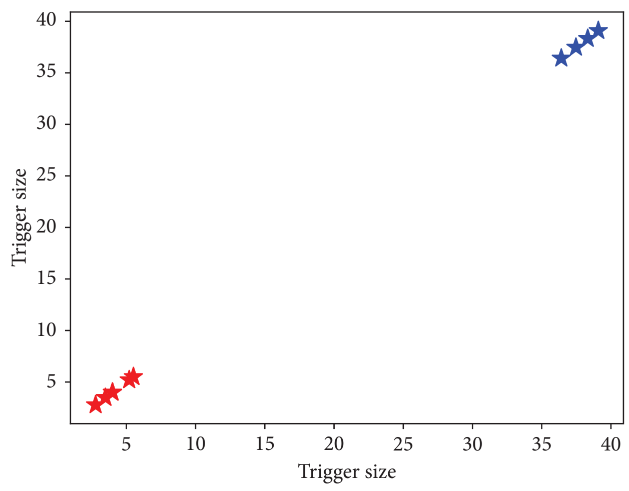


(e)

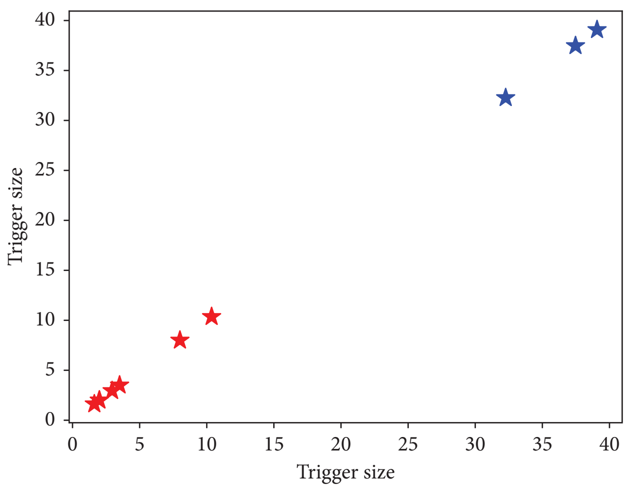


(f)

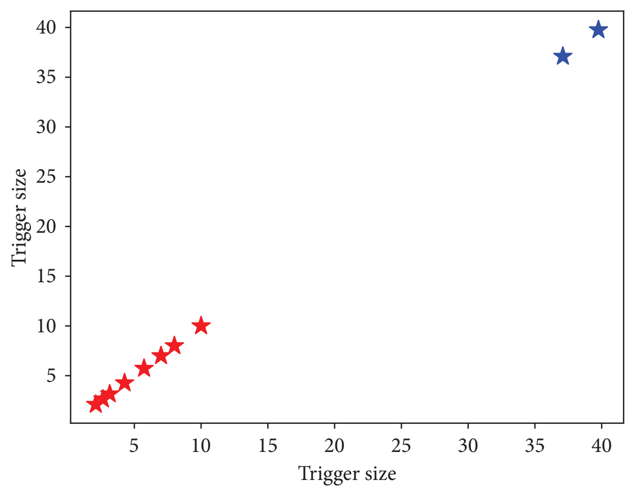
Figure 7: Continued.



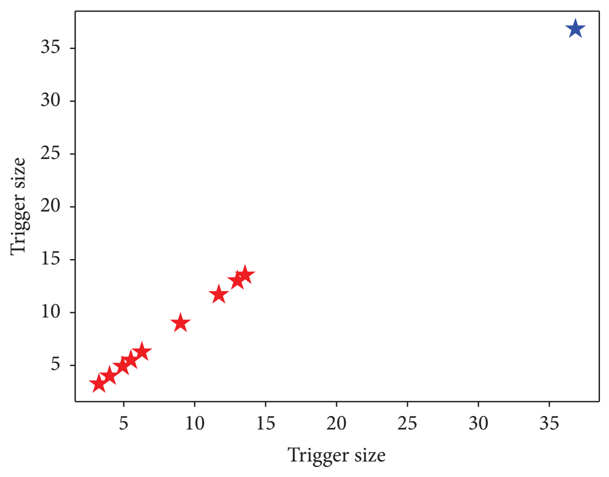
(g)



(h)



(i)



(j)

Figure 7: The identification results for different number of attacked labels. If the DNN model is not embedded with backdoors, the identification result is only one category (one color in the figure). Otherwise, the backdoors are embedded (two colors in the figure). (a) The number of attacked labels is 0. (b) The number of attacked labels is 1. (c) The number of attacked labels is 2. (d) The number of attacked labels is 3. (e) The number of attacked labels is 4. (f) The number of attacked labels is 5. (g) The number of attacked labels is 6. (h) The number of attacked labels is 7. (i) The number of attacked labels is 8. (j) The number of attacked labels is 9.

most common datasets to measure the effectiveness of backdoor identification algorithms. The network structure settings for these datasets are described as follows:

(i) MNIST. The DNN structure used for MNIST dataset is shown in Table 4. In the training phase, the base learning rate is set as 0.001, epochs are 10, the batch size is 32, and we use the Adam optimizer to optimize with a learning rate decay of $1 \times 10^{-5}$. The trained clean model achieved an accuracy of 99.25%. In this scenario, we use triggers that are $3 \times 3$ pixel squares.

(ii) SVHN [52]. We train a street view house number classifier on the DNN structure as in Table 5. We trained the model using the Adam optimizer with 15 epochs. The base learning rate is 0.001, the batch size is 32, and the learning rate decay is $1 \times 10^{-5}$. The final accuracy achieved for the trained clean model was 93.53%. Moreover, the trigger we used for this dataset is a $3 \times 3$ pixel square.

(iii) GTSRB [53]. We assume that the user specifies the structure of the trained deep neural network consists of 6 convolution layers and 2 dense layers, as shown in Table 6. We trained a traffic sign classifier using the same Adam classifier with 10 epochs. The base learning rate is 0.001, the batch size is 32, and the learning rate decay is $1 \times 10^{-5}$. The obtained clean model has a prediction accuracy of 96.53% on the test dataset. Moreover, the trigger we used for this dataset is a $3 \times 3$ pixel square.

*5.2.2. The Effectiveness of Backdoor Identification.* Figure 6 shows the trigger sizes for the different numbers of attacked labels before/after the backdoor identification

Table 4: The DNN structure used for MNIST dataset.

| Layer Type | # of Channels | Filter Size | Stride | Activation |
|---|---|---|---|---|
| Conv | 16 | $3\times3$ | 1 | ReLU |
| MaxPool | 16 | $2\times2$ | 2 | — |
| Conv | 32 | $3\times3$ | 1 | ReLU |
| MaxPool | 32 | $2\times2$ | 2 | — |
| FC | 512 | — | — | ReLU |
| FC | 10 | — | — | Softmax |

Table 5: The DNN structure used for SVHN dataset.

| Layer Type | # of Channels | Filter Size | Stride | Activation |
|---|---|---|---|---|
| Conv | 32 | $3\times3$ | 1 | ReLU |
| MaxPool | 32 | $2\times2$ | 2 | — |
| Conv | 64 | $3\times3$ | 1 | ReLU |
| MaxPool | 64 | $2\times2$ | 2 | — |
| FC | 512 | — | — | ReLU |
| FC | 10 | — | — | Softmax |

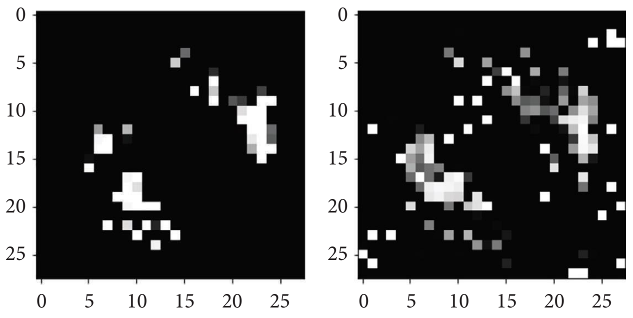
Table 6: The DNN structure used for GTSRB dataset.

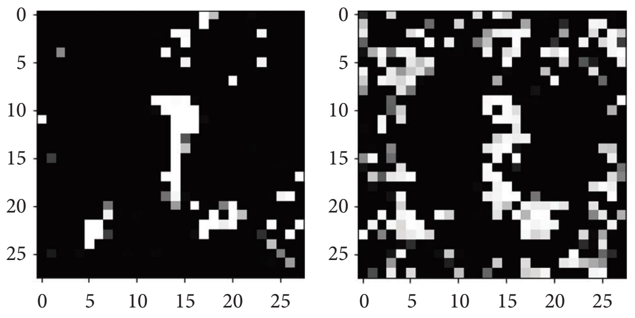
| Layer Type | # of Channels | Filter Size | Stride | Activation |
|---|---|---|---|---|
| Conv | 32 | $3\times3$ | 1 | ReLU |
| Conv | 32 | $3\times3$ | 1 | ReLU |
| MaxPool | 32 | $2\times2$ | 2 | — |
| Conv | 64 | $3\times3$ | 1 | ReLU |
| Conv | 64 | $3\times3$ | 1 | ReLU |
| MaxPool | 64 | $2\times2$ | 2 | — |
| Conv | 128 | $3\times3$ | 1 | ReLU |
| Conv | 128 | $3\times3$ | 1 | ReLU |
| MaxPool | 128 | $2\times2$ | 2 | — |
| FC | 512 | — | — | ReLU |
| FC | 43 | — | — | Softmax |

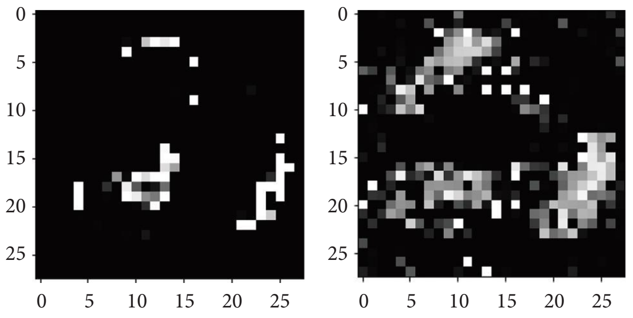
algorithm. In this section, we only show the trigger size calculated by the reverse engineering algorithm in plaintext for comparison. The location of the backdoor attack is shown in Figure 4(b). Experimental results show that the trigger size is smoother after our backdoor identification algorithm, thus greatly reducing the probability of the DBSCAN algorithm identifying the labels with large trigger sizes as outliers. In multi-label attacks, our backdoor identification algorithm does not process the trigger size of the normal label to be about the same size as the trigger of the attacked label, as shown in Figures 6(i) and 6(j).

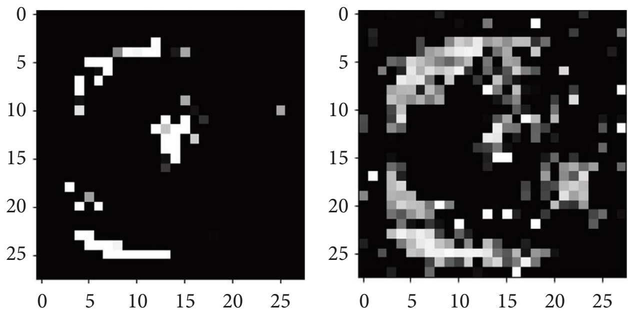
The identification results for different numbers of attacked labels are depicted in Figure 7. The backdoor identification algorithm classifies normal labels and target labels into two categories. It can identify the specific label being attacked by analyzing the values of the two categories. For the clean model, i.e., Figure 7(a), the classification result is for only one category. For the poisoned model, Algorithm 2 can classify the results into two categories, as shown in Figures 7(b)–7(j), where the blue color represents the normal label, while the red color represents the poisoned label. The experimental results show that our backdoor identification algorithm can effectively detect whether the model is injected with a backdoor and can identify the specific attacked labels.

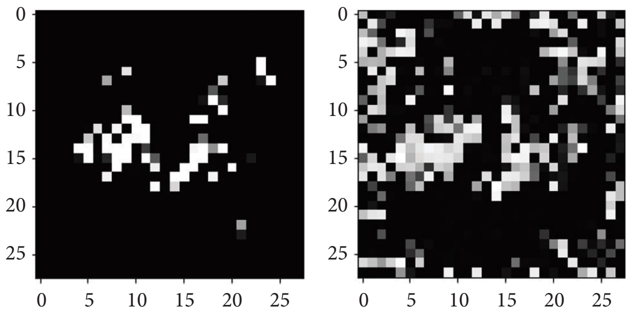
*5.2.3. Comparison with NC.* To further compare with the most related work [13], Figure 8 gives the results after recovering triggers in the plaintext domain (NC [13]) and in the ciphertext domain (this paper) on the MNIST dataset [13]. We can observe from Figure 8 that NC [13] and our algorithm can detect the target labels in the plaintext and ciphertext domains, respectively. Figure 8(g) shows the trigger corresponding to the target label 6 recovered by the reverse engineering algorithm, where the result in the plaintext domain is shown on the left, and the result in the ciphertext domain is shown on the right. Although the triggers recovered from the ciphertext domain are more complicated than those recovered from the plaintext domain, we can get the same results. That is, the size of the recovered trigger corresponding to the target label is much smaller than that of other normal labels.

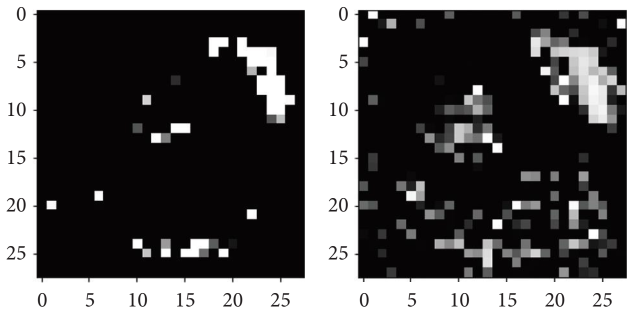

(a)

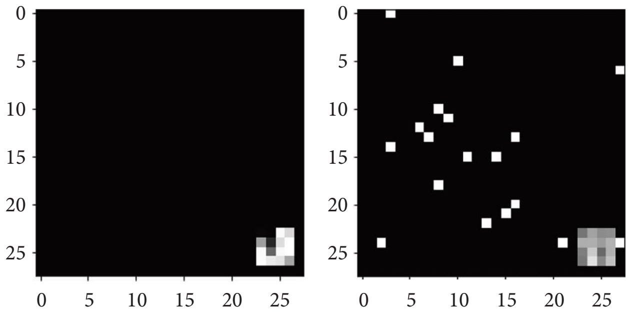

(b)


(c)


(d)


(e)


(f)


(g)

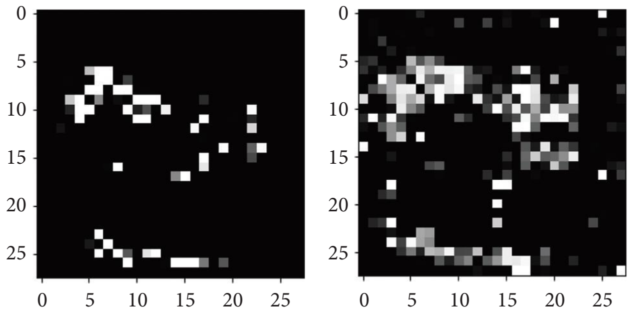
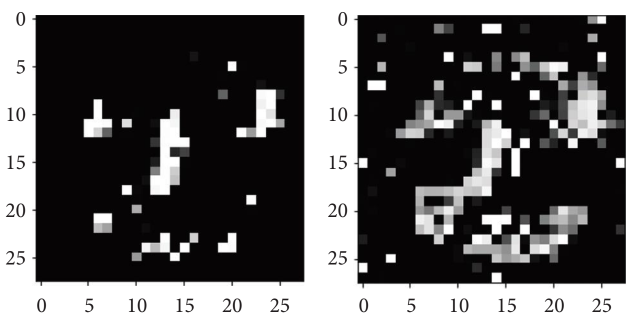

(h)

Figure 8: Continued.


(i)

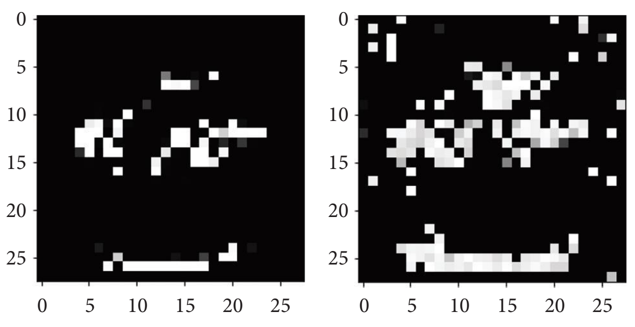

(j)

Figure 8: Triggers recovered by reverse engineering algorithm on MNIST data. The left side of each subgraph is recovered in the plaintext domain (NC [13]), while the right side of each subgraph is recovered in the ciphertext domain (this paper), which is close to NC [13]. Label 6 is the target label. (a) Label 0. (b) Label 1. (c) Label 2. (d) Label 3. (e) Label 4. (f) Label 5. (g) Label 6. (h) Label 7. (i) Label 8. (j) Label 9.

Table 7: Comparison on MNIST, SVHN, and GTSRB datasets.

| Dataset | Target labels | Attack location | Classification accuracy (%) | Attac success rate (%) | Backdoor detection and identification NC [13] | Ours |
|---|---|---|---|---|---|---|
| MNIST | – | – | 99.25% | — | 2 | — |
| | 6 | 9 | 9.30% | 100.00% | 6 | 6 |
| | 6, 1 | 9, 1 | 99.35% | 100.00% | 6, 1 | 6, 1 |
| | 6, 1, 2 | 9, 1, 7 | 99.38% | 100.00% | 6, 1,2 | 6, 1, 2 |
| | 6, 1, 2, 3 | 9, 1, 7, 6 | 99.34% | 100.00% | — | 6, 1, 2, 3 |
| | 6, 1, 2, 3, 4 | 9, 1, 7, 6, 2 | 99.31% | 100.00% | — | 6, 1, 2, 3, 4 |
| | 6, 1, 2, 3, 4, 5 | 9, 1, 7, 6, 2, 3 | 99.27% | 100.00% | — | 6, 1, 2, 3, 4, 5 |
| | 6, 1, 2, 3, 4, 5, 0 | 9, 1, 7, 6, 2, 3, 5 | 99.27% | 100.00% | — | 6, 1, 2, 3, 4, 5, 0 |
| | 6, 1, 2, 3, 4, 5, 0, 7 | 9, 1, 7, 6, 2, 3, 5, 8 | 99.29% | 100.00% | — | 6, 1 2, 3, 4, 5, 0, 7 |
| | 6, 1, 2, 3, 4, 5, 0, 7, 9 | 9, 1, 7, 6, 2, 3, 5, 8, 4 | 99.20% | 100.00% | — | 6, 1, 2, 3, 4, 5, 0, 7, 9 |
| SVHN | — | — | 93.53% | — | — | — |
| | 0 | | 94.56% | 98.67% | 0 | 0 |
| | 0, 1 | 1, 2 | 94.25% | 98.97% | 0, 1 | 0, 1 |
| | 0, 1, 2 | 1, 2, 3 | 94.26% | 99.03% | — | 0, 1, 2 |
| | 0, 1, 2, 3 | 1, 2, 3, 4 | 94.11% | 99.39% | — | 0, 1, 2, 3 |
| | 0, 1, 2, 3, 4 | 1, 2, 3, 4, 5 | 94.25% | 99.16% | — | 0, 1, 2, 3, 4 |
| | 0, 1, 2, 3, 4, 5 | 1, 2, 3, 4, 5, 6 | 94.17% | 98.98% | — | 0, 1, 2, 3, 4, 5 |
| | 0, 1, 2, 3, 4, 5, 6 | 1, 2, 3, 4, 5, 6, 7 | 94.11% | 99.25% | — | 0, 1, 2, 3, 4, 5, 6 |
| | 0, 1, 2, 3, 4, 5, 6, 8 | 1, 2, 3, 4, 5, 6, 7, 8 | 93.78% | 99.36% | — | 0, 1, 2, 3, 4, 5, 6, 8 |
| | 0, 1, 2, 3, 4, 5, 6, 8, 9 | 1, 2, 3, 4, 5, 6, 7, 8, 9 | 93.83% | 99.25% | — | 0, 1, 2, 3, 4, 5, 6, 8, 9 |
| GTSRB | — | — | 96.53% | — | — | — |
| | 28 | 1 | 96.54% | 92.64% | 28 | 28 |
| | 28, 33 | 1, 2 | 97.09% | 96.00% | 28, 33, 20 | 28, 33 |
| | 28, 33, 1 | 1, 2, 3 | 96.63% | 95.36% | 28, 33, 1 | 28, 33, 1 |
| | 28, 33, 1, 7 | 1, 2, 3, 4 | 95.75% | 95.97% | 28, 33, 1, 7 | 28, 33, 1, 7 |
| | 28, 33, 1, 7, 11 | 1, 2, 3, 4, 5 | 95.61% | 95.50% | 28, 33, 1, 7, 11 | 28, 33, 1, 7, 11 |
| | 28, 33, 1, 7, 11, 16 | 1, 2, 3, 4, 5, 6 | 96.04% | 96.27% | 28, 33, 1, 7, 11, 16 | 28, 33, 1, 7, 11, 16 |
| | 28, 33, 1, 7, 11, 16, 23 | 1, 2, 3, 4, 5, 6, 7 | 96.15% | 96.03% | 28, 33, 1, 7, 11, 16, 23 | 28, 33, 1, 7, 11, 16, 23 |
| | 28, 33, 1, 7, 11, 16, 23, 39 | 1, 2, 3, 4, 5, 6, 7, 8 | 95.39% | 97.08% | 28, 33, 1, 7, 11, 16, 23, 39 | 28, 33, 1, 7, 11, 16, 23, 39 |
| | 28, 33, 1, 7, 11, 16, 23, 39, 42 | 1, 2, 3, 4, 5, 6, 7, 8, 9 | 96.05% | 97.59% | 28, 33, 1, 7, 11, 16, 23, 39, 42 | 28, 33, 1, 7, 11, 16, 23, 39, 42 |

*Note.* Target labels denote the target labels of the attack. Attack location denotes the trigger location of the attack target labels, as shown in Figure 4(b). Classification accuracy and Attack success rate are the classification accuracies for the clean and poisoned images. The last two columns are the target labels detected and identified by related work NC [13] and our algorithm.

Table 7 demonstrates the effectiveness of the backdoor identification algorithm on the MNIST, SVHN, and GTSRB datasets. We suppose to attack each of the 9 different labels of the above datasets. The experiments show that the classification accuracy and the attack success rate are at a high level. Moreover, NC [13] and our algorithm both detect and identify the clean model and the poisoned model by single-label attacks, whereas NC [13] fails in the multi-label attacks by regarding the poisoned model as a clean model.

## 6. Conclusions

This paper presents a framework, DeepGuard, which considers both privacy-preserving and backdoor defense for DNNs in an outsourced cloud environment. We design an effective and efficient reverse engineering algorithm that enables to keep the confidentiality of the data and model parameters in the DNN training. We also propose a practical backdoor identification algorithm that achieves to detect both single-label and multi-label attacks. Finally, the extensive experiments on the various datasets validate the effectiveness and efficiency of our backdoor detection and identification algorithm.

In future, we will pay attention to the backdoor mitigation in the ciphertext domain, which is quite different from that in plaintext due to privacy-preserving DNNs. It is challenging to implement the existing backdoor mitigation scheme for the plaintext domain directly to those in the ciphertext domain. We believe that the efficient privacy-preserving backdoor detection, identification, and mitigation framework in ciphertext will be the future work in the research direction.

## Data Availability

We use the public datasets such as MNIST, SVHN, and GTSRB.

## Conflicts of Interest

The authors declare that they do not have any conflicts of interest.

## Acknowledgments

This work is supported in part by National Natural Science Foundation of China under Grant 61972241 and 61972094, in part by Natural Science Foundation of Shanghai under Grant 22ZR1427100 and 18ZR1417300, and in part by Luo-Zhaorao College Student Science and Technology Innovation Foundation of Shanghai Ocean University.

## References

[1] A. H. Ribeiro, M. H. Ribeiro, G. M. M. Paixão et al., "Automatic diagnosis of the 12-lead ecg using a deep neural network," *Nature Communications*, vol. 11, no. 1, pp. 1760–1769, 2020.

[2] M. Zhang, Y. Zhang, L. Zhang, C. Liu, and S. Khurshid, "Deeproad: Gan-based metamorphic testing and input validation framework for autonomous driving systems," in *Proceedings of the 2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 132–142, IEEE, Montpellier, France, September 2018.

[3] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song, "Sphereface: deep hypersphere embedding for face recognition," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 212–220, HI, USA, July 2017.

[4] X. Zhu, C. Vondrick, C. C. Fowlkes, and D. Ramanan, "Do we need more training data?" *International Journal of Computer Vision*, vol. 119, no. 1, pp. 76–92, 2016.

[5] Y. Li, B. Wu, Y. Jiang, Z. Li, and S. Xia, "Backdoor learning: a survey," *CoRR*, vol. abs/2007, Article ID 08745, 2020.

[6] W. Lifei, C. Congcong, Z. Lei, L. Mengsi, C. Yujiao, and W. Qin, "Security issues and privacy preserving in machine learning," *Journal of Computer Research and Development*, vol. 57, no. 10, p. 2066, 2020.

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

[8] A. Saha, A. Subramanya, and H. Pirsiavash, "Hidden trigger backdoor attacks," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 7, pp. 11957–11965, 2020.

[9] X. Gong, Y. Chen, Q. Wang et al., "Defense-resistant backdoor attacks against deep neural networks in outsourced cloud environment," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 8, pp. 2617–2631, 2021.

[10] S. Koffas, J. Xu, M. Conti, and S. Picek, "Can you hear it? backdoor attacks via ultrasonic triggers," *CoRR*, vol. abs/2107, Article ID 14569, 2021.

[11] Y. Liu, S. Ma, Y. Aafer, W. C. Lee, and X. Zhang, "Trojaning attack on neural networks," in *Network and Distributed System Security Symposium*, 2017.

[12] X. Chen, A. Salem, M. Backes, S. Ma, and Y. Zhang, "Badnl: backdoor attacks against nlp models," in *Proceedings of the ICML 2021 Workshop on Adversarial Machine Learning*, NY, USA, December 2021.

[13] B. Wang, Y. Yao, S. Shan et al., "Neural cleanse: identifying and mitigating backdoor attacks in neural networks," in *Proceedings of the 2019 IEEE Symposium on Security and Privacy (SP)*, pp. 707–723, IEEE, CA, USA, May 2019.

[14] Y. Liu, W.-C. Lee, G. Tao, S. Ma, Y. Aafer, and X. Zhang, "Abs: scanning neural networks for back-doors by artificial brain stimulation," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1265–1282, London, UK, November 2019.

[15] Y. Gao, C. Xu, D. Wang, S. Chen, D. C. Ranasinghe, and S. Nepal, "Strip: a defence against trojan attacks on deep neural networks," in *Proceedings of the 35th Annual Computer Security Applications Conference*, pp. 113–125, San Juan Puerto Rico USA, December 2019.

[16] A. C. Yao, "Protocols for secure computations," in *23rd Annual Symposium on Foundations of Computer Science (Sfcs 1982)*, pp. 160–164, IEEE, 1982.

[17] P. Mohassel and Y. Zhang, "Secureml a system for scalable privacy-preserving machine learning," in *Proceedings of the 2017 IEEE symposium on security and privacy SP*, pp. 19–38, IEEE, CA, USA, May 2017.

[18] M. S. Riazi, C. Weinert, O. Tkachenko, E. M. Songhori, T. Schneider, and F. Koushanfar, "Chameleon: a hybrid secure computation framework for machine learning applications," in *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, pp. 707–721, Incheon Republic of Korea, June 2018.

[19] P. Mohassel and P. Rindal, "Aby3: a mixed protocol framework for machine learning," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 35–52, Toronto Canada, October 2018.

[20] S. Wagh, S. Tople, F. Benhamouda, E. Kushilevitz, P. Mittal, and T. Rabin, "Falcon: honest-majority maliciously secure framework for private deep learning," *Proceedings on Privacy Enhancing Technologies*, vol. 2021, no. 1, pp. 188–208, 2021.

[21] M. Keller, "Mp-spdz: a versatile framework for multi-party computation," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1575–1590, USA, October 2020.

[22] E. Bagdasaryan and V. Shmatikov, "Blind backdoors in deep learning models," in *Proceedings of the USENIX Security Symposium*, pp. 1505–1521, USENIX Association, Vancouver, Canada, August 2021.

[23] R. Shokri, "Bypassing backdoor detection algorithms in deep learning," in *Proceedings of the 2020 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 175–183, IEEE, Genoa, Italy, September 2020.

[24] Y. Liu, X. Ma, J. Bailey, and F. Lu, "Reflection backdoor: a natural backdoor attack on deep neural networks," in *European Conference on Computer Vision*, pp. 182–199, Springer, 2020.

[25] A. Salem, M. Backes, and Y. Zhang, "Don't trigger me! A triggerless backdoor attack against deep neural networks," *CoRR*, vol. abs/2010, Article ID 03282, 2020.

[26] Y. Yao, H. Li, H. Zheng, and B. Y. Zhao, "Latent backdoor attacks on deep neural networks," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pp. 2041–2055, London, UK, November 2019.

[27] K. Liu, B. Dolan-Gavitt, and S. Garg, "Fine-pruning: defending against backdooring attacks on deep neural networks," in *Proceedings of the International Symposium on Research in Attacks, Intrusions, and Defenses*, pp. 273–294, Springer, Heraklion, Crete, Greece, September 2018.

[28] H. Chen, C. Fu, J. Zhao, and F. Koushanfar, *Deepinspect: A Black-Box Trojan Detection and Mitigation Framework for Deep Neural Networks*, pp. 4658–4664, IJCAI, Vienna, Austria, 2019.

[29] M. Du, R. Jia, and D. Song, "Robust anomaly detection and backdoor attack detection via differential privacy," 2019, https://arxiv.org/abs/1911.07116.

[30] H. Qiu, Y. Zeng, S. Guo, T. Zhang, M. Qiu, and B. Thuraisingham, "Deepsweep: an evaluation framework for mitigating dnn backdoor attacks using data augmentation," in *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*, pp. 363–377, Hongkong, China, June 2021.

[31] G. Shen, Y. Liu, G. Tao et al., "Backdoor scanning for deep neural networks through k-arm optimization," in *Proceedings of the International Conference on Machine Learning. PMLR*, pp. 9525–9536, Baltimore MD, February 2021.

[32] Y. Wang, W. Li, E. Sarkar, M. Shafique, M. Maniatakos, and S. E. Jabari, "Pidan: a coherence optimization approach for backdoor attack detection and mitigation in deep neural networks," 2022, https://arxiv.org/abs/2203.09289.

[33] D. Demmler, T. Schneider, and M. Zohner, *Aby-a Framework for Efficient Mixed-Protocol Secure Two-Party Computation*NDSS, Chennai, India, 2015.

[34] X. Zhang, X. Chen, J. K. Liu, and Y. Xiang, "Deeppar and deepdpa: privacy preserving and asynchronous deep learning for industrial iot," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 3, pp. 2081–2090, 2020.

[35] H. Chaudhari, A. Choudhury, A. Patra, and A. Suresh, "Astra: high throughput 3pc over rings with application to secure prediction," in *Proceedings of the 2019 ACM SIGSAC Conference on Cloud Computing Security Workshop*, pp. 81–92, London, UK, November 2019.

[36] M. Byali, H. Chaudhari, A. Patra, and A. Suresh, "Flash: fast and robust framework for privacy-preserving machine learning," *Proceedings on Privacy Enhancing Technologies*, vol. 2020, no. 2, pp. 459–480, 2020.

[37] H. Chaudhari, R. Rachuri, and A. Suresh, "Trident: efficient 4pc framework for privacy preserving machine learning," NDSS. The Internet Society, Chennai, India, 2020.

[38] A. Patra, T. Schneider, A. Suresh, and H. Yalame, "Aby2.0: improved mixed-protocol secure two-party computation," in *Proceedings of the 30th {USENIX} Security Symposium {USENIX} Security*, Canada, USA, November 2021.

[39] E. Chou, F. Tramèr, and G. Pellegrino, *Sentinet: Detecting Localized Universal Attacks against Deep Learning Systems*, S. P. Workshops, Ed., pp. 48–54, IEEE, 2020.

[40] S. Ma and Y. Liu, "Nic: detecting adversarial samples with neural network invariant checking," in *Proceedings of the 26th Network and Distributed System Security Symposium (NDSS 2019)*, China, July 2019.

[41] B. Chen, W. Carvalho, N. Baracaldo et al., "Detecting backdoor attacks on deep neural networks by activation clustering," in *SafeAI@AAAI, ser. CEUR Workshop Proceedings*vol. 2301, 2019.

[42] W. Guo, L. Wang, X. Xing, M. Du, and D. Song, "TABOR: a highly accurate approach to inspecting and restoring trojan backdoors in AI systems," *CoRR*, vol. abs/1908, Article ID 01763, 2019.

[43] S. Shen, S. Tople, and P. Saxena, "Auror: defending against poisoning attacks in collaborative deep learning systems," in *Proceedings of the 32nd Annual Conference on Computer Security Applications*, pp. 508–519, USA, December 2016.

[44] T. D. Nguyen, P. Rieger, H. Yalame et al., "FLGUARD: secure and private federated learning," *CoRR*, vol. abs/2101, Article ID 02281, 2021.

[45] Z. Zhang, J. Li, S. Yu, and C. Makaya, "Safelearning: enable backdoor detectability in federated learning with secure aggregation," *CoRR*, vol. abs/2102, Article ID 02402, 2021.

[46] C. Xie, M. Chen, P. Chen, and B. Li, "CRFL: certifiably robust federated learning against backdoor attacks," in *ICML, ser. Proceedings of Machine Learning Research*vol. 139, pp. 11372–11382, PMLR, 2021.

[47] C. Chen, L. Wei, L. Zhang, and J. Ning, "Mp-badnet: a backdoor-attack detection and identification protocol among multi-participants in private deep neural networks," in *Proceedings of the ACM Turing Award Celebration Conference-China (ACM TURC 2021)*, pp. 104–109, China, July 2021.

[48] T. Araki, J. Furukawa, Y. Lindell, A. Nof, and K. Ohara, "High-throughput semi-honest secure three-party

computation with an honest majority," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 805–817, France, April 2016.

[49] D. P. Kingma and J. Ba, *Adam: a method for stochastic optimization*ICLR, ND, India, 2015.

[50] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," *Kdd*, vol. 96, no. 34, pp. 226–231, 1996.

[51] L. Deng, "The mnist database of handwritten digit images for machine learning research [best of the web]," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141-142, 2012.

[52] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, vol. 2011, 2011, http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf.

[53] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "Man vs. computer: benchmarking machine learning algorithms for traffic sign recognition," *Neural Networks*, vol. 32, pp. 323–332, 2012, https://www.sciencedirect.com/science/article/pii/S0893608012000457.